# MC9S12ZVHY-Family
# Reference Manual

***HCS12***
***Microcontrollers***

MC9S12ZVHYRMV1
Rev. 1.00
09/2013

*freescale.com*

freescale

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

freescale.com

A full list of family members and options is included in the appendices.

The following revision history table summarizes changes contained in this document.

This document contains information for all constituent modules, with the exception of the S12Z CPU. For S12ZCPU information please refer to the CPU S12Z Reference Manual.

## Revision History

| Date | Revision Level | Description |
|---|---|---|
| Seq, 2013 | 1.00 | First release for external website<br>- Add ZVHY 32K<br>- Update OSC etc electrical spec |

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

# Chapter 1
# Device Overview MC9S12ZVHY-Family


1.1 Introduction . . . 21
1.2 Features . . . 22
1.2.1 MC9S12ZVHY-Family Member Comparison . . . 22
1.3 Maskset 0N39G and 1N39G device compare . . . 23
1.4 Chip-Level Features . . . 23
1.5 Module Features . . . 24
1.5.1 S12Z Central Processor Unit (CPU) . . . 24
1.5.2 Embedded Memory . . . 25
1.5.3 Clocks, Reset & Power Management Unit (CPMU) . . . 26
1.5.4 Main External Oscillator (XOSCLCP) . . . 27
1.5.5 32K External Oscillator . . . 27
1.5.6 System Integrity Support . . . 27
1.5.7 Real Time Clock (RTC) . . . 27
1.5.8 Timer (TIM) . . . 28
1.5.9 Pulse Width Modulation Module (PWM) . . . 28
1.5.10 Simple Sound Generator (SSG) . . . 28
1.5.11 Liquid Crystal Display driver (LCD) . . . 28
1.5.12 Stepper Motor Controller (MC) . . . 28
1.5.13 Stepper Stall Detect (SSD) . . . 29
1.5.14 Multi-Scalable Controller Area Network (MSCAN) . . . 29
1.5.15 Inter-IC Bus Module (IIC) . . . 29
1.5.16 Serial Communication Interface Module (SCI) . . . 30
1.5.17 Serial Peripheral Interface Module (SPI) . . . 30
1.5.18 Analog-to-Digital Converter Module (ADC) . . . 30
1.5.19 Supply Voltage Sensor (BATS) . . . 30
1.5.20 On-Chip Voltage Regulator system (VREG) . . . 30
1.6 Block Diagram . . . 32
1.6.1 Device Memory Map . . . 33
1.6.2 Part ID registers Assignments . . . 36
1.7 Signal Description and Device Pinouts . . . 36
1.7.1 Pin Assignment Overview . . . 36
1.7.2 Detailed Signal Descriptions . . . 37
1.7.3 VSENSE - Voltage Sensor Input . . . 43
1.7.4 BCTL . . . 43
1.7.5 Power Supply Pins . . . 44
1.8 Package and Pinouts . . . 45
1.9 Modes of Operation . . . 54
1.9.1 Chip Configuration Modes . . . 54
1.9.2 Debugging Modes . . . 55
1.9.3 Low Power Modes . . . 55
1.10 Security . . . 56
1.10.1 Features . . . 56
1.10.2 Securing the Microcontroller . . . 56

1.10.3 Operation of the Secured Microcontroller . . . 57
1.10.4 Unsecuring the Microcontroller . . . 57
1.10.5 Reprogramming the Security Bits . . . 58
1.10.6 Complete Memory Erase . . . 58
1.11 Resets and Interrupts . . . 59
1.11.1 Resets . . . 59
1.11.2 Interrupt Vectors . . . 59
1.11.3 Effects of Reset . . . 62
1.12 COP Configuration . . . 63
1.13 ADC0 Internal Channels . . . 63
1.14 The ADC0 VRH/VRL . . . 64
1.15 The ADC0 Conversion Resolution . . . 64
1.16 ADC Result Reference . . . 64
1.17 BDC Clock Source Connectivity . . . 65
1.18 FTMRZ Connectivity . . . 65
1.19 RTC Clock Source . . . 65
1.20 LCD Clock Source Connectivity . . . 66
1.21 32K OSC enable control . . . 66

## Chapter 2
## Port Integration Module (S12ZVHYPIMV1)

2.1 Introduction . . . 67
2.1.1 Overview . . . 67
2.1.2 Features . . . 68
2.2 External Signal Description . . . 69
2.3 Memory Map and Register Definition . . . 77
2.3.1 Register Map . . . 78
2.3.2 Register Descriptions . . . 85
2.4 Functional Description . . . 100
2.4.1 General . . . 100
2.4.2 Registers . . . 100
2.4.3 Interrupts . . . 104
2.4.4 Pin interrupts and Wakeup . . . 104
2.5 Initialization and Application Information . . . 105
2.5.1 Port Data and Data Direction Register writes . . . 105
2.5.2 SCI0,1 Baud Rate Detection . . . 105
2.5.3 RTC on chip calibration . . . 106
2.5.4 RTC off chip calibration . . . 106

## Chapter 3
## Memory Mapping Control (S12ZMMCV1)

3.1 Introduction . . . 107
3.1.1 Glossary . . . 108
3.1.2 Overview . . . 108

3.1.3 Features . . . 108
3.1.4 Modes of Operation . . . 108
3.1.5 Block Diagram . . . 109
3.2 External Signal Description . . . 109
3.3 Memory Map and Register Definition . . . 109
3.3.1 Memory Map . . . 109
3.3.2 Register Descriptions . . . 110
3.4 Functional Description . . . 115
3.4.1 Global Memory Map . . . 115
3.4.2 Illegal Accesses . . . 117
3.4.3 Uncorrectable ECC Faults . . . 118

## Chapter 4
## Interrupt (S12ZINTV0)

4.1 Introduction . . . 119
4.1.1 Glossary . . . 120
4.1.2 Features . . . 120
4.1.3 Modes of Operation . . . 121
4.1.4 Block Diagram . . . 121
4.2 External Signal Description . . . 122
4.3 Memory Map and Register Definition . . . 122
4.3.1 Module Memory Map . . . 122
4.3.2 Register Descriptions . . . 123
4.4 Functional Description . . . 127
4.4.1 S12Z Exception Requests . . . 128
4.4.2 Interrupt Prioritization . . . 128
4.4.3 Priority Decoder . . . 129
4.4.4 Reset Exception Requests . . . 129
4.4.5 Exception Priority . . . 129
4.4.6 Interrupt Vector Table Layout . . . 130
4.5 Initialization/Application Information . . . 130
4.5.1 Initialization . . . 130
4.5.2 Interrupt Nesting . . . 130
4.5.3 Wake Up from Stop or Wait Mode . . . 131

## Chapter 5
## Background Debug Controller (S12ZBDCV2)

5.1 Introduction . . . 133
5.1.1 Glossary . . . 133
5.1.2 Features . . . 134
5.1.3 Modes of Operation . . . 134
5.1.4 Block Diagram . . . 136
5.2 External Signal Description . . . 137
5.3 Memory Map and Register Definition . . . 137

5.3.1 Module Memory Map . . . 137
5.3.2 Register Descriptions . . . 138
5.4 Functional Description . . . 142
5.4.1 Security . . . 142
5.4.2 Enabling BDC And Entering Active BDM . . . 142
5.4.3 Clock Source . . . 143
5.4.4 BDC Commands . . . 143
5.4.5 BDC Access Of Internal Resources . . . 159
5.4.6 BDC Serial Interface . . . 162
5.4.7 Serial Interface Hardware Handshake (ACK Pulse) Protocol . . . 165
5.4.8 Hardware Handshake Abort Procedure . . . 167
5.4.9 Hardware Handshake Disabled (ACK Pulse Disabled) . . . 168
5.4.10 Single Stepping . . . 169
5.4.11 Serial Communication Timeout . . . 170
5.5 Application Information . . . 170
5.5.1 Clock Frequency Considerations . . . 170

## Chapter 6
## S12Z Debug (S12ZDBGV2) Module

6.1 Introduction . . . 171
6.1.1 Glossary . . . 171
6.1.2 Overview . . . 172
6.1.3 Features . . . 172
6.1.4 Modes of Operation . . . 173
6.1.5 Block Diagram . . . 173
6.2 External Signal Description . . . 174
6.2.1 External Event Input . . . 174
6.2.2 Profiling Output . . . 174
6.3 Memory Map and Registers . . . 174
6.3.1 Module Memory Map . . . 174
6.3.2 Register Descriptions . . . 177
6.4 Functional Description . . . 198
6.4.1 DBG Operation . . . 198
6.4.2 Comparator Modes . . . 198
6.4.3 Events . . . 202
6.4.4 State Sequence Control . . . 204
6.4.5 Trace Buffer Operation . . . 205
6.4.6 Code Profiling . . . 214
6.4.7 Breakpoints . . . 218
6.5 Application Information . . . 219
6.5.1 Avoiding Unintended Breakpoint Re-triggering . . . 219
6.5.2 Debugging Through Reset . . . 219
6.5.3 Breakpoints from other S12Z sources . . . 220
6.5.4 Code Profiling . . . 220

# Chapter 7
# S12 Clock, Reset and Power Management Unit (S12CPMU_UHV_V5)


7.1 Introduction . . . 223
7.1.1 Features . . . 224
7.1.2 Modes of Operation . . . 226
7.1.3 S12CPMU_UHV_V5 Block Diagram . . . 229
7.2 Signal Description . . . 231
7.2.1 RESET . . . 231
7.2.2 EXTAL and XTAL . . . 231
7.2.3 VSUP — Regulator Power Input Pin . . . 231
7.2.4 VDDA, VSSA — Regulator Reference Supply Pins . . . 231
7.2.5 VDDX, VSSX — Pad Supply Pins . . . 231
7.2.6 BCTL — Base Control Pin for external PNP . . . 232
7.2.7 VSS — Core Logic Ground Pin . . . 232
7.2.8 VDD — Internal Regulator Output Supply (Core Logic) . . . 232
7.2.9 VDDF — Internal Regulator Output Supply (NVM Logic) . . . 232
7.2.10 API_EXTCLK — API external clock output pin . . . 232
7.2.11 TEMPSENSE — Internal Temperature Sensor Output Voltage . . . 232
7.3 Memory Map and Registers . . . 233
7.3.1 Module Memory Map . . . 233
7.3.2 Register Descriptions . . . 235
7.4 Functional Description . . . 273
7.4.1 Phase Locked Loop with Internal Filter (PLL) . . . 273
7.4.2 Startup from Reset . . . 275
7.4.3 Stop Mode using PLLCLK as source of the Bus Clock . . . 275
7.4.4 Full Stop Mode using Oscillator Clock as source of the Bus Clock . . . 276
7.4.5 External Oscillator . . . 277
7.4.6 System Clock Configurations . . . 278
7.5 Resets . . . 279
7.5.1 General . . . 279
7.5.2 Description of Reset Operation . . . 280
7.5.3 Oscillator Clock Monitor Reset . . . 280
7.5.4 PLL Clock Monitor Reset . . . 280
7.5.5 Computer Operating Properly Watchdog (COP) Reset . . . 281
7.5.6 Power-On Reset (POR) . . . 282
7.5.7 Low-Voltage Reset (LVR) . . . 282
7.6 Interrupts . . . 283
7.6.1 Description of Interrupt Operation . . . 283
7.7 Initialization/Application Information . . . 285
7.7.1 General Initialization Information . . . 285
7.7.2 Application information for COP and API usage . . . 285
7.7.3 Application Information for PLL and Oscillator Startup . . . 285

## Chapter 8
## Timer Module (TIM16B8CV3) Block Description


8.1 Introduction . . . 287
8.1.1 Features . . . 287
8.1.2 Modes of Operation . . . 288
8.1.3 Block Diagrams . . . 288
8.2 External Signal Description . . . 291
8.2.1 IOC7 — Input Capture and Output Compare Channel 7 . . . 291
8.2.2 IOC6 - IOC0 — Input Capture and Output Compare Channel 6-0 . . . 291
8.3 Memory Map and Register Definition . . . 291
8.3.1 Module Memory Map . . . 291
8.3.2 Register Descriptions . . . 292
8.4 Functional Description . . . 308
8.4.1 Prescaler . . . 310
8.4.2 Input Capture . . . 310
8.4.3 Output Compare . . . 310
8.4.4 Pulse Accumulator . . . 311
8.4.5 Event Counter Mode . . . 312
8.4.6 Gated Time Accumulation Mode . . . 312
8.5 Resets . . . 312
8.6 Interrupts . . . 312
8.6.1 Channel [7:0] Interrupt (C[7:0]F) . . . 313
8.6.2 Pulse Accumulator Input Interrupt (PAOVI) . . . 313
8.6.3 Pulse Accumulator Overflow Interrupt (PAOVF) . . . 313
8.6.4 Timer Overflow Interrupt (TOF) . . . 313


## Chapter 9
## Pulse-Width Modulator (S12PWM8B8CV2)


9.1 Introduction . . . 315
9.1.1 Features . . . 315
9.1.2 Modes of Operation . . . 315
9.1.3 Block Diagram . . . 316
9.2 External Signal Description . . . 316
9.2.1 PWM7 - PWM0 — PWM Channel 7 - 0 . . . 317
9.3 Memory Map and Register Definition . . . 317
9.3.1 Module Memory Map . . . 317
9.3.2 Register Descriptions . . . 317
9.4 Functional Description . . . 332
9.4.1 PWM Clock Select . . . 332
9.4.2 PWM Channel Timers . . . 335
9.5 Resets . . . 343
9.6 Interrupts . . . 344

## Chapter 10
## Analog-to-Digital Converter (ADC12B_LBA_V1)


10.1 Introduction . . . 347
10.2 Key Features . . . 349
10.2.1 Modes of Operation . . . 350
10.2.2 Block Diagram . . . 353
10.3 Signal Description . . . 354
10.3.1 Detailed Signal Descriptions . . . 354
10.4 Memory Map and Register Definition . . . 355
10.4.1 Module Memory Map . . . 355
10.4.2 Register Descriptions . . . 358
10.5 Functional Description . . . 390
10.5.1 Overview . . . 390
10.5.2 Analog Sub-Block . . . 390
10.5.3 Digital Sub-Block . . . 391
10.6 Resets . . . 404
10.7 Interrupts . . . 404
10.7.1 ADC Conversion Interrupt . . . 404
10.7.2 ADC Sequence Abort Done Interrupt . . . 404
10.7.3 ADC Error and Conversion Flow Control Issue Interrupt . . . 405
10.8 Use Cases and Application Information . . . 406
10.8.1 List Usage - CSL single buffer mode and RVL single buffer mode . . . 406
10.8.2 List Usage - CSL single buffer mode and RVL double buffer mode . . . 406
10.8.3 List Usage - CSL double buffer mode and RVL double buffer mode . . . 407
10.8.4 List Usage - CSL double buffer mode and RVL single buffer mode . . . 407
10.8.5 List Usage - CSL double buffer mode and RVL double buffer mode . . . 408
10.8.6 RVL swapping in RVL double buffer mode and related registers ADCIMDRI and ADCEOLRI 408
10.8.7 Conversion flow control application information . . . 410
10.8.8 Continuous Conversion . . . 412
10.8.9 Triggered Conversion - Single CSL . . . 413
10.8.10 Fully Timing Controlled Conversion . . . 414


## Chapter 11
## Freescale’s Scalable Controller Area Network (S12MSCANV3)


11.1 Introduction . . . 415
11.1.1 Glossary . . . 416
11.1.2 Block Diagram . . . 416
11.1.3 Features . . . 417
11.1.4 Modes of Operation . . . 417
11.2 External Signal Description . . . 418
11.2.1 RXCAN — CAN Receiver Input Pin . . . 418
11.2.2 TXCAN — CAN Transmitter Output Pin . . . 418
11.2.3 CAN System . . . 418

11.3 Memory Map and Register Definition . . . . . 419
11.3.1 Module Memory Map . . . . . 419
11.3.2 Register Descriptions . . . . . 421
11.3.3 Programmer's Model of Message Storage . . . . . 440
11.4 Functional Description . . . . . 451
11.4.1 General . . . . . 451
11.4.2 Message Storage . . . . . 451
11.4.3 Identifier Acceptance Filter . . . . . 454
11.4.4 Modes of Operation . . . . . 460
11.4.5 Low-Power Options . . . . . 462
11.4.6 Reset Initialization . . . . . 466
11.4.7 Interrupts . . . . . 466
11.5 Initialization/Application Information . . . . . 468
11.5.1 MSCAN initialization . . . . . 468
11.5.2 Bus-Off Recovery . . . . . 468

## Chapter 12
## Serial Communication Interface (S12SCIV6)

12.1 Introduction . . . . . 469
12.1.1 Glossary . . . . . 469
12.1.2 Features . . . . . 470
12.1.3 Modes of Operation . . . . . 471
12.1.4 Block Diagram . . . . . 471
12.2 External Signal Description . . . . . 472
12.2.1 TXD — Transmit Pin . . . . . 472
12.2.2 RXD — Receive Pin . . . . . 472
12.3 Memory Map and Register Definition . . . . . 472
12.3.1 Module Memory Map and Register Definition . . . . . 472
12.3.2 Register Descriptions . . . . . 473
12.4 Functional Description . . . . . 485
12.4.1 Infrared Interface Submodule . . . . . 486
12.4.2 LIN Support . . . . . 486
12.4.3 Data Format . . . . . 487
12.4.4 Baud Rate Generation . . . . . 488
12.4.5 Transmitter . . . . . 489
12.4.6 Receiver . . . . . 494
12.4.7 Single-Wire Operation . . . . . 502
12.4.8 Loop Operation . . . . . 503
12.5 Initialization/Application Information . . . . . 503
12.5.1 Reset Initialization . . . . . 503
12.5.2 Modes of Operation . . . . . 504
12.5.3 Interrupt Operation . . . . . 504
12.5.4 Recovery from Wait Mode . . . . . 507
12.5.5 Recovery from Stop Mode . . . . . 507

## Chapter 13
## Serial Peripheral Interface (S12SPIV5)


13.1 Introduction . . . 509
13.1.1 Glossary of Terms . . . 509
13.1.2 Features . . . 509
13.1.3 Modes of Operation . . . 509
13.1.4 Block Diagram . . . 510
13.2 External Signal Description . . . 511
13.2.1 MOSI — Master Out/Slave In Pin . . . 511
13.2.2 MISO — Master In/Slave Out Pin . . . 511
13.2.3 SS — Slave Select Pin . . . 512
13.2.4 SCK — Serial Clock Pin . . . 512
13.3 Memory Map and Register Definition . . . 512
13.3.1 Module Memory Map . . . 512
13.3.2 Register Descriptions . . . 513
13.4 Functional Description . . . 521
13.4.1 Master Mode . . . 522
13.4.2 Slave Mode . . . 523
13.4.3 Transmission Formats . . . 524
13.4.4 SPI Baud Rate Generation . . . 529
13.4.5 Special Features . . . 530
13.4.6 Error Conditions . . . 531
13.4.7 Low Power Mode Options . . . 532


## Chapter 14
## Inter-Integrated Circuit (IICV3) Block Description


14.1 Introduction . . . 535
14.1.1 Features . . . 535
14.1.2 Modes of Operation . . . 537
14.1.3 Block Diagram . . . 537
14.2 External Signal Description . . . 538
14.2.1 IIC_SCL — Serial Clock Line Pin . . . 538
14.2.2 IIC_SDA — Serial Data Line Pin . . . 538
14.3 Memory Map and Register Definition . . . 538
14.3.1 Register Descriptions . . . 538
14.4 Functional Description . . . 550
14.4.1 I-Bus Protocol . . . 550
14.4.2 Operation in Run Mode . . . 555
14.4.3 Operation in Wait Mode . . . 555
14.4.4 Operation in Stop Mode . . . 555
14.5 Resets . . . 555
14.6 Interrupts . . . 555
14.7 Application Information . . . 556
14.7.1 IIC Programming Examples . . . 556

## Chapter 15
## Liquid Crystal Display (LCD40F4BV3) Block Description


15.1 Introduction . . . 563
15.1.1 Features . . . 564
15.1.2 Modes of Operation . . . 564
15.1.3 Block Diagram . . . 564
15.2 External Signal Description . . . 566
15.2.1 BP[3:0] — Analog Backplane Pins . . . 566
15.2.2 FP[39:0] — Analog Frontplane Pins . . . 566
15.2.3 VLCD — LCD Supply Voltage Pin . . . 566
15.3 Memory Map and Register Definition . . . 566
15.3.1 Module Memory Map . . . 566
15.3.2 Register Descriptions . . . 568
15.4 Functional Description . . . 575
15.4.1 LCD Driver Description . . . 575
15.4.2 Operation in Wait Mode . . . 577
15.4.3 Operation in Stop Mode . . . 577
15.4.4 LCD Waveform Examples . . . 578
15.4.5 LCD Clock Inputs & Reset Behavior . . . 584
15.5 Resets . . . 585
15.6 Interrupts . . . 585


## Chapter 16
## Motor Controller (MC10B8CV1)


16.1 Introduction . . . 587
16.1.1 Features . . . 587
16.1.2 Modes of Operation . . . 587
16.1.3 Block Diagram . . . 589
16.2 External Signal Description . . . 590
16.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM Output Pins for Motor 0 . . . 590
16.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM Output Pins for Motor 1 . . . 590
16.3 Memory Map and Register Definition . . . 590
16.3.1 Module Memory Map . . . 590
16.3.2 Register Descriptions . . . 592
16.4 Functional Description . . . 599
16.4.1 Modes of Operation . . . 599
16.4.2 PWM Duty Cycle . . . 611
16.4.3 Motor Controller Counter Clock Source . . . 611
16.4.4 Output Switching Delay . . . 612
16.4.5 Operation in Wait Mode . . . 613
16.4.6 Operation in Stop and Pseudo-Stop Modes . . . 613
16.5 Reset . . . 613
16.6 Interrupts . . . 613
16.6.1 Timer Counter Overflow Interrupt . . . 613

16.7 Initialization/Application Information . . . . . 614
16.7.1 Code Example . . . . . 614

## Chapter 17
## Stepper Stall Detector (SSDV2) Block Description

17.1 Introduction . . . . . 615
17.1.1 Modes of Operation . . . . . 615
17.1.2 Features . . . . . 615
17.1.3 Block Diagram . . . . . 616
17.2 External Signal Description . . . . . 617
17.2.1 COSM/COSP — Cosine Coil Pins for Motor . . . . . 617
17.2.2 SINM/SINP — Sine Coil Pins for Motor . . . . . 617
17.3 Memory Map and Register Definition . . . . . 618
17.3.1 Module Memory Map . . . . . 618
17.3.2 Register Descriptions . . . . . 618
17.4 Functional Description . . . . . 625
17.4.1 Return to Zero Modes . . . . . 626
17.4.2 Full Step States . . . . . 626
17.4.3 Operation in Low Power Modes . . . . . 630
17.4.4 Stall Detection Flow . . . . . 630

## Chapter 18
## Real-Time Counter With Calendar (RTCV2)

18.1 Introduction . . . . . 633
18.2 Features . . . . . 633
18.2.1 Modes of Operation . . . . . 633
18.2.2 Block Diagram . . . . . 634
18.3 External Signal Description . . . . . 635
18.3.1 OSCCLK . . . . . 635
18.3.2 OSCCLK_32K . . . . . 636
18.3.3 IRCCLK . . . . . 636
18.3.4 RTCCLK . . . . . 636
18.3.5 CALCLK . . . . . 636
18.4 Register Definition . . . . . 636
18.4.1 RTC Control Register 1(RTCCTL1) . . . . . 637
18.4.2 RTC Control Register 2 (RTCCTL2) . . . . . 638
18.4.3 RTC Control Register 3 (RTCCTL3) . . . . . 638
18.4.4 RTC Control Register 4 (RTCCTL4) . . . . . 639
18.4.5 RTC Status Register 1 (RTCS1) . . . . . 640
18.4.6 RTC Compensation Configure Register (RTCCCR) . . . . . 641
18.4.7 RTC Counter Register (RTCCNT) . . . . . 642
18.4.8 RTC Modulo Register (RTCMOD) . . . . . 643
18.4.9 RTC Second Register (RTCSECR) . . . . . 643
18.4.10RTC Minute Register (RTCMINR) . . . . . 644

18.4.11RTC Hour Register (RTCHRR) . . . . 644
18.5 Functional Description . . . . 645
18.5.1 RTC clock and reset behavior . . . . 645
18.5.2 Calendar Functions . . . . 645
18.5.3 Interrupts . . . . 645
18.5.4 RTC Clock Compensation . . . . 646
18.5.5 Calendar Register and Bit Write Protection . . . . 647
18.5.6 Load buffer register . . . . 648
18.6 Initialization/Application Information . . . . 648
18.6.1 RTC Calibration . . . . 648
18.6.2 RTC compensation . . . . 648

## Chapter 19
## Simple Sound Generator (SSGV1)

19.1 Introduction . . . . 651
19.1.1 Features . . . . 651
19.1.2 Block Diagram . . . . 651
19.2 External Signal Description . . . . 652
19.2.1 SGT . . . . 652
19.2.2 SGA . . . . 653
19.3 Memory Map and Register Definition . . . . 653
19.3.1 Module Memory Map . . . . 653
19.3.2 Register Descriptions . . . . 653
19.4 Functional Description . . . . 664
19.4.1 SSG Amplitude Generation . . . . 664
19.4.2 SSG Tone Generation . . . . 665
19.4.3 SSG Attack and Decay function . . . . 665
19.4.4 SSG Start and Stop . . . . 668
19.4.5 Register Reload . . . . 669
19.4.6 SSG Output Control . . . . 670
19.5 Interrupt . . . . 670

## Chapter 20
## ECC Generation module (SRAM_ECCV1)

20.1 Introduction . . . . 671
20.1.1 Features . . . . 671
20.2 Memory Map and Register Definition . . . . 671
20.2.1 Register Summary . . . . 671
20.2.2 Register Descriptions . . . . 673
20.3 Functional Description . . . . 677
20.3.1 Aligned 2 and 4 Byte Memory write access . . . . 678
20.3.2 Other Memory write access . . . . 678
20.3.3 Memory read access . . . . 679
20.3.4 Memory initialization . . . . 679

20.3.5 Interrupt handling . . . 679
20.3.6 ECC Algorithm . . . 680
20.3.7 ECC Debug Behavior . . . 680

## Chapter 21
## 64 KB Flash Module (S12ZFTMRZ64K2KV2)

21.1 Introduction . . . 684
21.1.1 Glossary . . . 684
21.1.2 Features . . . 685
21.1.3 Block Diagram . . . 685
21.2 External Signal Description . . . 686
21.3 Memory Map and Registers . . . 687
21.3.1 Module Memory Map . . . 687
21.3.2 Register Descriptions . . . 691
21.4 Functional Description . . . 711
21.4.1 Modes of Operation . . . 711
21.4.2 IFR Version ID Word . . . 711
21.4.3 Flash Block Read Access . . . 711
21.4.4 Internal NVM resource . . . 712
21.4.5 Flash Command Operations . . . 712
21.4.6 Allowed Simultaneous P-Flash and EEPROM Operations . . . 717
21.4.7 Flash Command Description . . . 718
21.4.8 Interrupts . . . 734
21.4.9 Wait Mode . . . 734
21.4.10Stop Mode . . . 735
21.5 Security . . . 735
21.5.1 Unsecuring the MCU using Backdoor Key Access . . . 735
21.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM . . . 736
21.5.3 Mode and Security Effects on Flash Command Availability . . . 736
21.6 Initialization . . . 737

## Chapter 22
## Supply Voltage Sensor - (BATSV2)

22.1 Introduction . . . 739
22.1.1 Features . . . 739
22.1.2 Modes of Operation . . . 739
22.1.3 Block Diagram . . . 740
22.2 External Signal Description . . . 740
22.2.1 VSENSE — Supply (Battery) Voltage Sense Pin . . . 740
22.2.2 VSUP — Voltage Supply Pin . . . 741
22.3 Memory Map and Register Definition . . . 741
22.3.1 Register Summary . . . 741
22.3.2 Register Descriptions . . . 742
22.4 Functional Description . . . 747

22.4.1 General . . . 747
22.4.2 Interrupts . . . 748

## Appendix AMCU Electrical Specifications

A.1 General . . . 753
A.1.1 Parameter Classification . . . 753
A.1.2 Pins . . . 754
A.1.3 Current Injection . . . 755
A.1.4 Absolute Maximum Ratings . . . 756
A.1.5 ESD Protection and Latch-up Immunity . . . 756
A.1.6 Operating Conditions . . . 757
A.1.7 Power Dissipation and Thermal Characteristics . . . 758
A.1.8 I/O Characteristics . . . 761
A.1.9 Supply Currents . . . 763
A.1.10 ADC Conversion Result Reference . . . 765

## Appendix BADC Electricals

B.1 ADC Operating Characteristics . . . 767
B.1.1 Factors Influencing Accuracy . . . 767
B.1.2 ADC Accuracy . . . 769

## Appendix CPLL Electrical Specifications

C.1 Reset, Oscillator and PLL . . . 773
C.1.1 Phase Locked Loop . . . 773

## Appendix DIRC Electrical Specifications

## Appendix ELCD Electrical Specifications

E.1 LCD Driver . . . 777

## Appendix FMSCAN Electrical Specifications

F.1 MSCAN Electrical Characteristics . . . 779

## Appendix G
## NVM Electrical Parameters

G.1 NVM Timing Parameters . . . 781
G.2 NVM Reliability Parameters . . . 782

## Appendix H
## BATS Electrical Specifications

H.1 Maximum Ratings . . . 785
H.2 Static Electrical Characteristics . . . 786
H.3 Dynamic Electrical Characteristics . . . 787

## Appendix IVREG Electrical Specifications

## Appendix JElectrical Characteristics for the Oscillator (OSCLCPcr)

J.1 OSCLCP Electrical characteristics . . . . . 791

## Appendix KOSC32K Electrical Specifications

K.1 DC Electrical Specifications . . . . . 793
K.2 Frequency Specifications . . . . . 793

## Appendix LSPI Electrical Specifications

L.1 Master Mode . . . . . 795
L.2 Slave Mode . . . . . 797

## Appendix MOrdering Information

## Appendix NPackage Information

N.1 144 LQFP . . . . . 804
N.2 100 LQFP . . . . . 807

## Appendix ODetailed Register Address Map

O.1 Detailed Register Map. . . . . 811

Chapter 1 Device Overview MC9S12ZVHY-Family . . . . . .21
Chapter 2 Port Integration Module (S12ZVHYPIMV1) . . . . . .67
Chapter 3 Memory Mapping Control (S12ZMMCV1) . . . . . .107
Chapter 4 Interrupt (S12ZINTV0) . . . . . .119
Chapter 5 Background Debug Controller (S12ZBDCV2) . . . . . .133
Chapter 6 S12Z Debug (S12ZDBGV2) Module . . . . . .171
Chapter 7 S12 Clock, Reset and Power Management Unit (S12CPMU_UHV_V5)221
Chapter 8 Timer Module (TIM16B8CV3) Block Description . . . . . .287
Chapter 9 Pulse-Width Modulator (S12PWM8B8CV2) . . . . . .315
Chapter 10 Analog-to-Digital Converter (ADC12B_LBA_V1) . . . . . .345
Chapter 11 Freescale’s Scalable Controller Area Network (S12MSCANV3). 415
Chapter 12 Serial Communication Interface (S12SCIV6) . . . . . .469
Chapter 13 Serial Peripheral Interface (S12SPIV5) . . . . . .509
Chapter 14 Inter-Integrated Circuit (IICV3) Block Description . . . . . .535
Chapter 15 Liquid Crystal Display (LCD40F4BV3) Block Description . .563
Chapter 16 Motor Controller (MC10B8CV1) . . . . . .587
Chapter 17 Stepper Stall Detector (SSDV2) Block Description . . . . . .615
Chapter 18 Real-Time Counter With Calendar (RTCV2) . . . . . .633
Chapter 19 Simple Sound Generator (SSGV1) . . . . . .651
Chapter 20 ECC Generation module (SRAM_ECCV1) . . . . . .671
Chapter 21 64 KB Flash Module (S12ZFTMRZ64K2KV2) . . . . . .683
Chapter 22 Supply Voltage Sensor - (BATSV2) . . . . . .739
Appendix A MCU Electrical Specifications . . . . . .753
Appendix B ADC Electricals . . . . . .767
Appendix C PLL Electrical Specifications . . . . . .773
Appendix D IRC Electrical Specifications . . . . . .775

Appendix E LCD Electrical Specifications . . . . . .777
Appendix F MSCAN Electrical Specifications . . . . . .779
Appendix G NVM Electrical Parameters . . . . . .781
Appendix H BATS Electrical Specifications . . . . . .785
Appendix I VREG Electrical Specifications . . . . . .789
Appendix J Electrical Characteristics for the Oscillator (OSCLCPcr) . . .791
Appendix K OSC32K Electrical Specifications . . . . . .793
Appendix L SPI Electrical Specifications . . . . . .795
Appendix M Ordering Information . . . . . .801
Appendix N Package Information . . . . . .803
Appendix O Detailed Register Address Map . . . . . .811

# Chapter 1
# Device Overview MC9S12ZVHY-Family

**Table 1-1. Revision History**

| Version Number | Revision Date | Description of Changes |
|---|---|---|
| 0.0 | Feb 2012 | • initial Draft |
| 0.01 | Mar 2012 | • fix typos |
| 0.02 | April 2012 | • Add the BDCFCLK source to bus clock<br>• Add RTC clock source |
| 0.04 | July 2012 | • Fix typos |
| 0.05 | Sep 2012 | • Update ADC conversion reference IFR location<br>• Correct 100LQFP pinout signals typos<br>• Fix base on review feedback<br>• Add FTMRZ related connection |
| 0.06 | Nov 2012 | • Update for 1N39G |
| 0.07 | July 2013 | • Add 32K device |

## 1.1 Introduction

The MC9S12ZVHY-Family is an optimized automotive 16-bit microcontroller product line focused on low-cost, high-performance and application component count reduction. This family integrates many of components of the S12ZVH-family MagniV mixed signal microcontroller, including a 5V regulator system to supply the microcontroller and other components. The MC9S12ZVHY-Family is targeted at automotive & motorcycle instrument cluster applications requiring stepper motor gauges, and segment LCD displays.

The MC9S12ZVHY-Family features a 4x40 liquid crystal display (LCD) controller/driver and a pulse width modulated motor controller (MC) consisting of up to 16 high current outputs. The device is capable of stepper motor stall detection (SSD) via hardware or software, please contact Freescale sales office for detailed information on software SSD.

The MC9S12ZVHY-Family delivers an optimized solution with the integration of several key system components into a single device, optimizing system architecture and achieving significant PCB space savings. The MC9S12ZVHY-Family delivers all the advantages and efficiencies of a 16-bit MCU while retaining the low cost, power consumption, EMC, and code-size efficiency advantages currently enjoyed by users of Freescale's existing 16-bit MCU families. The MC9S12ZVHY-Family also features the revolutionary S12Z CPU with code size & execution efficiencies even higher than our class leading S12X CPU. It also provides a linear memory map for flash memory for all members of the family, eliminating the inconvenience and performance impact of page swapping. In addition to the I/O ports available in each module, further I/O ports are available with interrupt capability allowing wake-up from stop or wait modes.

## 1.2 Features

This section describes the key features of the MC9S12ZVHY-Family.

### 1.2.1 MC9S12ZVHY-Family Member Comparison

Table 1-2 provides a summary of feature set differences within the MC9S12ZVHY-Family. All other features are common to all MC9S12ZVHY-Family members.

**Table 1-2. MC9S12ZVHY-Family Comparison**

| Feature | MC9S12ZVHY64 | | MC9S12ZVHY32 | |
|---|---|---|---|---|
| Package | 100 pins (LQFP) | 144 pins (LQFP) | 100 pins (LQFP) | 144 pins (LQFP) |
| CPU | HCS12Z | | HCS12Z | |
| Flash memory (ECC) | 64 KB | | 32 KB | |
| EEPROM (ECC) | 2 KB | | 2KB | |
| RAM (ECC) | 4 KB | | 2 KB | |
| Stepper Motor Drive (with HW SSD) | 2 | | 2 | |
| Segment LCD | 4 x 32 | 4 x 40 | 4 x 32 | 4 x 40 |
| Simple Sound Generator (SSG) | Yes | | Yes | |
| SCI | 2 | | 2 | |
| SPI | 1 | | 1 | |
| IIC | 1 | | 1 | |
| CAN (digital communication module) | 1 | | 1 | |
| Timer | Two 8ch x 16-bit (not all IOC avaiable on pins) | Two 8ch x 16-bit | Two 8ch x 16-bit (not all IOC avaiable on pins) | Two 8ch x 16-bit |
| PWM | 8 ch (8-bit) / 4ch (16-bit) | | 8 ch (8-bit) / 4ch (16-bit) | |
| RTC | Yes | | Yes | |
| ADC Resolution | 10-bit resolution | | 10-bit resolution | |
| ADC Inputs | 4 pins + internal signals | 8 pins + internal signals | 4 pins + internal signals | 8 pins + internal signals |
| Frequency modulated PLL | Yes | | Yes | |
| Internal 1 MHz RC oscillator | Yes | | Yes | |

| Feature | MC9S12ZVHY64 | | MC9S12ZVHY32 | |
|---|---|---|---|---|
| Autonomous window watchdog | 1 (with independent clock source) | | 1 (with independent clock source) | |
| Key Wakeup I/Os | 19 | 24 | 19 | 24 |
| General purpose I/Os (5 V)(1) | up to 73 | up to 100 | up to 73 | up to 100 |
| Direct Battery Voltage sense pin | Yes | | Yes | |
| Vsup sense | Yes | | Yes | |
| Chip temperature sensor | 1 General sensor | | 1 General sensor | |
| VSUP Supply voltage | 5.5 V – 18 V (normal operation)<br>up to 40V (protected operation) | | 5.5 V – 18 V (normal operation)<br>up to 40V (protected operation) | |
| VDDX Output current | Determined by power dissipation of external ballast | | Determined by power dissipation of external ballast | |
| Maximum Bus Frequency | 32 MHz | | 32 MHz | |

1. Maximum I/O count based on multiplexing with peripherals.

## 1.3 Maskset 0N39G and 1N39G device compare

0N39G and 1N39G devices have difference on some module versions. Table 1-3 shows the difference.

**NOTE User should take care when switch from 0N39G to 1N39G device**

**Table 1-3. Device Difference for 0N39G and 1N39G**

| | 0N39G | 1N39G |
|---|---|---|
| SCI | V5 | V6 |
| BDC | V1 | V2 |
| MCU | no ADC reference voltage to IFR | ADC reference voltage to IFR |

## 1.4 Chip-Level Features

On-chip modules available within the family include the following features:

- S12Z CPU core
- Up to 64 KB on-chip flash with ECC
- 2 KB EEPROM with ECC
- Up to 4 KB on-chip SRAM with ECC
- Phase locked loop (IPLL) frequency multiplier with internal filter
- 1 MHz internal RC oscillator with +/-1.3% accuracy over rated temperature range
- 4-20 MHz amplitude controlled pierce oscillator

- 32 KHz oscillator for RTC and LCD
- Internal COP (watchdog) module
- LCD driver for segment LCD with 40 frontplanes x 4 backplanes
- Stepper Motor Controller (MC) with drivers for up to 2 motors
- Up to 2 Stepper Stall Detector (SSD) modules (one for each motor)
- Real Time Clock (RTC) support the Hour/Minute/Second function and frequency compensation
- One Analog-to-Digital Converters (ADC) with 10-bit resolution and up to 8 channels available on external pins
- Two Timer module (TIM) supporting input/output channels that provide a range of 16-bit input capture & output compare (8 channels)
- One Pulse Width Modulation (PWM) modules with up to 8 x 8-bit channels
- Simple Sound Generation (SSG) for monotonic tone generation
- One Inter-Integrated Circuit (IIC) module
- One Serial Peripheral Interface (SPI) module
- Two Serial Communication Interface (SCI) module supporting LIN 1.3, 2.0, 2.1 and SAE J2602 communications
- One MSCAN (up to 1 Mbp/s, CAN 2.0 A, B compliant) module
- On-chip Voltage Regulator (VREG) for regulation of input supply and all internal voltages
- Autonomous Periodic Interrupt (API) (combination with cyclic, watchdog)
- Supply voltage sense with low battery warning.
- Chip temperature sensor

## 1.5 Module Features

The following sections provide more details of the integrated modules.

### 1.5.1 S12Z Central Processor Unit (CPU)

The S12Z CPU is a revolutionary high-speed core, with code size and execution efficiencies over the S12X CPU. The S12Z CPU also provides a linear memory map eliminating the inconvenience & performance impact of page swapping.

- Harvard Architecture - parallel data and code access
- 3 stage pipeline
- 32-Bit wide instruction and databus
- 32-Bit ALU
- 24-bit addressing, i.e. 16 MB linear address space
- Instructions and Addressing modes optimized for C-Programming & Compiler
- Optimized address path so it is capable to run at maximum bus cycle without Flash wait states
  - MAC unit 32bit += 32bit*32bit
  - Hardware divider

  - Single cycle multi-bit shifts (Barrel shifter)
- Special instructions for fixed point match
- Unimplemented opcode traps
- Unprogrammed byte value (0xFF) defaults to SWI instruction

#### 1.5.1.1 Background Debug Controller (BDC)

- Background Debug Controller (BDC) with single-wire interface
  - Non-intrusive memory access commands
  - Supports in-circuit programming of on-chip nonvolatile memory

#### 1.5.1.2 Debugger (DBG)

- Enhanced DBG module including:
  - Four comparators (A, B, C and D) each configurable to monitor PC addresses or addresses of data accesses
  - A and C compare full address bus and full 32-bit data bus with data bus mask register
  - B and D compare full address bus only
  - Three modes: simple address/data match, inside address range, or outside address range
  - Tag-type or force-type hardware breakpoint requests
- State sequencer control
- 64 x 64-bit circular trace buffer to capture change-of-flow addresses or address and data of every access
  - Begin, End and Mid alignment of tracing to trigger
- Profiling mode

### 1.5.2 Embedded Memory

#### 1.5.2.1 Flash

On-chip flash memory on the MC9S12ZVHY-Family:

- Up to 64 KB of program flash memory
  - 32 data bits plus 7 syndrome ECC (error correction code) bits allow single bit fault correction and double fault detection
  - Erase sector size 512 bytes
  - Automated program and erase algorithm
  - User margin level setting for reads
  - Protection scheme to prevent accidental program or erase

#### 1.5.2.2 EEPROM

- Up to 2 KB EEPROM

— 16 data bits plus 6 syndrome ECC (error correction code) bits allow single bit error correction and double fault detection
— Erase sector size 4 bytes
— Automated program and erase algorithm
— User margin level setting for reads

#### 1.5.2.3 SRAM

- Up to 4 KB of general-purpose RAM with ECC
  — Single bit error correction and double bit error detection

### 1.5.3 Clocks, Reset & Power Management Unit (CPMU)

- Real Time Interrupt (RTI)
- Clock Monitor, supervising the correct function of the oscillator (CM)
- System reset generation
- Autonomous periodic interrupt (API) (combination with cyclic, watchdog)
- Low Power Operation
  — RUN mode is the main full performance operating mode with the entire device clocked.
  — WAIT mode when the internal CPU clock is switched off, so the CPU does not execute instructions.
  — Pseudo STOP - system clocks are stopped but the RTI, COP, API, RTC and LCD modules can be enabled with clock source from the osc.
  — STOP - the oscillator is stopped in this mode, all clocks are switched off and all counters and dividers remain frozen. The 32K oscillator can be enabled, RTC and LCD can be still function if enabled. The API and COP can still function if their clock source are from API clock(ACLK).

#### 1.5.3.1 Internal Phase-Locked Loop (IPLL)

- Phase-locked-loop clock frequency multiplier
  — No external components required
  — Reference divider and multiplier allow large variety of clock rates
  — Automatic bandwidth control mode for low-jitter operation
  — Automatic frequency lock detector
  — Configurable option to spread spectrum for reduced EMC radiation (frequency modulation)
  — Reference clock sources:
    – Internal 1 MHz RC oscillator (IRC)
    – External 4-20 MHz crystal oscillator/resonator

#### 1.5.3.2 Internal RC Oscillator (IRC)

- Trimmable internal reference clock.
  - Frequency: 1 MHz; Trimmed accuracy over -40°C to 150°C junction temperature range: ±1.3%

### 1.5.4 Main External Oscillator (XOSCLCP)

- Loop control Pierce oscillator using 4 MHz to 20 MHz crystal
  - Current gain control on amplitude output
  - Signal with low harmonic distortion
  - Low power
  - Good noise immunity
  - Eliminates need for external current limiting resistor
  - Transconductance sized for optimum start-up margin for typical crystals
  - Oscillator pins shared with GPIO functionality

### 1.5.5 32K External Oscillator

- Low speed oscillator using 32 kHz to 40 kHz crystal
  - Low power
  - Good noise immunity
  - Oscillator pins shared with GPIO functionality

### 1.5.6 System Integrity Support

- Power-On Reset (POR)
- Illegal address detection
- Low-voltage detection and low voltage reset generation
- Clock monitor
- High temperature Interrupt
- Computer Operating Properly (COP) watchdog
  - Configurable as window COP for enhanced failure detection
  - Can be initialized out of reset using option bits located in flash memory
- Unimplemented opcode traps
- Unprogrammed byte value (0xFF) defaults to SWI instruction
- ECC support on embedded NVM and SRAM

### 1.5.7 Real Time Clock (RTC)

- Basic Clock functions with separate counters for Hour, Minutes and Seconds.

- Hardware Compensation to reduce the effects of frequency variation on the 1 Hz clock (to the counters) caused by temperature changes of crystal characteristics. Correction factor calculated by firmware. (Programmable correction factor).
- 16-bit CPU register programming interface with protection against run-away code.
- Option to output the buffered 32.768 kHz clock or the compensated 1 Hz clock for calibration.

### 1.5.8 Timer (TIM)

- Up to two timer modules for input capture or output compare
  — 8 x 16-bit channels per module
- 16-bit free-running counter with 8-bit precision prescaler
- 16-bit pulse accumulator

### 1.5.9 Pulse Width Modulation Module (PWM)

- 8 channels x 8-bit (4 channels x 16-bit)
- Programmable period and duty cycle per channel
- Center-aligned or edge-aligned outputs
- Programmable clock select logic with a wide range of frequencies

### 1.5.10 Simple Sound Generator (SSG)

- Programmable amplitude level with maximum 11 bit resolution from zero amplitude to max amplitude
- Sound STOP function to stop sound generation immediately
- Registers double-buffered synchronously reload at edge of tone to avoid distortion of output tone.
- Interrupt generated when SSG configure registers reload occurs
- Input clock prescaler with 11 bit resolution
- Module disable for power saving when SSG is not in use
- Separate or mixed frequency and amplitude outputs for flexibility in external hardware variation.
- Decay/attack function which can decrease/increase sound amplitude automatically without cpu interaction. The function includs linear, gong and exponential decay/attack profiles

### 1.5.11 Liquid Crystal Display driver (LCD)

- Up to 40 frontplanes and 4 backplanes or general-purpose input or output
- 5 modes of operation allow for different display sizes to meet application requirements
- Unused frontplane and backplane pins can be used as general-purpose I/O

### 1.5.12 Stepper Motor Controller (MC)

- PWM motor controller (MC) with up to 16 high current outputs

- Each PWM channel switchable between two drivers in an H-bridge configuration
- Left, right and center aligned outputs
- Support for sine and cosine drive
- Dithering
- Output slew rate control

### 1.5.13 Stepper Stall Detect (SSD)

- Up to two SSD
- Programmable Full Step State
- Programmable Integration polarity
- Blanking (recirculation) state
- 16-bit Integration Accumulator register
- 16-Bit Modulus Down Counter with interrupt

### 1.5.14 Multi-Scalable Controller Area Network (MSCAN)

- Implementation of the CAN protocol — Version 2.0A/B
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a “local priority” concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or either 8-bit filters
- Programmable wake-up functionality with integrated low-pass filter

### 1.5.15 Inter-IC Bus Module (IIC)

- Compatible with $I^2C$ bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- General Call Address detection
- Compliant to ten-bit address

### 1.5.16 Serial Communication Interface Module (SCI)

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- 16-bit baud rate selection
- Programmable character length
- Programmable polarity for transmitter and receiver
- Active edge receive wakeup
- Break detection/generation supporting LIN communications

### 1.5.17 Serial Peripheral Interface Module (SPI)

- Configurable 8- or 16-bit data size
- Full-duplex or single-wire bidirectional
- Double-buffered transmit and receive
- Master or slave mode
- MSB-first or LSB-first shifting
- Serial clock phase and polarity options

### 1.5.18 Analog-to-Digital Converter Module (ADC)

- One ADC
  - — 10-bit resolution
  - — Up to 8 external channels & 8 internal channels
  - — Left or right aligned result data
  - — Continuous conversion mode
- ADC directly writes results to RAM, preventing stall of further conversions
- Internal signals monitored with the ADC module
  - — Vrh, Vrl, Vrl+Vrh/2, Vsup or Vsense monitor, Vbg, TempSense.
- External pins can also be used as digital I/O

### 1.5.19 Supply Voltage Sensor (BATS)

- VSENSE & VSUP pin low or a high voltage interrupt
- VSENSE & VSUP pin can be routed via an internal divider to the internal ADC channel
- Generation of low or high voltage interrupts

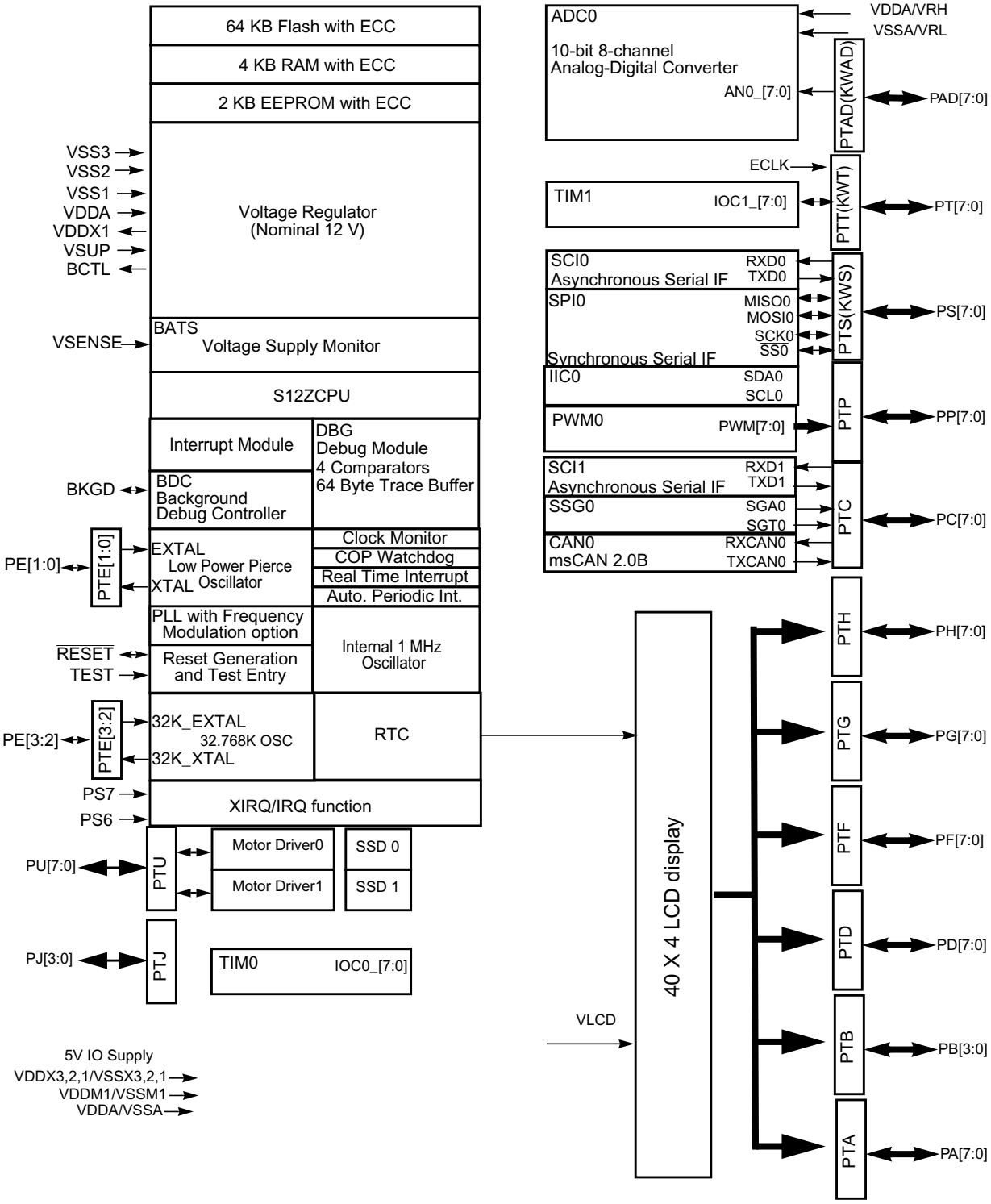
### 1.5.20 On-Chip Voltage Regulator system (VREG)

- Voltage regulator
  - — Linear voltage regulator directly supplied by $V_{SUP}$ (protected $V_{BAT}$)

— Low-voltage detect with low-voltage interrupt $V_{SUP}$
— Power-On Reset (POR)
— Low-Voltage Reset (LVR)
— External ballast device support to reduce internal power dissipation
— Capable of supplying both the MCU internally plus external components
— Over-temperature protection and interrupt

# 1.6 Block Diagram

**Figure 1-1. MC9S12ZVHY-Family Block Diagram**



### 1.6.1 Device Memory Map

Table 1-4 shows the device register memory map.

**Table 1-4. Module Register Address Ranges**

| Address | Module | Size (Bytes) |
|---|---|---|
| 0x0000-0x0003 | ID Registers | 4 |
| 0x0004-0x000F | Reserved | 12 |
| 0x0010-0x001F | INT | 16 |
| 0x0020-0x006F | Reserved | 80 |
| 0x0070-0x00FF | MMC | 144 |
| 0x0100-0x017F | DBG | 128 |
| 0x0180-0x01FF | Reserved | 128 |
| 0x0200-0x037F | PIM | 384 |
| 0x0380-0x039F | FTMRZ | 32 |
| 0x03A0-0x03BF | Reserved | 32 |
| 0x03C0-0x03CF | RAM ECC | 16 |
| 0x03D0-0x03FF | Reserved | 48 |
| 0x0400-0x042F | TIM1 | 48 |
| 0x0430-0x047F | Reserved | 80 |
| 0x0480-0x04AF | PWM | 48 |
| 0x04B0-0x05BF | Reserved | 272 |
| 0x05C0-0x05EF | TIM0 | 48 |
| 0x05F0-0x05FF | Reserved | 16 |
| 0x0600-0x063F | ADC0 | 64 |
| 0x0640-0x06BF | Reserved | 128 |
| 0x06C0-0x06DF | CPMU | 32 |
| 0x06E0-0x06EF | Reserved | 16 |
| 0x06F0-0x06F7 | BATS | 8 |
| 0x06F8-0x06FF | Reserved | 8 |
| 0x0700-0x0707 | SCI0 | 8 |
| 0x0708-0x070F | Reserved | 8 |
| 0x0710-0x0717 | SCI1 | 8 |
| 0x0718-0x077F | Reserved | 104 |
| 0x0780-0x0787 | SPI0 | 8 |

**Table 1-4. Module Register Address Ranges**

| Address | Module | Size (Bytes) |
|---|---|---|
| 0x0788-0x07BF | Reserved | 56 |
| 0x07C0-0x07C7 | IIC0 | 8 |
| 0x07C8-0x07FF | Reserved | 56 |
| 0x0800-0x083F | CAN0 | 64 |
| 0x0840-0x09FF | Reserved | 448 |
| 0x0A00-0x0A1F | LCD | 32 |
| 0x0A20-0x0A3F | Reserved | 32 |
| 0x0A40-0x0A7F | MC | 64 |
| 0x0A80-0x0A87 | SSD0 | 8 |
| 0x0A88-0x0A8F | Reserved | 8 |
| 0x0A90-0x0A97 | SSD1 | 8 |
| 0x0A98-0x0ADF | Reserved | 72 |
| 0x0AE0-0x0AEF | RTC | 16 |
| 0x0AF0-0x0AFF | Reserved | 16 |
| 0x0B00-0x0B17 | SSG0 | 24 |
| 0x0B18-0x0FFF | Reserved | 1256 |

**NOTE**

Reserved register space shown in the table is not allocated to any module. This register space is reserved for future use. Writing to these locations has no effect. Read access to these locations returns zero.

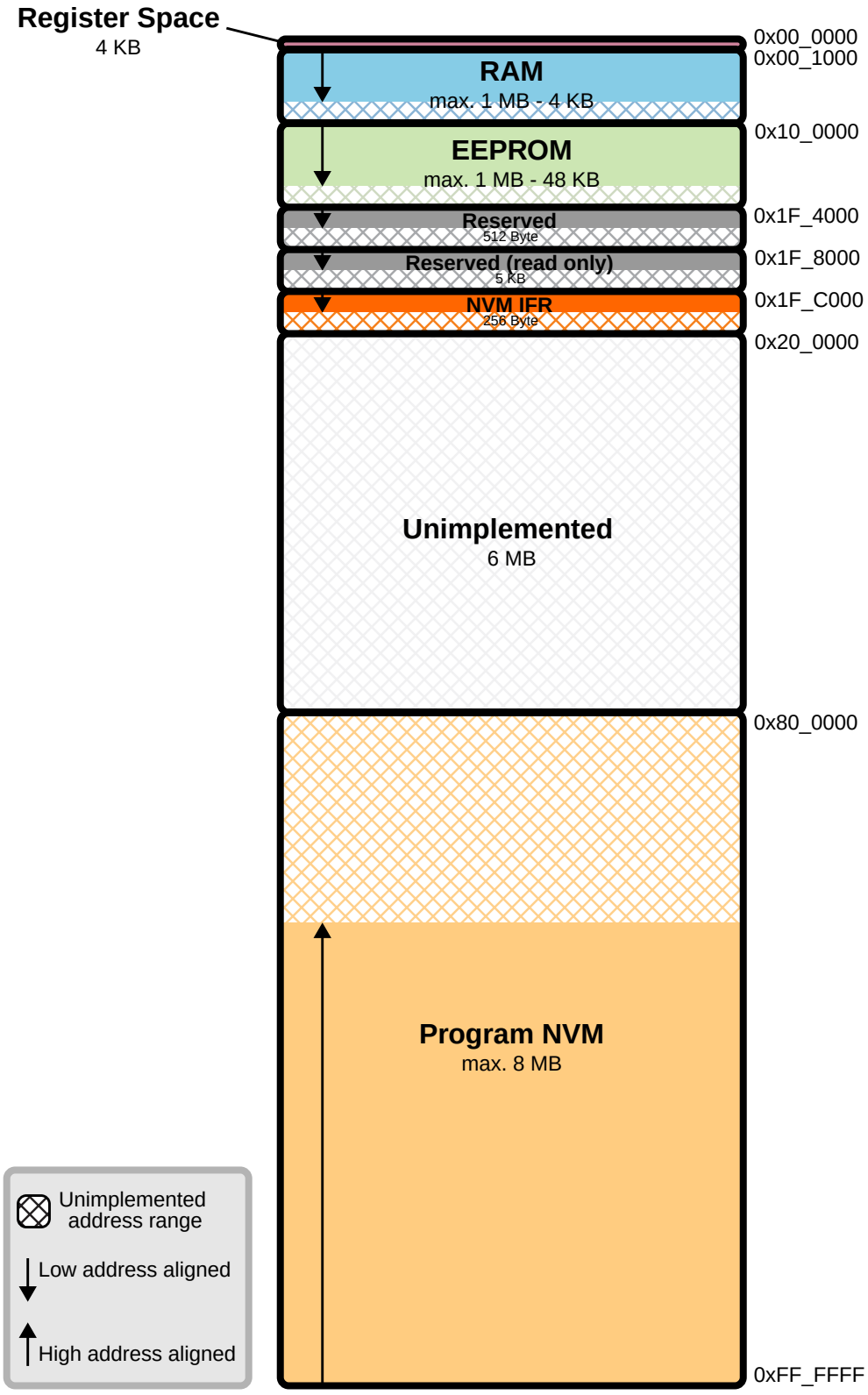



**Figure 1-2. MC9S12ZVHY-Family Global Memory Map.**

### 1.6.2 Part ID registers Assignments

The Part ID registers is located in four 8-bit registers at addresses 0x0000-0x0003. The read-only value is a unique ID for each revision of the chip. Table 1-5 shows the assigned Part ID register value.

**Table 1-5. Assigned IDs Numbers**

| Device | Mask Set number | Part ID |
|---|---|---|
| MC9S12ZVHY64 | 0N39G | 32'h03160000 |
| MC9S12ZVHY32 | 0N39G | 32'h03160000 |
| MC9S12ZVHY64 | 1N39G | 32'h03161000 |
| MC9S12ZVHY32 | 1N39G | 32'h03161000 |

## 1.7 Signal Description and Device Pinouts

This section describes signals that connect off-chip. It includes a pinout diagram, a table of signal properties, and detailed discussion of signals. It is built from the signal description sections of the individual IP blocks on the device.

### 1.7.1 Pin Assignment Overview

Table 1-6 provides a summary of which ports are available for 100-pin and 144-pin package option.

**Table 1-6. Port Availability by Package Option**

| Port | 144 LQFP | 100 LQFP |
|---|---|---|
| Port AD | PAD[7:0] | PAD[3:0] |
| Port A | PA[7:0] | PA[7:2] |
| Port B | PB[3:0] | PB[3:0] |
| Port C | PC[7:0] | PC[5:4] |
| Port D | PD[7:0] | PD[7:3] |
| Port E | PE[3:0] | PE[3:0] |
| Port F | PF[7:0] | PF[7:0] |
| Port G | PG[7:0] | PG[7:0] |
| Port H | PH[7:0] | PH[4:0] |
| Port P | PP[7:0] | PP[1,3,5,7] |
| Port S | PS[7:0] | PS[7:0] |
| Port T | PT[7:0] | PT[7:6], PT[4:0] |
| Port U | PU[7:0] | PU[7:0] |

**Table 1-6. Port Availability by Package Option**

| Port | 144 LQFP | 100 LQFP |
|---|---|---|
| Port J | PJ[3:0] | — |
| sum of ports | 100 | 73 |

**NOTE**

To avoid current drawn from floating inputs, all non-bonded pins should be configured as output or configured as input with a pull up or pull down device enabled

## 1.7.2 Detailed Signal Descriptions

### 1.7.2.1 $\overline{\text{RESET}}$ — External Reset Signal

The $\overline{\text{RESET}}$ signal is an active low bidirectional control signal. It acts as an input to initialize the MCU to a known start-up state, and an output when an internal MCU function causes a reset. The $\overline{\text{RESET}}$ pin has an internal pull-up device.

### 1.7.2.2 TEST — Test Pin

This input only pin is reserved for factory test. This pin has an internal pull-down device.

**NOTE**

The TEST pin must be tied to ground in all applications.

### 1.7.2.3 MODC — Mode C Signal

The MODC signal is used as a MCU operating mode select during reset. The state of this signal is latched to the MODC bit at the rising edge of $\overline{\text{RESET}}$. The signal has an internal pull-up device.

### 1.7.2.4 PAD[7:0] / KWAD[7:0] — Port AD, Input Pins of ADC

PAD[7:0] are general-purpose input or output signals. The signals can be configured on per signal basis as interrupt inputs with wake-up capability (KWAD[7:0]). These signals can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull devices are disabled.

### 1.7.2.5 PA[7:0] — Port A I/O Signals

PA[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.6 PB[3:0] — Port B I/O Signals

PB[3:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.7 PC[7:0] — Port C I/O Signals

PC[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull devices are disabled.

### 1.7.2.8 PD[7:0] — Port D I/O Signals

PD[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.9 PE[3:0] — Port E I/O Signals

PE[3:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.10 PF[7:0] — Port F I/O Signals

PF[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.11 PJ[3:0] — Port J I/O Signals

PJ[3:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are disabled.

### 1.7.2.12 PG[7:0] — Port G I/O Signals

PG[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.13 PH[7:0] — Port H I/O Signals

PH[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull-down devices are enabled.

### 1.7.2.14 PP[7:0] — Port P I/O Signals

PP[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull devices are disabled.

### 1.7.2.15 PS[7:0] / KWS[7:0] — Port S I/O signals

PS[7:0] are general-purpose input or output signals. The signals can be configured on per signal basis as interrupt inputs with wake-up capability (KWS[7:0]). These signals can have a pull-up or pull-down device selected and enabled on per signal basis. The signals can be configured on per signals basis as open drain output. Out of reset the pull-up devices are enabled.

### 1.7.2.16 PT[7:0] / KWT[7:0] — Port T I/O signals

PT[7:0] are general-purpose input or output signals. The signals can be configured on per signal basis as interrupt inputs with wake-up capability (KWT[7:0]). These signals can have a pull-up or pull-down device selected and enabled on per signal basis. Out of reset the pull devices are disabled.

### 1.7.2.17 PU[7:0] — Port U I/O Signals

PU[7:0] are general-purpose input or output signals. They can have a pull-up or pull-down device selected and enabled on per signal basis. They can have a slew rate enabled per signal basis also. Out of reset the pull devices are disabled.

### 1.7.2.18 AN0_[7:0] — ADC0 Input Signals

AN0_[7:0] are the analog inputs of the Analog-to-Digital Converters.

### 1.7.2.19 VRH, VRL — ADC0 Reference Signals

VRH and VRL are the reference voltage input pins for the analog-to-digital converter.

### 1.7.2.20 SPI0 Signals

#### 1.7.2.20.1 $\overline{SS}$0 Signal

This signal is associated with the slave select SS functionality of the serial peripheral interface SPI0.

#### 1.7.2.20.2 SCK0 Signal

This signal is associated with the serial clock SCK functionality of the serial peripheral interface SPI0.

#### 1.7.2.20.3 MISO0 Signal

This signal is associated with the MISO functionality of the serial peripheral interface SPI0. This signal acts as master input during master mode or as slave output during slave mode.

#### 1.7.2.20.4 MOSI0 Signal

This signal is associated with the MOSI functionality of the serial peripheral interface SPI0. This signal acts as master output during master mode or as slave input during slave mode

### 1.7.2.21 SCI[1:0] Signals

#### 1.7.2.21.1 RXD[1:0] Signals

These signals are associated with the receive functionality of the serial communication interfaces (SCI[1:0]).

#### 1.7.2.21.2 TXD[1:0] Signals

These signals are associated with the transmit functionality of the serial communication interfaces (SCI[1:0]).

### 1.7.2.22 CAN0 Signals

#### 1.7.2.22.1 RXCAN0 Signal

This signal is associated with the receive functionality of the scalable controller area network controller (MSCAN0).

#### 1.7.2.22.2 TXCAN0 Signal

This signal is associated with the transmit functionality of the scalable controller area network controller (MSCAN0).

### 1.7.2.23 Timer IOC0_[7:0] & IOC1_[7:0] Signals

The signals IOC0_[7:0] are associated with the input capture or output compare functionality of the timer (TIM0) module.

The signals IOC1_[7:0] are associated with the input capture or output compare functionality of the timer (TIM1) module.

### 1.7.2.24 PWM[7:0] Signals

The signals PWM[7:0] are associated with the PWM module digital channel outputs.

### 1.7.2.25 LCD Signals

#### 1.7.2.25.1 FP[39:0] Signals

These signals are associated with the segment LCD frontplane driver output.

#### 1.7.2.25.2 BP[3:0] Signals

These signals are associate the segment LCD backplane driver output.

### 1.7.2.26 RTC Signals

#### 1.7.2.26.1 RTC_CAL Signal

The signal can be the RTC output clock CALCLK for external clock calibration or external 1HZ standard clock input for on chip clock calibration.

### 1.7.2.27 SSG0 Signals

#### 1.7.2.27.1 SGT0 Signals

The signal is from SSG0 output, it contain tone or tone mixed with amplitude digital output.

#### 1.7.2.27.2 SGA0 Signals

The signal is from SSG0 output, it contain the amplitude digital output.

### 1.7.2.28 IIC0 Signals

#### 1.7.2.28.1 SDA0 Signal

This signal is associated with the serial data pin of IIC0.

#### 1.7.2.28.2 SCL0 Signal

This signal is associated with the serial clock pin of IIC0.

### 1.7.2.29 MC Signals

#### 1.7.2.29.1 M0C0M, M0C0P, M0C1M and M0C1P Signals

These signal are associated with the high current PWM out pin for the motor driver.

#### 1.7.2.29.2 M1C0M, M1C0P, M1C1M and M1C1P Signals

These signal are associated with the high current PWM out pin for the motor driver.

### 1.7.2.30 SSD[1:0] Signals

#### 1.7.2.30.1 M0COSM, M0COSP, M0SINM and M0SINP Signals

These signal are used to measure the back EMF to calibrate the pointer reset position which are associated with SSD[0].

#### 1.7.2.30.2 M1COSM, M1COSP, M1SINM and M1SINP Signals

These signal are used to measure the back EMF to calibrate the pointer reset position which are associated with SSD[1].

### 1.7.2.31 Interrupt Signals — $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$

$\overline{\text{IRQ}}$ is a maskable level or falling edge sensitive input. $\overline{\text{XIRQ}}$ is a non-maskable level-sensitive interrupt.

### 1.7.2.32 Oscillator and Clock Signals

#### 1.7.2.32.1 4-20MHz main Oscillator Pins — EXTAL and XTAL

EXTAL and XTAL are the crystal driver. On reset, the OSC is not enabled, all the device clocks are derived from the internal reference clock. EXTAL is the oscillator input. XTAL is the oscillator output.

#### 1.7.2.32.2 32.768kHz Oscillator Pins — 32K_EXTAL and 32K_XTAL

32K_EXTAL and 32K_XTAL are the 32.768KHZ crystal driver. On reset the OSC is not enabled. 32K_EXTAL is the oscillator input. 32K_XTAL is the oscillator output. Figure 1-3 is the 32K OSC connection diagram. Refer to the Appendix Table K-1., “OSC32K DC Electrical Specifications for the $C_x$, $C_y$ and $R_F$ requirement. Both RTC and LCD clock source can from the 32K OSC. The OSC enable control is from the RTC. If the RTCCTL2[CLKSRC] is set, then it will enable the 32K OSC. After enable the OSC, it needs to wait enough time before enable the RTC and LCD. Refer to Appendix Table K-2., “OSC32K Frequency Specifications for the startup time requirement.

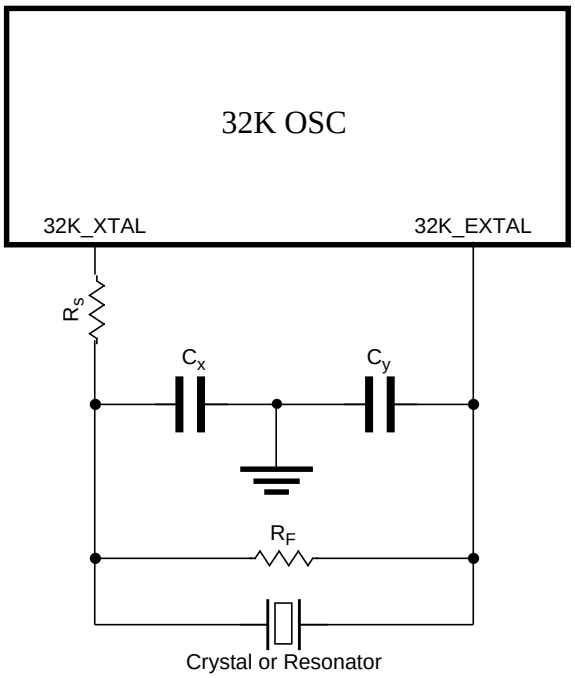



**Figure 1-3. 32K OSC Crystal/Resonator Connection**

#### 1.7.2.32.3 API_EXTCLK

This signal is associated with the output of the API.

#### 1.7.2.32.4 ECLK

This signal is associated with the output of the divided bus clock (ECLK).

**NOTE**

This feature is only intended for debug purposes at room temperature.
It must not be used for clocking external devices in an application.

### 1.7.2.33 BDC and Debug Signals

#### 1.7.2.33.1 BKGD — Background Debug signal

The BKGD signal is used as a pseudo-open-drain signal for the background debug communication. The BKGD signal has an internal pull-up device.

#### 1.7.2.33.2 PDO — Profiling Data Output

This is the profiling data output signal used when the DBG module profiling feature is enabled. This signal is output only and provides a serial, encoded data stream that can be used by external development tools to reconstruct the internal CPUcode flow.

#### 1.7.2.33.3 PDOCLK — Profiling Data Output Clock

This is the PDO clock signal used when the DBG module profiling feature is enabled. This signal is output only. During code profiling this is the clock signal that can be used by external development tools to sample the PDO signal.

#### 1.7.2.33.4 DBGEEV — External Event Input

This signal is the DBG external event input. It is input only. Within the DBG module, it allows an external event to force a state sequencer transition, or trace buffer entry, or to gate trace buffer entries. A falling edge at the external event signal constitutes an event. Rising edges have no effect. The maximum frequency of events is half the internal core bus frequency.

## 1.7.3 VSENSE - Voltage Sensor Input

This pin can be connected to the supply (Battery) line for voltage measurements. The voltage present at this input is scaled down by an internal voltage divider, and can be routed to the internal ADC via an analog multiplexer. The pin itself is protected against reverse battery connections. To protect the pin from external fast transients an external resistor is needed.

## 1.7.4 BCTL

BCTL provides the base current of an external bipolar of the VDDM, VDDA and VDDX supplies.

## 1.7.5 Power Supply Pins

The power and ground pins are described below. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible.

**NOTE**

All ground pins must be connected together in the application.

### 1.7.5.1 VDDX1, VDDX2, VDDX3, VSSX1, VSSX2, VSSX3 — Digital I/O Power and Ground Pins

VDDX1 is a dedicated voltage regulator output for the digital I/O drivers. It must be connected externally to the VDDX2 and VDDX3 pin, which supplies the VDDX domain pads. The VSSX1, VSSX2 and VSSX3 pins are the ground pin for the digital I/O drivers. The VDDX1 and VDDX2 are internal connected by metal, VDDX3 is not connect to VDDX1 or VDDX2 internally.

Bypass requirements on VDDX1/VSSX1, VDDX2/VSSX2 and VDDX3/VSSX3 depend on how heavily the MCU pins are loaded.

### 1.7.5.2 VDDA, VSSA — External Power Supply Pins for ADC and VREG

These are the power supply and ground pins for the analog-to-digital converter and the voltage regulator.

### 1.7.5.3 VDDM1, VSSM1 — External Power Supply Pins for Motor PAD

These are the power supply and ground pins for the motor driver pads. It should be supply by external power transistor.

### 1.7.5.4 VLCD- Power Supply Reference Pin for LCD driver

VLCD is the voltage reference pin for the LCD driver. Adjusting the voltage on this pin will chan1ge the display contrast.

### 1.7.5.5 VSS1, VSS2 — Core Ground Pin

The VDD voltage supply of nominally 1.8V is generated by the internal voltage regulator. The return current path is through the VSS1 and VSS2 pin.

### 1.7.5.6 VSS3 — Padring Ground Pin

VSS3 is the ground pin for the padring.

### 1.7.5.7 VSUP — Voltage Supply Pin for Voltage Regulator

VSUP is the 12V supply voltage pin for the on chip voltage regulator. This is the voltage supply input from which the voltage regulator generates the on chip voltage supplies. It must be protected externally against a reverse battery connection.

## 1.8 Package and Pinouts

The MC9S12ZVHY-Family will be offered in 100 pin and 144 pin LQFP packages.

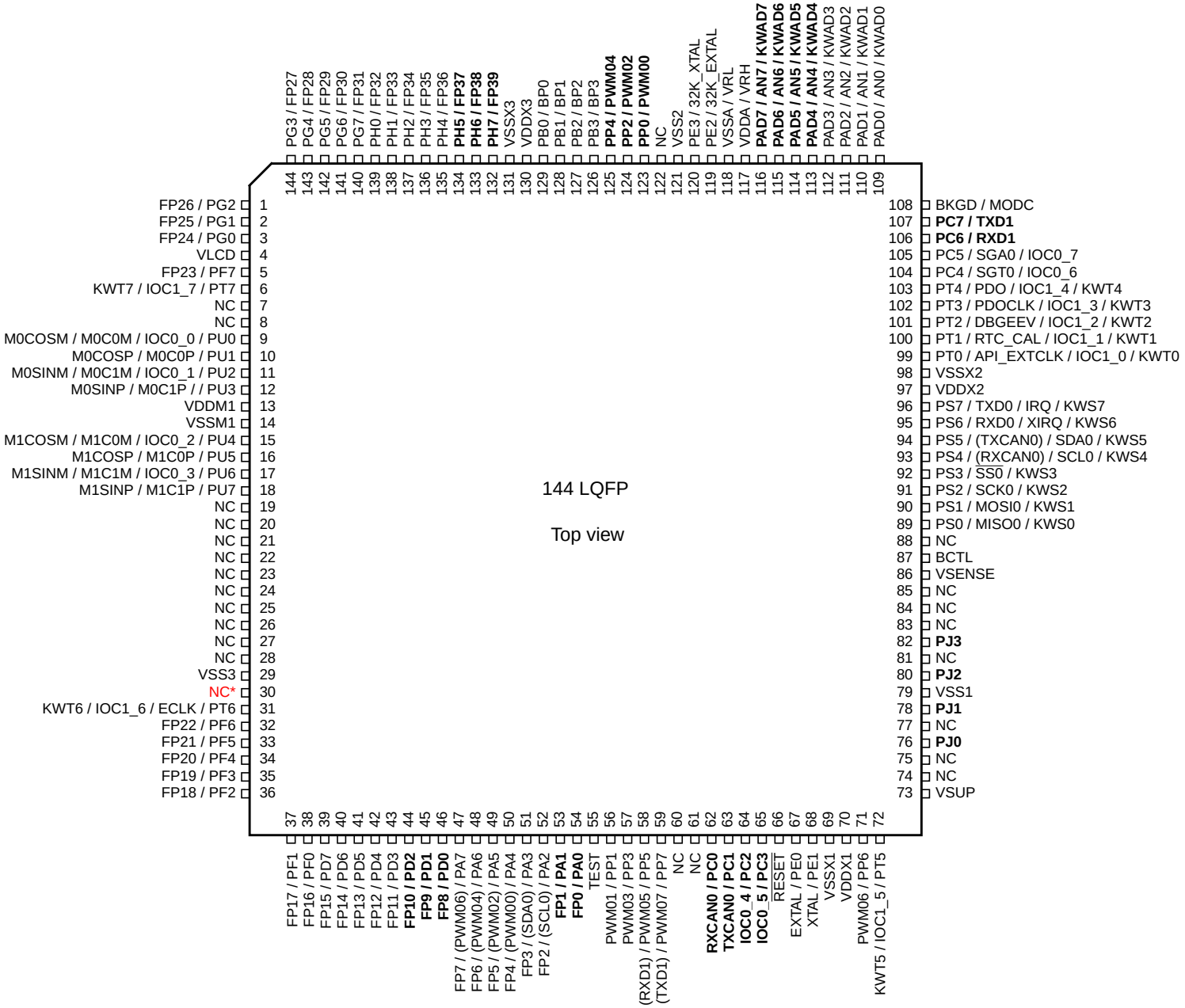



**Figure 1-4. MC9S12ZVHY 144-pin LQFP pin out**

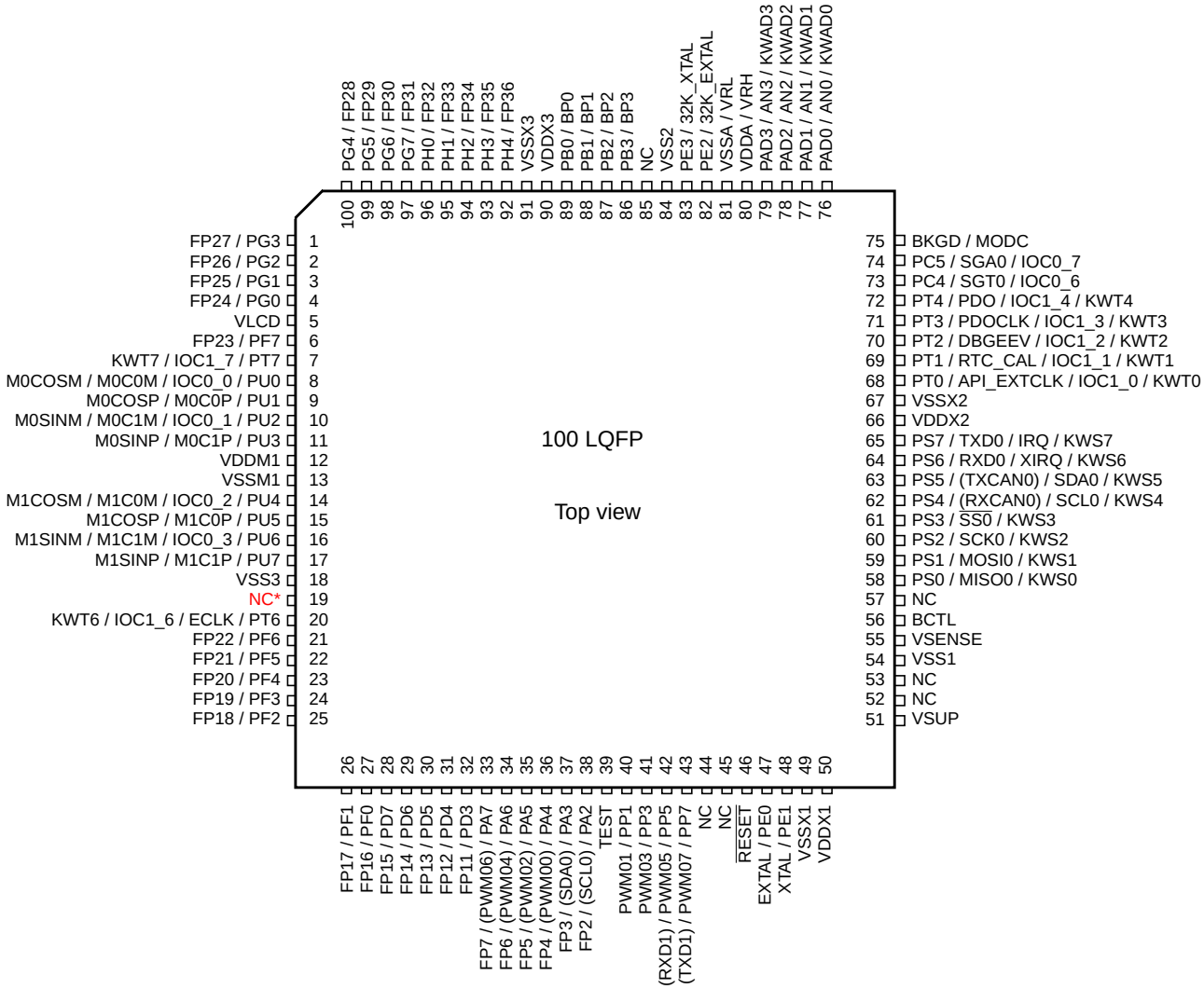



**Figure 1-5. MC9S12ZVHY 100-pin LQFP pin out**

**NOTE**

**Dont’t connect NC* to any VDD/VSS, it must keep floating**

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 1 | 2 | PG2 | FP26 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 2 | 3 | PG1 | FP25 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 3 | 4 | PG0 | FP24 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 4 | 5 | VLCD | — | — | — | — | — | VDDX | — | — |
| 5 | 6 | PF7 | FP23 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 6 | 7 | PT7 | IOC1_7 | KWT7 | — | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 7 | — | — | — | — | — | — | — | — | — | — |
| 8 | — | — | — | — | — | — | — | — | — | — |
| 9 | 8 | PU0 | IOC0_0 | M0C0M | M0COSM | — | — | VDDM | PERU/ PPSU | Disabled |
| 10 | 9 | PU1 | — | M0C0P | M0COSP | — | — | VDDM | PERU/ PPSU | Disabled |
| 11 | 10 | PU2 | IOC0_1 | M0C1M | M0SINM | — | — | VDDM | PERU/ PPSU | Disabled |
| 12 | 11 | PU3 | — | M0C1P | M0SINP | — | — | VDDM | PERU/ PPSU | Disabled |
| 13 | 12 | VDDM1 | — | — | — | — | — | — | — | — |
| 14 | 13 | VSSM1 | — | — | — | — | — | VDDM | — | — |
| 15 | 14 | PU4 | IOC0_2 | M1C0M | M1COSM | — | — | VDDM | PERU/ PPSU | Disabled |
| 16 | 15 | PU5 | — | M1C0P | M1COSP | — | — | VDDM | PERU/ PPSU | Disabled |
| 17 | 16 | PU6 | IOC0_3 | M1C1M | M1SINM | — | — | VDDM | PERU/ PPSU | Disabled |
| 18 | 17 | PU7 | — | M1C1P | M1SINP | — | — | VDDM | PERU/ PPSU | Disabled |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 19 | — | — | — | — | — | — | — | — | — | — |
| 20 | — | — | — | — | — | — | — | — | — | — |
| 21 | — | — | — | — | — | — | — | — | — | — |
| 22 | — | — | — | — | — | — | — | — | — | — |
| 23 | — | — | — | — | — | — | — | — | — | — |
| 24 | — | — | — | — | — | — | — | — | — | — |
| 25 | — | — | — | — | — | — | — | — | — | — |
| 26 | — | — | — | — | — | — | — | — | — | — |
| 27 | — | — | — | — | — | — | — | — | — | — |
| 28 | — | — | — | — | — | — | — | — | — | — |
| 29 | 18 | VSS3 | — | — | — | — | — | — | — | — |
| 30 | 19 | NC*[1] | — | — | — | — | — | — | — | — |
| 31 | 20 | PT6 | ECLK | IOC1_6 | KWT6 | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 32 | 21 | PF6 | FP22 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 33 | 22 | PF5 | FP21 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 34 | 23 | PF4 | FP20 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 35 | 24 | PF3 | FP19 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 36 | 25 | PF2 | FP18 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 37 | 26 | PF1 | FP17 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 38 | 27 | PF0 | FP16 | — | — | — | — | VDDX | PERF/ PPSF | Pull Down |
| 39 | 28 | PD7 | FP15 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 40 | 29 | PD6 | FP14 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 41 | 30 | PD5 | FP13 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 42 | 31 | PD4 | FP12 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 43 | 32 | PD3 | FP11 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 44 | — | PD2 | FP10 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 45 | — | PD1 | FP9 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 46 | — | PD0 | FP8 | — | — | — | — | VDDX | PERD/ PPSD | Pull Down |
| 47 | 33 | PA7 | (PWM6) | FP7 | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 48 | 34 | PA6 | (PWM4) | FP6 | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 49 | 35 | PA5 | (PWM2) | FP5 | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 50 | 36 | PA4 | (PWM0) | FP4 | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 51 | 37 | PA3 | (SDA0) | FP3 | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 52 | 38 | PA2 | (SCL0) | FP2 | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 53 | — | PA1 | FP1 | — | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 54 | — | PA0 | FP0 | — | — | — | — | VDDX | PERA/ PPSA | Pull Down |
| 55 | 39 | TEST | — | — | — | — | — | VDDX | — | — |
| 56 | 40 | PP1 | PWM1 | — | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 57 | 41 | PP3 | PWM3 | — | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 58 | 42 | PP5 | PWM5 | (RXD1) | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 59 | 43 | PP7 | PWM7 | (TXD1) | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 60 | 44 | — | — | — | — | — | — | — | — | — |
| 61 | 45 | — | — | — | — | — | — | — | — | — |
| 62 | — | PC0 | RXCAN0 | — | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 63 | — | PC1 | TXCAN0 | — | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 64 | — | PC2 | IOC0_4 | — | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 65 | — | PC3 | IOC0_5 | — | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 66 | 46 | RESET | — | — | — | — | — | $V_{DDX}$ | TEST pin | Pull Up |
| 67 | 47 | PE0 | EXTAL | — | — | — | — | $V_{DDX}$ | PERE/ PPSE | Pull Down |
| 68 | 48 | PE1 | XTAL | — | — | — | — | $V_{DDX}$ | PERE/ PPSE | Pull Down |
| 69 | 49 | VSSX1 | — | — | — | — | — | — | — | — |
| 70 | 50 | VDDX1 | — | — | — | — | — | $V_{DDX}$ | — | — |
| 71 | — | PP6 | PWM6 | — | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 72 | — | PT5 | IOC1_5 | KWT5 | — | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 73 | 51 | VSUP | — | — | — | — | — | $V_{SUP}$ | — | — |
| 74 | 52 | — | — | — | — | — | — | — | — | — |
| 75 | 53 | — | — | — | — | — | — | — | — | — |
| 76 | — | PJ0 | — | — | — | — | — | VDDX | PERJ/ PPSJ | Disabled |
| 77 | — | — | — | — | — | — | — | — | — | — |
| 78 | — | PJ1 | — | — | — | — | — | VDDX | PERJ/ PPSJ | Disabled |
| 79 | 54 | VSS1 | — | — | — | — | — | — | — | — |
| 80 | — | PJ2 | — | — | — | — | — | VDDX | PERJ/ PPSJ | Disabled |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 81 | — | — | — | — | — | — | — | — | — | — |
| 82 | — | PJ3 | — | — | — | — | — | VDDX | PERJ/ PPSJ | Disabled |
| 83 | — | — | — | — | — | — | — | — | — | — |
| 84 | — | — | — | — | — | — | — | — | — | — |
| 85 | — | — | — | — | — | — | — | — | — | — |
| 86 | 55 | VSENSE | — | — | — | — | — | — | — | — |
| 87 | 56 | BCTL | — | — | — | — | — | — | — | — |
| 88 | 57 | — | — | — | — | — | — | — | — | — |
| 89 | 58 | PS0 | MISO0 | KWS0 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 90 | 59 | PS1 | MOSI0 | KWS1 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 91 | 60 | PS2 | SCK0 | KWS2 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 92 | 61 | PS3 | $\overline{SS0}$ | KWS3 | — | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 93 | 62 | PS4 | (RXCAN 0) | SCL0 | KWS4 | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 94 | 63 | PS5 | (TXCAN 0) | SDA0 | KWS5 | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 95 | 64 | PS6 | RXD0 | $\overline{XIRQ}$ | KWS6 | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 96 | 65 | PS7 | TXD0 | $\overline{IRQ}$ | KWS7 | — | — | $V_{DDX}$ | PERS/ PPSS | Pull Up |
| 97 | 66 | VDDX2 | — | — | — | — | — | $V_{DDX}$ | — | — |
| 98 | 67 | VSSX2 | — | — | — | — | — | — | — | — |
| 99 | 68 | PT0 | API_EXTCLK | IOC1_0 | KWT0 | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 100 | 69 | PT1 | RTC_CAL | IOC1_1 | KWT1 | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 101 | 70 | PT2 | DBGEEV | IOC1_2 | KWT2 | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 102 | 71 | PT3 | PDOCLK | IOC1_3 | KWT3 | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 103 | 72 | PT4 | PDO | IOC1_2 | KWT2 | — | — | $V_{DDX}$ | PERT/ PPST | Disabled |
| 104 | 73 | PC4 | SGT0 | IOC0_6 | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 105 | 74 | PC5 | SGA0 | IOC0_7 | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 106 | — | PC6 | RXD1 | — | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 107 | — | PC7 | TXD1 | — | — | — | — | $V_{DDX}$ | PERC/ PPSC | Disabled |
| 108 | 75 | BKGD | MODC | — | — | — | — | VDDX | | Pull Up |
| 109 | 76 | PAD0 | AN0_0 | KWAD0 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 110 | 77 | PAD1 | AN0_1 | KWAD1 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 111 | 78 | PAD2 | AN0_2 | KWAD2 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 112 | 79 | PAD3 | AN0_3 | KWAD3 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 113 | — | PAD4 | AN0_4 | KWAD4 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 114 | — | PAD5 | AN0_5 | KWAD5 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 115 | — | PAD6 | AN0_6 | KWAD6 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 116 | — | PAD7 | AN0_7 | KWAD7 | — | — | — | $V_{DDA}$ | PER1AD/ PPS1AD | Disabled |
| 117 | 80 | VDDA/VRH | — | — | — | — | — | $V_{DDA}$ | — | — |
| 118 | 81 | VSSA/VRL | — | — | — | — | — | — | — | — |
| 119 | 82 | PE2 | 32K_EXTAL | — | — | — | — | $V_{DDX}$ | PERE/ PPSE | Pull Down |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 120 | 83 | PE3 | 32K_XTAL | — | — | — | — | $V_{DDX}$ | PERE/ PPSE | Pull Down |
| 121 | 84 | VSS2 | — | — | — | — | — | — | — | — |
| 122 | 85 | — | — | — | — | — | — | $V_{DD}$ | — | — |
| 123 | — | PP0 | PWM0 | — | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 124 | — | PP2 | PWM2 | — | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 125 | — | PP4 | PWM4 | — | — | — | — | $V_{DDX}$ | PERP/ PPSP | Disabled |
| 126 | 86 | PB3 | BP3 | — | — | — | — | $V_{DDX}$ | PERB/ PPSB | Pull Down |
| 127 | 87 | PB2 | BP2 | — | — | — | — | $V_{DDX}$ | PERB/ PPSB | Pull Down |
| 128 | 88 | PB1 | BP1 | — | — | — | — | $V_{DDX}$ | PERB/ PPSB | Pull Down |
| 129 | 89 | PB0 | BP0 | — | — | — | — | $V_{DDX}$ | PERB/ PPSB | Pull Down |
| 130 | 90 | VDDX3 | — | — | — | — | — | $V_{DDX}$ | — | — |
| 131 | 91 | VSSX3 | — | — | — | — | — | — | — | — |
| 132 | — | PH7 | FP39 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 133 | — | PH6 | FP38 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 134 | — | PH5 | FP37 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 135 | 92 | PH4 | FP36 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 136 | 93 | PH3 | FP35 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 137 | 94 | PH2 | FP34 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 138 | 95 | PH1 | FP33 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |

**Table 1-7. Pin Summary**

| LQFP Option | | | Function | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|---|---|
| 144 | 100 | Pin | 1st Func. | 2nd Func. | 3rd Func. | 4th Func. | 5th Func. | | CTRL | Reset State |
| 139 | 96 | PH0 | FP32 | — | — | — | — | VDDX | PERH/ PPSH | Pull Down |
| 140 | 97 | PG7 | FP31 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 141 | 98 | PG6 | FP30 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 142 | 99 | PG5 | FP29 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 143 | 100 | PG4 | FP28 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |
| 144 | 1 | PG3 | FP27 | — | — | — | — | VDDX | PERG/ PPSG | Pull Down |

1. Don’t connect NC* to any VDD/VSS signals, it must keep floating.

## 1.9 Modes of Operation

The MCU can operate in different modes. These are described in 1.9.1 Chip Configuration Modes.

The MCU can operate in different power modes to facilitate power saving when full system performance is not required. These are described in 1.9.3 Low Power Modes.

Some modules feature a software programmable option to freeze the module status whilst the background debug module is active to facilitate debugging. This is referred to as freeze mode at module level.

### 1.9.1 Chip Configuration Modes

The different modes and the security state of the MCU affect the debug features (enabled or disabled).

The operating mode out of reset is determined by the state of the MODC signal during reset (Table 1-8). The MODC bit in the MODE register shows the current operating mode and provides limited mode switching during operation. The state of the MODC signal is latched into this bit on the rising edge of $\overline{\text{RESET}}$

**Table 1-8. Chip Modes**

| Chip Modes | MODC |
|---|---|
| Normal single chip | 1 |

**Table 1-8. Chip Modes**

| Chip Modes | MODC |
|---|---|
| Special single chip | 0 |

#### 1.9.1.1 Normal Single-Chip Mode

This mode is intended for normal device operation. The opcode from the on-chip memory is being executed after reset (requires the reset vector to be programmed correctly). The processor program is executed from internal memory.

#### 1.9.1.2 Special Single-Chip Mode

This mode is used for debugging operation, boot-strapping, or security related operations. The background debug mode BDM is active on leaving reset in this mode.

### 1.9.2 Debugging Modes

The background debug mode (BDM) can be activated by the BDC module or directly when resetting into Special Single-Chip mode. Detailed information can be found in the BDC module section.

Writing to internal memory locations using the debugger, whilst code is running or at a breakpoint, can change the flow of application code.

The MC9S12ZVHY-Family supports BDC communication throughout the device Stop mode. During Stop mode, writes to control registers can alter the operation and lead to unexpected results. It is thus recommended not to reconfigure the peripherals during STOP using the debugger.

### 1.9.3 Low Power Modes

The device has two dynamic-power modes (run and wait) and two static low-power modes stop and pseudo stop). For a detailed description refer to Chapter 7, “S12 Clock, Reset and Power Management Unit (S12CPMU_UHV_V5).

- Dynamic power mode: Run
  - Run mode is the main full performance operating mode with the entire device clocked. The user can configure the device operating speed through selection of the clock source and the phase locked loop (PLL) frequency. To save power, unused peripherals must not be enabled.
- Dynamic power mode: Wait
  - This mode is entered when the CPU executes the WAI instruction. In this mode the CPU does not execute instructions. The internal CPU clock is switched off. All peripherals can be active in system wait mode. For further power consumption reduction, the peripherals can individually turn off their local clocks. Asserting $\overline{\text{RESET}}$, $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$, or any other interrupt that is not masked ends system wait mode.
- Static power mode Pseudo-stop:

— In this mode the system clocks are stopped but the oscillator is still running and the real time interrupt (RTI), watchdog (COP), RTC, LCD and Autonomous Periodic Interrupt (API) may be enabled. Other peripherals are turned off. This mode consumes more current than system STOP mode but, as the oscillator continues to run, the full speed wake up time from this mode is significantly shorter.

- Static power mode: Stop
  — The oscillator is stopped in this mode. By default, all clocks are switched off and all counters and dividers remain frozen. The Autonomous Periodic Interrupt (API), Key Wake-Up, RTC, CAN and the CAN physical layer transceiver modules may be enabled to wake the device.
  — If the BDC is enabled, in Stop mode, the VREG remains in full performance mode.With BDC enabled and BDCCIS bit set, then all clocks remain active during Stop mode to allow BDC access to internal peripherals. If the BDC is enabled and BDCCIS is clear, then the BDCSI clock remains active to allow BDC register access, but other clocks (with the exception of the API) are switched off. With the BDC enabled during Stop, the VREG full performance mode and clock activity lead to higher current consumption than with BDC disabled
  — If the BDC is enabled in Stop mode, then the voltage monitoring remains enabled.

## 1.10 Security

The MCU security mechanism prevents unauthorized access to the flash memory. It must be emphasized that part of the security must lie with the application code. An extreme example would be application code that dumps the contents of the internal memory. This would defeat the purpose of security. Also, if an application has the capability of downloading code through a serial port and then executing that code (e.g. an application containing bootloader code), then this capability could potentially be used to read the EEPROM and Flash memory contents even when the microcontroller is in the secure state. In this example, the security of the application could be enhanced by requiring a response authentication before any code can be downloaded.

Device security details are also described in the flash block description (Section 21.5, “Security”).

### 1.10.1 Features

The security features of the S12Z chip family are:

- Prevent external access of the non-volatile memories (Flash, EEPROM) content
- Restrict execution of NVM commands

### 1.10.2 Securing the Microcontroller

The chip can be secured by programming the security bits located in the options/security byte in the Flash memory array. These non-volatile bits keep the device secured through reset and power-down.

This byte can be erased and programmed like any other Flash location. Two bits of this byte are used for security (SEC[1:0]). The contents of this byte are copied into the Flash security register (FSEC) during a reset sequence.

The meaning of the security bits SEC[1:0] is shown in Table 1-9. For security reasons, the state of device security is controlled by two bits. To put the device in unsecured mode, these bits must be programmed to SEC[1:0] = ‘10’. All other combinations put the device in a secured mode. The recommended value to put the device in secured state is the inverse of the unsecured state, i.e. SEC[1:0] = ‘01’.

**Table 1-9. Security Bits**

| SEC[1:0] | Security State |
|---|---|
| 00 | 1 (secured) |
| 01 | 1 (secured) |
| ***10*** | ***0 (unsecured)*** |
| 11 | 1 (secured) |

**NOTE**

Please refer to the Section 21.5, “Security” for more security byte details.

## 1.10.3 Operation of the Secured Microcontroller

By securing the device, unauthorized access to the EEPROM and Flash memory contents is prevented. Secured operation has the following effects on the microcontroller:

### 1.10.3.1 Normal Single Chip Mode (NS)

- Background Debug Controller (BDC) operation is completely disabled.
- Execution of Flash and EEPROM commands is restricted (described in flash block description).

### 1.10.3.2 Special Single Chip Mode (SS)

- Background Debug Controller (BDC) commands are restricted
- Execution of Flash and EEPROM commands is restricted (described in flash block description).

In special single chip mode the device is in active BDM after reset. In special single chip mode on a secure device, only the BDC mass erase and BDC control and status register commands are possible. BDC access to memory mapped resources is disabled. The BDC can only be used to erase the EEPROM and Flash memory without giving access to their contents.

## 1.10.4 Unsecuring the Microcontroller

Unsecuring the microcontroller can be done using three different methods:

1. Backdoor key access
2. Reprogramming the security bits
3. Complete memory erase

### 1.10.4.1 Unsecuring the MCU Using the Backdoor Key Access

In normal single chip mode, security can be temporarily disabled using the backdoor key access method. This method requires that:

- The backdoor key has been programmed to a valid value.
- The KEYEN[1:0] bits within the Flash options/security byte select ‘enabled’.
- The application program programmed into the microcontroller has the capability to write to the backdoor key locations.

The backdoor key values themselves would not normally be stored within the application data, which means the application program would have to be designed to receive the backdoor key values from an external source (e.g. through a serial port).

The backdoor key access method allows debugging of a secured microcontroller without having to erase the Flash. This is particularly useful for failure analysis.

**NOTE**

No word backdoor key word is allowed to have the value 0x0000 or 0xFFFF.

## 1.10.5 Reprogramming the Security Bits

In normal single chip mode, security can also be disabled by erasing and reprogramming the security bits within Flash options/security byte to the unsecured value. Because the erase operation will erase the entire sector from (0xFF_FE00–0xFF_FFFF), the backdoor key and the interrupt vectors will also be erased; this method is not recommended for normal single chip mode. The application software can only erase and program the Flash options/security byte if the Flash sector containing the Flash options/security byte is not protected (see Flash protection). Thus Flash protection is a useful means of preventing this method. The microcontroller will enter the unsecured state after the next reset following the programming of the security bits to the unsecured value.

This method requires that:

- The application software previously programmed into the microcontroller has been designed to have the capability to erase and program the Flash options/security byte.
- The Flash sector containing the Flash options/security byte is not protected.

## 1.10.6 Complete Memory Erase

The microcontroller can be unsecured by erasing the entire EEPROM and Flash memory contents. If ERASE_FLASH is successfully completed, then the Flash unsecures the device and programs the security byte automatically.

## 1.11 Resets and Interrupts

### 1.11.1 Resets

Table 1-10. lists all reset sources and the vector locations. Resets are explained in detail in the Chapter 7, “S12 Clock, Reset and Power Management Unit (S12CPMU_UHV_V5)”.

**Table 1-10. Reset Sources and Vector Locations**

| Vector Address | Reset Source | CCR Mask | Local Enable |
|---|---|---|---|
| 0xFFFFFC | Power-On Reset (POR) | None | None |
| | Low Voltage Reset (LVR) | None | None |
| | External pin $\overline{\text{RESET}}$ | None | None |
| | Clock monitor reset | None | CPMUOSC(OSCE) |
| | COP watchdog reset | None | CR[2:0] in CPMUCOP register |

### 1.11.2 Interrupt Vectors

Table 1-11 lists all interrupt sources and vectors in the default order of priority. The interrupt module description provides an Interrupt Vector Base register (IVBR) to relocate the vectors.

**Table 1-11. Interrupt Vector Locations (Sheet 1 of 4)**

| Vector Address[1] | Interrupt Source | CCR Mask | Local Enable | Wake up from STOP | Wake up from WAIT |
|---|---|---|---|---|---|
| Vector base + 0x1F8 | Unimplemented page1 op-code trap (SPARE) | None | None | - | - |
| Vector base + 0x1F4 | Unimplemented page2 op-code trap (TRAP) | None | None | - | - |
| Vector base + 0x1F0 | Software interrupt instruction (SWI) | None | None | - | - |
| Vector base + 0x1EC | System call interrupt instruction (SYS) | None | None | - | - |
| Vector base + 0x1E8 | Machine exception | None | None | - | - |
| Vector base + 0x1E4 | Reserved | | | | |
| Vector base + 0x1E0 | Reserved | | | | |
| Vector base + 0x1DC | Spurious interrupt | — | None | - | - |
| Vector base + 0x1D8 | $\overline{\text{XIRQ}}$ interrupt request | X bit | None | Yes | Yes |
| Vector base + 0x1D4 | $\overline{\text{IRQ}}$ interrupt request | I bit | IRQCR(IRQEN) | Yes | Yes |
| Vector base + 0x1D0 | RTI time-out interrupt | I bit | CPMUINT (RTIE) | **See 7.6 Interrupts** | Yes |
| Vector base + 0x1CC | TIM0 timer channel 0 | I bit | TIM0TIE (C0I) | No | Yes |
| Vector base + 0x1C8 | TIM0 timer channel 1 | I bit | TIM0TIE (C1I) | No | Yes |

**Table 1-11. Interrupt Vector Locations (Sheet 2 of 4)**

| Vector Address[1] | Interrupt Source | CCR Mask | Local Enable | Wake up from STOP | Wake up from WAIT |
|---|---|---|---|---|---|
| Vector base + 0x1C4 | TIM0 timer channel 2 | I bit | TIM0TIE (C2I) | No | Yes |
| Vector base + 0x1C0 | TIM0 timer channel 3 | I bit | TIM0TIE (C3I) | No | Yes |
| Vector base + 0x1BC | TIM0 timer channel 4 | I bit | TIM0TIE (C4I) | No | Yes |
| Vector base + 0x1B8 | TIM0 timer channel 5 | I bit | TIM0TIE (C5I) | No | Yes |
| Vector base + 0x1B4 | TIM0 timer channel 6 | I bit | TIM0TIE (C6I) | No | Yes |
| Vector base + 0x1B0 | TIM0 timer channel 7 | I bit | TIM0TIE (C7I) | No | Yes |
| Vector base + 0x1AC | TIM0 timer overflow | I bit | TIM0TSCR2(TOI) | No | Yes |
| Vector base + 0x1A8 | TIM0 Pulse accumulator A overflow | I bit | TIM0PACTL(PAOVI) | No | Yes |
| Vector base + 0x1A4 | TIM0 Pulse accumulator input edge | I bit | TIM0PACTL(PAI) | No | Yes |
| Vector base + 0x1A0 | SPI0 | I bit | SPI0CR1 (SPIE, SPTIE) | No | Yes |
| Vector base + 0x19C | SCI0 | I bit | SCI0CR2 | Yes | Yes |
| Vector base + 0x198 | SCI1 | I bit | SCI1CR2 | Yes | Yes |
| Vector base + 0x194 | Reserved | | | | |
| Vector base + 0x190 | Reserved | | | | |
| Vector base + 0x18C | ADC0 Error | I bit | ADC0EIE(IA_EIE,CMD_EIE, EOL_EIE,TRIG_EIE,RSTAR_EIE,LDOK_EIE) | No | Yes |
| Vector base + 0x188 | ADC0 conversion sequence abort | I bit | ADC0IE(SEQAR_IE,CONIF_OIE) | No | Yes |
| Vector base + 0x184 | ADC0 conversion complete | I bit | ADC0CONIE[15:0] | No | Yes |
| Vector base + 0x180 | Oscillator status interrupt | I bit | CPMUINT (OSCIE) | No | Yes |
| Vector base + 0x17C | PLL lock interrupt | I bit | CPMUINT (LOCKIE) | No | Yes |
| Vector base + 0x178 to Vector base + 0x174 | Reserved | | | | |
| Vector base + 0x170 | RAM error | I bit | ECCIE (SBEEIE) | No | Yes |
| Vector base + 0x16C to Vector base + 0x168 | Reserved | | | | |
| Vector base + 0x164 | FLASH error | I bit | FERCNFG (SFDIE) | No | Yes |
| Vector base + 0x160 | FLASH command | I bit | FCNFG (CCIE) | No | Yes |
| Vector base + 0x15C | CAN0 wake-up | I bit | CAN0RIER (WUPIE) | Yes | Yes |
| Vector base + 0x158 | CAN0 errors | I bit | CAN0RIER (CSCIE, OVRIE) | No | Yes |
| Vector base + 0x154 | CAN0 receive | I bit | CAN0RIER (RXFIE) | No | Yes |
| Vector base + 0x150 | CAN0 transmit | I bit | CAN0RIER (TXEIE[2:0]) | No | Yes |

**Table 1-11. Interrupt Vector Locations (Sheet 3 of 4)**

| Vector Address[1] | Interrupt Source | CCR Mask | Local Enable | Wake up from STOP | Wake up from WAIT |
|---|---|---|---|---|---|
| Vector base + 0x14C to Vector base + 0x144 | Reserved | | | | |
| Vector base + 0x140 | BATS supply voltage monitor interrupt | I bit | BATIE (BVHIE,BVLIE) | No | Yes |
| Vector base + 0x13C to Vector base + 0x130 | Reserved | | | | |
| Vector base + 0x12C | Port T interrupt | I bit | PIET(PIET[7:0]) | Yes | Yes |
| Vector base + 0x128 | Reserved | | | | |
| Vector base + 0x124 | Port S interrupt | I bit | PIES(PIES[7:0]) | Yes | Yes |
| Vector base + 0x120 to Vector base + 0x108 | Reserved | | | | |
| Vector base + 0x104 | Low-voltage interrupt (LVI) | I bit | CPMUCTRL (LVIE) | No | Yes |
| Vector base + 0x100 | Autonomous periodical interrupt (API) | I bit | CPMUAPICTRL (APIE) | Yes | Yes |
| Vector base + 0xFC | High temperature interrupt | I bit | CPMUHTCTL(HTIE) | No | Yes |
| Vector base + 0xF8 | Reserved | | | | |
| Vector base + 0xF4 | Port AD interrupt | I bit | PIEADL(PIEADL[7:0]) | Yes | Yes |
| Vector base + 0xF0 to Vector base + 0xB8 | Reserved | | | | |
| Vector base + 0xB4 | IIC | I bit | IIC0IBCR(IBIE) | No | Yes |
| Vector base + 0xB0 | Reserved | | | | |
| Vector base + 0xAC | TIM1 timer channel 0 | I bit | TIM1TIE (C0I) | No | Yes |
| Vector base + 0xA8 | TIM1 timer channel 1 | I bit | TIM1TIE (C1I) | No | Yes |
| Vector base + 0xA4 | TIM1 timer channel 2 | I bit | TIM1TIE (C2I) | No | Yes |
| Vector base + 0xA0 | TIM1 timer channel 3 | I bit | TIM1TIE (C3I) | No | Yes |
| Vector base + 0x9C | TIM1 timer channel 4 | I bit | TIM1TIE (C4I) | No | Yes |
| Vector base + 0x98 | TIM1 timer channel 5 | I bit | TIM1TIE (C5I) | No | Yes |
| Vector base + 0x94 | TIM1 timer channel 6 | I bit | TIM1TIE (C6I) | No | Yes |
| Vector base + 0x90 | TIM1 timer channel 7 | I bit | TIM1TIE (C7I) | No | Yes |
| Vector base + 0x8C | TIM1 timer overflow | I bit | TIM1TSCR2(TOI) | No | Yes |
| Vector base + 0x88 | TIM1 Pulse accumulator A overflow | I bit | TIM1PACTL(PAOVI) | No | Yes |
| Vector base + 0x84 | TIM1 Pulse accumulator input edge | I bit | TIM1PACTL(PAI) | No | Yes |

**Table 1-11. Interrupt Vector Locations (Sheet 4 of 4)**

| Vector Address[1] | Interrupt Source | CCR Mask | Local Enable | Wake up from STOP | Wake up from WAIT |
|---|---|---|---|---|---|
| Vector base + 0x80 to Vector base + 0x7C | Reserved | | | | |
| Vector base + 0x78 | Motor Control Timer Overflow | I bit | MCCTL1(MCOCIE) | No | Yes |
| Vector base + 0x74 | SSD0 | I bit | MDC0CTL(MCZIE,AOVIE) | No | Yes |
| Vector base + 0x70 | SSD1 | I bit | MDC1CTL(MCZIE,AOVIE) | No | Yes |
| Vector base + 0x6C to Vector base + 0x68 | Reserved | | | | |
| Vector base + 0x64 | RTC | I bit | RTCCTL4(HRIE,MINIE,SECIE,COMPIE,TB0IE) | Yes | Yes |
| Vector base + 0x60 | SSG0 Ready For Next Data(RNDI) | I bit | SSG0IE(RNDIE) | No | Yes |
| Vector base + 0x5C to Vector base + 0x10 | Reserved | | | | |

1. 15 bits vector address based

## 1.11.3 Effects of Reset

When a reset occurs, MCU registers and control bits are initialized. For RTC block, some registers are power on reset only. Refer to the respective block sections for register reset states.

On each reset, the Flash module executes a reset sequence to load Flash configuration registers

### 1.11.3.1 Flash Configuration Reset Sequence Phase

On each reset, the Flash module will hold CPU activity while loading Flash module registers from the Flash memory. If double faults are detected in the reset phase, Flash module protection and security may be active on leaving reset. This is explained in more detail in the Section 21.6, “Initialization” of flash block.

### 1.11.3.2 Reset While Flash Command Active

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

### 1.11.3.3 I/O Pins

Refer to Chapter 2, “Port Integration Module (S12ZVHYPIMV1) for reset configurations of all peripheral module ports.

#### 1.11.3.4 RAM

The system RAM arrays, including their ECC syndromes, are initialized following a power on reset, but not out of warm reset. All other RAM arrays are not initialized out of any type of reset.

With the exception of resets resulting from low voltage conditions, the RAM content is unaltered by a reset occurrence.

## 1.12 COP Configuration

The COP time-out rate bits CR[2:0] and the WCOP bit in the CPMUCOP register are loaded from the Flash configuration field byte at global address 0xFF_FE0E during the reset sequence. See Table 1-12 and Table 1-13 for coding

**Table 1-12. Initial COP Rate Configuration**

| NV[2:0] in FOPT Register | CR[2:0] in COPCTL Register |
|---|---|
| 000 | 111 |
| 001 | 110 |
| 010 | 101 |
| 011 | 100 |
| 100 | 011 |
| 101 | 010 |
| 110 | 001 |
| 111 | 000 |

**Table 1-13. Initial WCOP Configuration**

| NV[3] in FOPT Register | WCOP in COPCTL Register |
|---|---|
| 1 | 0 |
| 0 | 1 |

## 1.13 ADC0 Internal Channels

Table 1-14 lists the internal sources which are connected to these special conversion channels.

**Table 1-14. ADC0 Channel Assignment**

| ADC0CMD_1 CH_SEL[5:0] | | | | | | | |
|---|---|---|---|---|---|---|---|
| [5] | [4] | [3] | [2] | [1] | [0] | **Analog Input Channel** | **Usage** |
| 0 | 0 | 1 | 0 | 0 | 0 | Internal_0 | ADC temperature sensor |
| 0 | 0 | 1 | 0 | 0 | 1 | Internal_1 | Bandgap Voltage $V_{BG}$ or Vreg temperature sensor $V_{HT}$(see Section 7.3.2.14, "High Temperature Control Register (CPMUHTCTL)" on how to config) |
| 0 | 0 | 1 | 0 | 1 | 0 | Internal_2 | RESERVED |
| 0 | 0 | 1 | 0 | 1 | 1 | Internal_3 | RESERVED |
| 0 | 0 | 1 | 1 | 0 | 0 | Internal_4 | $V_{SENSE}$ or $V_{SUP}$ selectable in BATS module |
| 0 | 0 | 1 | 1 | 0 | 1 | Internal_5 | RESERVED |
| 0 | 0 | 1 | 1 | 1 | 0 | Internal_6 | RESERVED |
| 0 | 0 | 1 | 1 | 1 | 1 | Internal_7 | RESERVED |

## 1.14 The ADC0 VRH/VRL

The ADC0 offers two possible sources for both reference voltages VRH[1:0] and VRL[1:0]. On the MC9S12ZVHY-Family only VRH[1], VRL[1] sources are connected at device level(to VDDA, VSSA respectively), the VRH[0], VRL[0] sources are not connected. Thus the application must set both VRH_SEL and VRL_SEL in the ADC0CCMD_1 to 1.

## 1.15 The ADC0 Conversion Resolution

The MC9S12ZVHY-Family only support 10 and 8 bit converstion resolution, although ADC block guide still has 12 bit related descrition.

## 1.16 ADC Result Reference

MCUs of the MC9S12ZVHY-Family are able to measure the internal reference voltage $V_{BG}$(see Table 1-14). $V_{BG}$ is a constant voltage with a narrow distribution over temperature and external voltage supply (see Table A-16).
A 10-bit left justified[1] ADC conversion result of $V_{BG}$ is provided at address 0x1F_C040/0x1F_C041 in the NVM's IFR for reference.The measurement conditions of the reference conversion are listed in Section A.1.10, "ADC Conversion Result Reference"". By measuring the voltage $V_{BG}$ (see Table 1-14)

1. The format of the stored $V_{BG}$ reference value is still subject to change.

and comparing the result to the reference value in the IFR, it is possible to determine the ADC's reference voltage $V_{RH}$ in the application environment:

$$V_{RH} = \frac{StoredReference}{ConvertedReference} \bullet 5V$$

The exact absolute value of an analog conversion can be determined as follows:

$$Result = ConvertedADInput \bullet \frac{StoredReference \bullet 5V}{ConvertedReference \bullet 2^n}$$

With:

| | |
|---|---|
| ConvertedADInput: | Result of the analog to digital conversion of the desired pin |
| ConvertedReference: | Result of channel "Internal_1" conversion |
| StoredReference: | Value in IFR location 0x1F_C040/0x1F_C041 |
| n: | ADC resolution (10 bit) |

## 1.17 BDC Clock Source Connectivity

The BDC clock, BDCCLK, is mapped to the IRCCLK generated in CPMU module. The BDC clock, BDCFCLK, is mapped to the bus clock.

## 1.18 FTMRZ Connectivity

The soc_erase_all_req input to the flash module is driven directly by a BDC erase flash request resulting from the BDC ERASE_FLASH command.

The FTMRZ FCLKDIV register is forced to 0x05 by the BDC ERASE_FLASH command. This configures the clock frequency correctly for the initial bus frequency on leaving reset. The bus frequency must not be changed before launching the ERASE_FLASH command.

## 1.19 RTC Clock Source

The RTC has three clock source, the 32K OSC, main OSC or IRC, refer to Section 18.4.2, "RTC Control Register 2 (RTCCTL2)" for more information. When select main OSC, user need to config the registers in CPMU block, refer to CPMU block guide for more detailed information. And main OSC will be stop if silicon enter full stop mode. And if select the 1 MHz internal IRC clock, then the clock will be off when enter full stop or pseudo stop mode, the RTC function will be stop.

## 1.20 LCD Clock Source Connectivity

The LCD's clock is connected to the RTC's RTCCLK output. User need to set the RTCCTL2[RTCPS] in Section 18.4.2, "RTC Control Register 2 (RTCCTL2)" to get the expect RTCCLK frequency if it uses the main OSC as clock source.

## 1.21 32K OSC enable control

The 32K OSC enable is controlled by the RTCCTL2[CLKSRC] in Section 18.4.2, "RTC Control Register 2 (RTCCTL2)". Setting the bits to 2'b01 enable the 32K OSC, it also selects the 32K OSC as the source for LCD and RTC clock. RTCCTL2 is write one time only in NSC mode, once enable the 32K OSC, it will be not able to switch off.

# Chapter 2
# Port Integration Module (S12ZVHYPIMV1)

## Revision History

| Rev. No. (Item No.) | Date (Submitted By) | Sections Affected | Substantial Change(s) |
|---|---|---|---|
| V0.01 | Mar 2012 | | • Initial Version |
| V0.02 | Mar 2012 | | • fix typos |
| V0.04 | Oct 2012 | | • fix typos, add XIRQ function explain |

## 2.1 Introduction

### 2.1.1 Overview

The MC9S12ZVHY-Family port integration module establishes the interface between the peripheral modules and the I/O pins for all ports. It controls the electrical pin properties as well as the signal prioritization and multiplexing on shared pins.

This document covers:

- 8-pin port A associated with the LCD FP[7:0] and rerouting of PWM0, PWM2, PWM4, PWM6 channels and rerouting of IIC
- 4-pin port B associated with the LCD BP[3:0]
- 8-pin port C associated with TIM0_IOC[7:4], MSCAN0, SCI1, SSG0
- 8-pin port D associated with LCD FP[15:8]
- 4-pin port E associated with the external 4-16MHZ oscillator and 32.768KHZ oscillator
- 8-pin port F associated with LCD FP[23:16]
- 8-pin port G associated with LCD FP[31:24]
- 8-pin port H associated with LCD FP[39:32]
- 4-pin Port J
- 8-pin port P associated with 8 PWM channels; associated with the rerouting SCI1 function also.
- 8-pin port S associated with SCI0, IIC0, SPI0 modules and rerouting of MSCAN0. PS7 and PS6 also associated with $\overline{\text{IRQ}}$, $\overline{\text{XIRQ}}$ interrupt inputs; associated with the key wakeup functions .
- 8-pin port T with the key wakeup function and 8 TIM1 channels, also associated with

  - API_EXTCLK
  - DBG external signals PDO, PDOCLK and DBGEEV
  - ECLK output
- 8-pin port AD associated with 8 ADC0 channels; associated with the key wakeup function also
- 8-pin port U associated with SSD0, SSD1, 2 Motor controls and 4 TIM0 channels

Most I/O pins can be configured by register bits to select data direction and to enable and select pullup or pulldown devices.

**NOTE**

This document assumes the availability of all features (144-pin package option). Some functions are not available on lower pin count package options. Refer to the pin-out summary in the SOC Guide.

## 2.1.2 Features

The PIM includes these distinctive registers:

- Data registers and data direction registers for ports A, B, C, D, E, F, G, H, T, S, P, AD, U and J when used as general-purpose I/O
- Control registers to enable pull devices and select pullups/pulldowns on ports A, B, C, D, E, F, G, H, T, S, P, AD, U and J
- Single control register bit to enable pullups on BKGD pin
- Control register to enable open-drain (wired-or) mode on port S and port A (only PA3, PA2)
- Control register to enable digital input buffers on port AD
- Control register to enable the slew rate control on Port U.
- Interrupt flag register for pin interrupts on port S, T and AD
- Control register to configure $\overline{\text{IRQ}}$ pin operation
- Control register to enable ECLK output
- Control register to enable the RTC_CAL input or output
- Routing registers to support signal relocation on external pins and control internal routings:
  - IIC0 to alternative pins
  - SCI1 to alternative pins
  - MSCAN0 to alternative pins
  - PWM0, PWM2, PWM4, PWM6 to alternative pins
  - rerouting the RTC_CAL to TIM1 channel
  - rerouting the RXD0 and RXD1 to TIM1 channel for the baud rate detection

A standard port pin has the following minimum features:

- Input/output selection
- 5V output drive
- 5V digital and analog input

- Input with selectable pullup or pulldown device

Optional features supported on dedicated pins:

- Open drain for wired-or connections
- Interrupt input with glitch filtering
- Slew rate control on motor pads

## 2.2 External Signal Description

This section lists and describes the signals that do connect off-chip.

Table 2-1 shows all pins with the pins and functions that are controlled by the PIM. Routing options are denoted in parenthesis.

**NOTE**

If there is more than one function associated with a pin, the output priority is indicated by the position in the table from top (highest priority) to bottom (lowest priority).

**Table 2-1. Pin Functions and Priorities**

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| - | BKGD | MODC[2] | I | MODC input during $\overline{\text{RESET}}$ | | BKGD |
| | | BKGD | I/O | S12ZBDC communication | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| A | PA7 | FP7 | O | LCD FP7 signal | | GPIO |
| | | (PWM6) | O | PWM channel 6 | PWM6RR | |
| | | PTA[7] | I/O | General-purpose | | |
| | PA6 | FP6 | O | LCD FP6 signal | | |
| | | (PWM4) | O | PWM channel 4 | PWM4RR | |
| | | PTA[6] | I/O | General-purpose | | |
| | PA5 | FP5 | O | LCD FP5 signal | | |
| | | (PWM2) | O | PWM channel 2 | PWM2RR | |
| | | PTA[5] | I/O | General-purpose | | |
| | PA4 | FP4 | O | LCD FP4 signal | | |
| | | (PWM0) | O | PWM channel 0 | PWM6RR | |
| | | PTA[4] | I/O | General-purpose | | |
| | PA3 | FP3 | O | LCD FP3 signal | | |
| | | (SDA0) | I/O | SDA of IIC0 signal | IIC0RR | |
| | | PTA[3] | I/O | General-purpose | | |
| | PA2 | FP2 | O | LCD FP2 signal | | |
| | | (SCL0) | I/O | SCL of IIC0 signal | IIC0RR | |
| | | PTA[2] | I/O | General-purpose | | |
| | PA1 | FP1 | O | LCD FP1 signal | | |
| | | PTA[1] | I/O | General-purpose | | |
| | PA0 | FP0 | O | LCD FP0 signal | | |
| | | PTA[0] | I/O | General-purpose | | |
| B | PB3 | BP3 | O | LCD BP3 signals | | GPIO |
| | | PTB[3] | I/O | General-purpose | | |
| | PB2 | BP2 | O | LCD BP2 signal | | |
| | | PTB[2] | I/O | General-purpose | | |
| | PB1 | BP1 | O | LCD BP1 signal | | |
| | | PTB[1] | I/O | General-purpose | | |
| | PB0 | BP0 | O | LCD BP0 signal | | |
| | | PTB[0] | I/O | General-purpose | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| C | PC7 | TXD1 | O | TXD of SCI1 | | GPIO |
| | | PTC[7] | I/O | General-purpose | | |
| | PC6 | RXD1 | I | RXD of SCI1 | | |
| | | PTC[6] | I/O | General-purpose | | |
| | PC5 | IOC0_7 | O | TIM0 channel 7 | | |
| | | SGA0 | O | SGA of SSG0 | | |
| | | PTC[5] | I/O | General-purpose | | |
| | PC4 | IOC0_6 | O | TIM0 channel 6 | | |
| | | SGT0 | O | SGT of SSG0 | | |
| | | PTC[4] | I/O | General-purpose | | |
| | PC3 | IOC0_5 | O | TIM0 channel 5 | | |
| | | PTC[3] | I/O | General-purpose | | |
| | PC2 | IOC0_4 | O | TIM0 channel 4 | | |
| | | PTC[2] | I/O | General-purpose | | |
| | PC1 | TXCAN0 | O | TX of MSCAN0 | C0RR | |
| | | PTC[1] | I/O | General-purpose | | |
| | PC0 | RXCAN0 | I | RX of MSCAN0 | C0RR | |
| | | PTC[0] | I/O | General-purpose | | |
| D | PD7 | FP15 | O | LCD FP15 signal | | GPIO |
| | | PTD[7] | I/O | General-purpose | | |
| | PD6 | FP14 | O | LCD FP14 signal | | |
| | | PTD[6] | I/O | General-purpose | | |
| | PD5 | FP13 | O | LCD FP13 signal | | |
| | | PTD[5] | I/O | General-purpose | | |
| | PD4 | FP12 | O | LCD FP12 signal | | |
| | | PTD[4] | I/O | General-purpose | | |
| | PD3 | FP11 | O | LCD FP11 signal | | |
| | | PTD[3] | I/O | General-purpose | | |
| | PD2 | FP10 | O | LCD FP10 signal | | |
| | | PTD[2] | I/O | General-purpose | | |
| | PD1 | FP9 | O | LCD FP9 signal | | |
| | | PTD[1] | I/O | General-purpose | | |
| | PD0 | FP8 | O | LCD FP8 signal | | |
| | | PTD[0] | I/O | General-purpose | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| E | PE3 | 32K_XTAL | - | 32K OSC signal | | GPIO |
| | | PTE[3] | I/O | General-purpose | | |
| | PE2 | 32K_EXTAL | - | 32K OSC signal | | |
| | | PTE[2] | I/O | General-purpose | | |
| | PE1 | XTAL | - | CPMU OSC signal | | |
| | | PTE[1] | I/O | General-purpose | | |
| | PE0 | EXTAL | - | CPMU OSC signal | | |
| | | PTE[0] | I/O | General-purpose | | |
| F | PF7 | FP23 | O | LCD FP23 signal | | GPIO |
| | | PTF[7] | I/O | General-purpose | | |
| | PF6 | FP22 | O | LCD FP22 signal | | |
| | | PTF[6] | I/O | General-purpose | | |
| | PF5 | FP21 | O | LCD FP21 signal | | |
| | | PTF[5] | I/O | General-purpose | | |
| | PF4 | FP20 | O | LCD FP20 signal | | |
| | | PTF[4] | I/O | General-purpose | | |
| | PF3 | FP19 | O | LCD FP19 signal | | |
| | | PTF[3] | I/O | General-purpose | | |
| | PF2 | FP18 | O | LCD FP18 signal | | |
| | | PTF[2] | I/O | General-purpose | | |
| | PF1 | FP17 | O | LCD FP17 signal | | |
| | | PTF[1] | I/O | General-purpose | | |
| | PF0 | FP16 | O | LCD FP16 signal | | |
| | | PTF[0] | I/O | General-purpose | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| G | PG7 | FP31 | O | LCD FP31 signal | | GPIO |
| | | PTG[7] | I/O | General-purpose | | |
| | PG6 | FP30 | O | LCD FP30 signal | | |
| | | PTG[6] | I/O | General-purpose | | |
| | PG5 | FP29 | O | LCD FP29 signal | | |
| | | PTG[5] | I/O | General-purpose | | |
| | PG4 | FP28 | O | LCD FP28 signal | | |
| | | PTG[4] | I/O | General-purpose | | |
| | PG3 | FP27 | O | LCD FP27 signal | | |
| | | PTG[3] | I/O | General-purpose | | |
| | PG2 | FP26 | O | LCD FP26 signal | | |
| | | PTG[2] | I/O | General-purpose | | |
| | PG1 | FP25 | O | LCD FP25 signal | | |
| | | PTG[1] | I/O | General-purpose | | |
| | PG0 | FP24 | O | LCD FP24 signal | | |
| | | PTG[0] | I/O | General-purpose | | |
| H | PH7 | FP39 | O | LCD FP39 signal | | GPIO |
| | | PTH[7] | I/O | General-purpose | | |
| | PH6 | FP38 | O | LCD FP38 signal | | |
| | | PTH[6] | I/O | General-purpose | | |
| | PH5 | FP37 | O | LCD FP37 signal | | |
| | | PTH[5] | I/O | General-purpose | | |
| | PH4 | FP36 | O | LCD FP36 signal | | |
| | | PTH[4] | I/O | General-purpose | | |
| | PH3 | FP35 | O | LCD FP35 signal | | |
| | | PTH[3] | I/O | General-purpose | | |
| | PH2 | FP34 | O | LCD FP34 signal | | |
| | | PTH[2] | I/O | General-purpose | | |
| | PH1 | FP33 | O | LCD FP33 signal | | |
| | | PTH[1] | I/O | General-purpose | | |
| | PH0 | FP32 | O | LCD FP32 signal | | |
| | | PTH[0] | I/O | General-purpose | | |
| J | PJ3 | PTJ[3] | I/O | General-purpose | | GPIO |
| | PJ2 | PTJ[2] | I/O | General-purpose | | |
| | PJ1 | PTJ[1] | I/O | General-purpose | | |
| | PJ0 | PTJ[0] | I/O | General-purpose | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| P | PP7 | (TXD1) | O | TXD of SCI1 | SCI1RR | GPIO |
| | | PWM7 | O | PWM channel 7 | | |
| | | PP[7] | I/O | General-purpose | | |
| | PP6 | PWM6 | O | PWM channel 6 | | |
| | | PP[6] | I/O | General-purpose | | |
| | PP5 | (RXD1) | O | RXD of SCI1 | SCI1RR | |
| | | PWM5 | O | PWM channel 5 | | |
| | | PP[5] | I/O | General-purpose | | |
| | PP4 | PWM4 | O | PWM channel 4 | | |
| | | PP[4] | I/O | General-purpose | | |
| | PP3 | PWM3 | O | PWM channel 3 | | |
| | | PP[3] | I/O | General-purpose | | |
| | PP2 | PWM2 | O | PWM channel 2 | | |
| | | PP[2] | I/O | General-purpose | | |
| | PP1 | PWM1 | O | PWM channel 1 | | |
| | | PP[1] | I/O | General-purpose | | |
| | PP0 | PWM0 | O | PWM channel 0 | | |
| | | PP[0] | I/O | General-purpose | | |
| S | PS7 | $\overline{\text{IRQ}}$ | O | IRQ interrupt input | | GPIO |
| | | TXD0 | I/O | TXD of SCI0 | | |
| | | PTS[7]/KWS[7] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS6 | $\overline{\text{XIRQ}}$ | O | XIRQ interrupt input[3] | | |
| | | RXD0 | I/O | RXD of SCI0 | | |
| | | PTS[6]/KWS[6] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS5 | SDA0 | O | SDA of IIC0 | | |
| | | (TXCAN0) | I/O | TX of MSCAN0 | C0RR | |
| | | PTS[5]/KWS[5] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS4 | SCL0 | O | SCL of IIC0 | | |
| | | (RXCAN) | I/O | RX of MSCAN0 | C0RR | |
| | | PTS[4]/KWS[4] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS3 | $\overline{\text{SS0}}$ | I/O | SPI0 slave select | | |
| | | PTS[3]/KWS[3] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS2 | SCK0 | I/O | SPI0 serial clock | | |
| | | PTS[2]/KWS[2] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS1 | MOSI0 | I/O | SPI0 master out/slave in | | |
| | | PTS[1]/KWS[1] | I/O | General-purpose; with interrupt and wakeup | | |
| | PS0 | MISO0 | I/O | SPI0 master in/slave out | | |
| | | PTS[0]/KWS[0] | I/O | General-purpose; with interrupt and wakeup | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| T | PT7 | IOC1_7 | O | TIM1 channel 7 | | GPIO |
| | | PTT[7]/KWT[7] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT6 | IOC1_6 | I/O | TIM1 channel 6 | | |
| | | ECLK | O | Free running clock output | | |
| | | PTT[6]/KWT[6] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT5 | IOC1_5 | I/O | TIM1 channel 5 | | |
| | | PTT[5]/KWT[5] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT4 | IOC1_4 | I/O | TIM1 channel 4 | | |
| | | PDO | O | DBG profiling data output | | |
| | | PTT[4]/KWT[4] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT3 | IOC1_3 | I/O | TIM1 channel 3 | | |
| | | PDOCLK | O | DBG profiling clock | | |
| | | PTT[3]/KWT[3] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT2 | IOC1_2 | I/O | TIM1 channel 2 | | |
| | | DBGEEV | I | DBG external event input | | |
| | | PTT[2]/KWT[2] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT1 | IOC1_1 | I/O | TIM1 channel 1 | | |
| | | RTC_CAL | I/O | RTC CALCLK output or external 1HZ input | | |
| | | PTT[1]/KWT[1] | I/O | General-purpose; with interrupt and wakeup | | |
| | PT0 | (IOC1_0) | I/O | TIM1 channel 0 | T1IC0RR1-0 | |
| | | API_EXTCLK | O | API clock output | | |
| | | PTT[0]/KWT[0] | I/O | General-purpose; with interrupt and wakeup | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| AD | PAD7 | AN0_7 | I | ADC0 analog input 7 | | GPIO |
| | | PTADL[7]/ KWADL[7] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD6 | AN0_6 | I | ADC0 analog input 6 | | |
| | | PTADL[6]/ KWADL[6] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD5 | AN0_5 | I | ADC0 analog input 5 | | |
| | | PTADL[5]/ KWADL[5] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD4 | AN0_4 | I | ADC0 analog input 4 | | |
| | | PTADL[4]/ KWADL[4] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD3 | AN0_3 | I | ADC0 analog input 3 | | |
| | | PTADL[3]/ KWADL[3] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD2 | AN0_2 | I | ADC0 analog input 2 | | |
| | | PTADL[2]/ KWADL[2] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD1 | AN0_1 | I | ADC0 analog input 1 | | |
| | | PTADL[1]/ KWADL[1] | I/O | General-purpose; with interrupt and wakeup | | |
| | PAD0 | AN0_0 | I | ADC0 analog input 0 | | |
| | | PTADL[0]/ KWADL[0] | I/O | General-purpose; with interrupt and wakeup | | |

| Port | Pin Name | Pin Function & Priority[1] | I/O | Description | Routing Register | Pin Function after Reset |
|---|---|---|---|---|---|---|
| U | PU[7] | M1SINP | I/O | SSD1 Sine+ Node | | GPIO |
| | | M1C1P | O | Motor control output for motor 1 | | |
| | | PTU[7] | I/O | General purpose | | |
| | PU[6] | M1SINM | I/O | SSD1 Sine- Node | | |
| | | M1C1M | O | Motor control output for motor 1 | | |
| | | IOC0_3 | I/O | TIM0 channel 3 | | |
| | | PTU[6] | I/O | General purpose | | |
| | PU[5] | M1COSP | I/O | SSD1 Cosine+ Node | | |
| | | M1C0P | O | Motor control output for motor 1 | | |
| | | PTU[5] | I/O | General purpose | | |
| | PU[4] | M1COSM | I/O | SSD1 Cosine- Node | | |
| | | M1C0M | O | Motor control output for motor 1 | | |
| | | IOC0_2 | I/O | TIM0 channel2 | | |
| | | PTU[4] | I/O | General purpose | | |
| | PU[3] | M0SINP | I/O | SSD0 Sine+ Node | | |
| | | M0C1P | O | Motor control output for motor 0 | | |
| | | PTU[3] | I/O | General purpose | | |
| | PU[2] | M0SINM | I/O | SSD0 Sine- Node | | |
| | | M0C1M | O | Motor control output for motor 0 | | |
| | | IOC0_1 | I/O | TIM0 channel 1 | | |
| | | PTU[2] | I/O | General purpose | | |
| | PU[1] | M0COSP | I/O | SSD0 Cosine+ Node | | |
| | | M0C0P | O | Motor control output for motor 0 | | |
| | | PTU[1] | I/O | General purpose | | |
| | PU[0] | M0COSM | I/O | SSD0 Cosine- Node | | |
| | | M0C0M | O | Motor control output for motor 0 | | |
| | | IOC0_0 | I/O | TIM0 channel 0 | | |
| | | PTU[0] | I/O | General purpose | | |

1. Signals in parenthesis denote alternative module routing pins
2. Function active when $\overline{\text{RESET}}$ asserted.
3. The interrupt is enabled by clearing the X mask bit in the CPU CCR. The pin is forced to input upon first clearing of the X bit and is held in this state until reset. A stop or wait recovery with the X bit set (refer to S12ZCPU reference manual) is not available.

## 2.3 Memory Map and Register Definition

This section provides a detailed description of all port integration module registers.

## 2.3.1 Register Map

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0200 | MODRR0 | R | 0 | 0 | 0 | 0 | C0RR | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0201 | MODRR1 | R | 0 | 0 | 0 | 0 | PWM6RR | PWM4RR | PWM2RR | PWM0RR |
| | | W | | | | | | | | |
| 0x0202 | MODRR2 | R | 0 | 0 | SCI1RR | IIC0RR | 0 | 0 | T1IC0RR1 | T1IC0RR0 |
| | | W | | | | | | | | |
| 0x0203–0x0207 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0208 | ECLKCTL | R | NECLK | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0209 | IRQCR | R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x020A | PIMMISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CALCLKEN |
| | | W | | | | | | | | |
| 0x020B–0x020D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x020E | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x020F | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x0210–0x021F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0220 | PTA | R | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | W | | | | | | | | |
| 0x0221 | PTB | R | 0 | 0 | 0 | 0 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | W | | | | | | | | |
| 0x0222 | PTIA | R | PTIA7 | PTIA6 | PTIA5 | PTIA4 | PTIA3 | PTIA2 | PTIA1 | PTIA0 |
| | | W | | | | | | | | |
| 0x0223 | PTIB | R | 0 | 0 | 0 | 0 | PTIB3 | PTIB2 | PTIB1 | PTIB0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0224 | DDRA | R | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | W | | | | | | | | |
| 0x0225 | DDRB | R | 0 | 0 | 0 | 0 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | W | | | | | | | | |
| 0x0226 | PERA | R | PERA7 | PERA6 | PERA5 | PERA4 | PERA3 | PERA2 | PERA1 | PERA0 |
| | | W | | | | | | | | |
| 0x0227 | PERB | R | 0 | 0 | 0 | 0 | PERB3 | PERB2 | PERB1 | PERB0 |
| | | W | | | | | | | | |
| 0x0228 | PPSA | R | PPSA7 | PPSA6 | PPSA5 | PPSA4 | PPSA3 | PPSA2 | PPSA1 | PPSA0 |
| | | W | | | | | | | | |
| 0x0229 | PPSB | R | 0 | 0 | 0 | 0 | PPSB3 | PPSB2 | PPSB1 | PPSB0 |
| | | W | | | | | | | | |
| 0x022A–0x023D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x023E | WOMA | R | 0 | 0 | 0 | 0 | WOMA3 | WOMA2 | 0 | 0 |
| | | W | | | | | | | | |
| 0x023F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0240 | PTC | R | PTC7 | PTC6 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | W | | | | | | | | |
| 0x0241 | PTD | R | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| | | W | | | | | | | | |
| 0x0242 | PTIC | R | PTIC7 | PTIC6 | PTIC5 | PTIC4 | PTIC3 | PTIC2 | PTIC1 | PTIC0 |
| | | W | | | | | | | | |
| 0x0243 | PTID | R | PTID7 | PTID6 | PTID5 | PTID4 | PTID3 | PTID2 | PTID1 | PTID0 |
| | | W | | | | | | | | |
| 0x0244 | DDRC | R | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | W | | | | | | | | |
| 0x0245 | DDRD | R | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| | | W | | | | | | | | |
| 0x0246 | PERC | R | PERC7 | PERC6 | PERC5 | PERC4 | PERC3 | PERC2 | PERC1 | PERC0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0247 | PERD | R | PERD7 | PERD6 | PERD5 | PERD4 | PERD3 | PERD2 | PERD1 | PERD0 |
| | | W | | | | | | | | |
| 0x0248 | PPSC | R | PPSC7 | PPSC6 | PPSC5 | PPSC4 | PPSC3 | PPSC2 | PPSC1 | PPSC0 |
| | | W | | | | | | | | |
| 0x0249 | PPSD | R | PPSD7 | PPSD6 | PPSD5 | PPSD4 | PPSD3 | PPSD2 | PPSD1 | PPSD0 |
| | | W | | | | | | | | |
| 0x024A–0x025F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0260 | PTE | R | 0 | 0 | 0 | 0 | PTE3 | PTE2 | PTE1 | PTE0 |
| | | W | | | | | | | | |
| 0x0261 | PTF | R | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | | W | | | | | | | | |
| 0x0262 | PTIE | R | 0 | 0 | 0 | 0 | PTIE3 | PTIE2 | PTIE1 | PTIE0 |
| | | W | | | | | | | | |
| 0x0263 | PTIF | R | PTIF7 | PTIF6 | PTIF5 | PTIF4 | PTIF3 | PTIF2 | PTIF1 | PTIF0 |
| | | W | | | | | | | | |
| 0x0264 | DDRE | R | 0 | 0 | 0 | 0 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| | | W | | | | | | | | |
| 0x0265 | DDRF | R | DDRF7 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | W | | | | | | | | |
| 0x0266 | PERE | R | 0 | 0 | 0 | 0 | PERE3 | PERE2 | PERE1 | PERE0 |
| | | W | | | | | | | | |
| 0x0267 | PERF | R | PERF7 | PERF6 | PERF5 | PERF4 | PERF3 | PERF2 | PERF1 | PERF0 |
| | | W | | | | | | | | |
| 0x0268 | PPSE | R | 0 | 0 | 0 | 0 | PPSE3 | PPSE2 | PPSE1 | PPSE0 |
| | | W | | | | | | | | |
| 0x0269 | PPSF | R | PPSF7 | PPSF6 | PPSF5 | PPSF4 | PPSF3 | PPSF2 | PPSF1 | PPSF0 |
| | | W | | | | | | | | |
| 0x026A–0x027F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0280 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0281 | PTADL | R | PTADL7 | PTADL6 | PTADL5 | PTADL4 | PTADL3 | PTADL2 | PTADL1 | PTADL0 |
| | | W | | | | | | | | |
| 0x0282 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0283 | PTIADL | R | PTIADL7 | PTIADL6 | PTIADL5 | PTIADL4 | PTIADL3 | PTIADL2 | PTIADL1 | PTIADL0 |
| | | W | | | | | | | | |
| 0x0284 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0285 | DDRADL | R | DDRADL7 | DDRADL6 | DDRADL5 | DDRADL4 | DDRADL3 | DDRADL2 | DDRADL1 | DDRADL0 |
| | | W | | | | | | | | |
| 0x0286 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0287 | PERADL | R | PERADL7 | PERADL6 | PERADL5 | PERADL4 | PERADL3 | PERADL2 | PERADL1 | PERADL0 |
| | | W | | | | | | | | |
| 0x0288 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0289 | PPSADL | R | PPSADL7 | PPSADL6 | PPSADL5 | PPSADL4 | PPSADL3 | PPSADL2 | PPSADL1 | PPSADL0 |
| | | W | | | | | | | | |
| 0x028A–0x028B | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x028C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x028D | PIEADL | R | PIEADL7 | PIEADL6 | PIEADL5 | PIEADL4 | PIEADL3 | PIEADL2 | PIEADL1 | PIEADL0 |
| | | W | | | | | | | | |
| 0x028E | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x028F | PIFADL | R | PIFADL7 | PIFADL6 | PIFADL5 | PIFADL4 | PIFADL3 | PIFADL2 | PIFADL1 | PIFADL0 |
| | | W | | | | | | | | |
| 0x0290–0x0298 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0299 | DIENADL | R | DIENADL7 | DIENADL6 | DIENADL5 | DIENADL4 | DIENADL3 | DIENADL2 | DIENADL1 | DIENADL0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x029A–0x02BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02C0 | PTT | R | PTT7 | PTT6 | PTT5 | PTT4 | PTT3 | PTT2 | PTT1 | PTT0 |
| | | W | | | | | | | | |
| 0x02C1 | PTIT | R | PTIT7 | PTIT6 | PTIT5 | PTIT4 | PTIT3 | PTIT2 | PTIT1 | PTIT0 |
| | | W | | | | | | | | |
| 0x02C2 | DDRT | R | DDRT7 | DDRT6 | DDRT5 | DDRT4 | DDRT3 | DDRT2 | DDRT1 | DDRT0 |
| | | W | | | | | | | | |
| 0x02C3 | PERT | R | PERT7 | PERT6 | PERT5 | PERT4 | PERT3 | PERT2 | PERT1 | PERT0 |
| | | W | | | | | | | | |
| 0x02C4 | PPST | R | PPST7 | PPST6 | PPST5 | PPST4 | PPST3 | PPST2 | PPST1 | PPST0 |
| | | W | | | | | | | | |
| 0x02C5 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02C6 | PIET | R | PIET7 | PIET6 | PIET5 | PIET4 | PIET3 | PIET2 | PIET1 | PIET0 |
| | | W | | | | | | | | |
| 0x02C7 | PIFT | R | PIFT7 | PIFT6 | PIFT5 | PIFT4 | PIFT3 | PIFT2 | PIFT1 | PIFT0 |
| | | W | | | | | | | | |
| 0x02C8–0x02CF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02D0 | PTS | R | PTS7 | PTS6 | PTS5 | PTS4 | PTS3 | PTS2 | PTS1 | PTS0 |
| | | W | | | | | | | | |
| 0x02D1 | PTIS | R | PTIS7 | PTIS6 | PTIS5 | PTIS4 | PTIS3 | PTIS2 | PTIS1 | PTIS0 |
| | | W | | | | | | | | |
| 0x02D2 | DDRS | R | DDRS7 | DDRS6 | DDRS5 | DDRS4 | DDRS3 | DDRS2 | DDRS1 | DDRS0 |
| | | W | | | | | | | | |
| 0x02D3 | PERS | R | PERS7 | PERS6 | PERS5 | PERS4 | PERS3 | PERS2 | PERS1 | PERS0 |
| | | W | | | | | | | | |
| 0x02D4 | PPSS | R | PPSS7 | PPSS6 | PPSS5 | PPSS4 | PPSS3 | PPSS2 | PPSS1 | PPSS0 |
| | | W | | | | | | | | |
| 0x02D5 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02D6 | PIES | R | PIES7 | PIES6 | PIES5 | PIES4 | PIES3 | PIES2 | PIES1 | PIES0 |
| | | W | | | | | | | | |
| 0x02D7 | PIFS | R | PIFS7 | PIFS6 | PIFS5 | PIFS4 | PIFS3 | PIFS2 | PIFS1 | PIFS0 |
| | | W | | | | | | | | |
| 0x02D8–0x02DE | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02DF | WOMS | R | WOMS7 | WOMS6 | WOMS5 | WOMS4 | WOMS3 | WOMS2 | WOMS1 | WOMS0 |
| | | W | | | | | | | | |
| 0x02E0–0x02EF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02F0 | PTP | R | PTP7 | PTP6 | PTP5 | PTP4 | PTP3 | PTP2 | PTP1 | PTP0 |
| | | W | | | | | | | | |
| 0x02F1 | PTIP | R | PTIP7 | PTIP6 | PTIP5 | PTIP4 | PTIP3 | PTIP2 | PTIP1 | PTIP0 |
| | | W | | | | | | | | |
| 0x02F2 | DDRP | R | DDRP7 | DDRP6 | DDRP5 | DDRP4 | DDRP3 | DDRP2 | DDRP1 | DDRP0 |
| | | W | | | | | | | | |
| 0x02F3 | PERP | R | PERP7 | PERP6 | PERP5 | PERP4 | PERP3 | PERP2 | PERP1 | PERP0 |
| | | W | | | | | | | | |
| 0x02F4 | PPSP | R | PPSP7 | PPSP6 | PPSP5 | PPSP4 | PPSP3 | PPSP2 | PPSP1 | PPSP0 |
| | | W | | | | | | | | |
| 0x02F5–0x02FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0300 | PTH | R | PTH7 | PTH6 | PTH5 | PTH4 | PTH3 | PTH2 | PTH1 | PTH0 |
| | | W | | | | | | | | |
| 0x0301 | PTIH | R | PTIH7 | PTIH6 | PTIH5 | PTIH4 | PTIH3 | PTIH2 | PTIH1 | PTIH0 |
| | | W | | | | | | | | |
| 0x0302 | DDRH | R | DDRH7 | DDRH6 | DDRH5 | DDRH4 | DDRH3 | DDRH2 | DDRH1 | DDRH0 |
| | | W | | | | | | | | |
| 0x0303 | PERH | R | PERH7 | PERH6 | PERH5 | PERH4 | PERH3 | PERH2 | PERH1 | PERH0 |
| | | W | | | | | | | | |
| 0x0304 | PPSH | R | PPSH7 | PPSH6 | PPSH5 | PPSH4 | PPSH3 | PPSH2 | PPSH1 | PPSH0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0305–0x030F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0310 | PTJ | R | 0 | 0 | 0 | 0 | PTJ3 | PTJ2 | PTJ1 | PTJ0 |
| | | W | | | | | | | | |
| 0x0311 | PTIJ | R | 0 | 0 | 0 | 0 | PTIJ3 | PTIJ2 | PTIJ1 | PTIJ0 |
| | | W | | | | | | | | |
| 0x0312 | DDRJ | R | 0 | 0 | 0 | 0 | DDRJ3 | DDRJ2 | DDRJ1 | DDRJ0 |
| | | W | | | | | | | | |
| 0x0313 | PERJ | R | 0 | 0 | 0 | 0 | PERJ3 | PERJ2 | PERJ1 | PERJ0 |
| | | W | | | | | | | | |
| 0x0314 | PPSJ | R | 0 | 0 | 0 | 0 | PPSJ3 | PPSJ2 | PPSJ1 | PPSJ0 |
| | | W | | | | | | | | |
| 0x0315–0x031F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0320 | PTG | R | PTG7 | PTG6 | PTG5 | PTG4 | PTG3 | PTG2 | PTG1 | PTG0 |
| | | W | | | | | | | | |
| 0x0321 | PTIG | R | PTIG7 | PTIG6 | PTIG5 | PTIG4 | PTIG3 | PTIG2 | PTIG1 | PTIG0 |
| | | W | | | | | | | | |
| 0x0322 | DDRG | R | DDRG7 | DDRG6 | DDRG5 | DDRG4 | DDRG3 | DDRG2 | DDRG1 | DDRG0 |
| | | W | | | | | | | | |
| 0x0323 | PERG | R | PERG7 | PERG6 | PERG5 | PERG4 | PERG3 | PERG2 | PERG1 | PERG0 |
| | | W | | | | | | | | |
| 0x0324 | PPSG | R | PPSG7 | PPSG6 | PPSG5 | PPSG4 | PPSG3 | PPSG2 | PPSG1 | PPSG0 |
| | | W | | | | | | | | |
| 0x0325–0x032F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0350 | PTU | R | PTU7 | PTU6 | PTU5 | PTU4 | PTU3 | PTU2 | PTU1 | PTU0 |
| | | W | | | | | | | | |
| 0x0351 | PTIU | R | PTIU7 | PTIU6 | PTIU5 | PTIU4 | PTIU3 | PTIU2 | PTIU1 | PTIU0 |
| | | W | | | | | | | | |
| 0x0352 | DDRU | R | DDRU7 | DDRU6 | DDRU5 | DDRU4 | DDRU3 | DDRU2 | DDRU1 | DDRU0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0353 | PERU | R | PERU7 | PERU6 | PERU5 | PERU4 | PERU3 | PERU2 | PERU1 | PERU0 |
| | | W | | | | | | | | |
| 0x0354 | PPSU | R | PPSU7 | PPSU6 | PPSU5 | PPSU4 | PPSU3 | PPSU2 | PPSU1 | PPSU0 |
| | | W | | | | | | | | |
| 0x0355–0x035D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x035E | SRRU | R | SRRU7 | SRRU6 | SRRU5 | SRRU4 | SRRU3 | SRRU2 | SRRU1 | SRRU0 |
| | | W | | | | | | | | |
| 0x035F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0360–0x037F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 2.3.2 Register Descriptions

This section describes the details of all configuration registers.

- If not stated differently, writing to reserved bits has no effect and read returns zero.
- All register read accesses are synchronous to internal clocks.
- All registers can be written at any time, however a specific configuration might not become active. E.g. a pullup device does not become active while the port is used as a push-pull output.
- General-purpose data output availability depends on prioritization; input data registers always reflect the pin status independent of the use.
- Pull-device availability, pull-device polarity, wired-or mode, key-wake up functionality are independent of the prioritization unless noted differently.
- The description of registers PTx, PTIx, DDRx, DIENx, PERx, PPSx, SRRx, WOMx, PIEx and PIFx generically assumes a fully implemented 8-bit register. For availability of individual bits refer to Section 2.3.1, “Register Map“”.

### 2.3.2.1 Module Routing Register 0 (MODRR0)

Address 0x0200 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | C0RR | 0 | 0 | 0 |
| W | | | | | | | | |
| | — | — | — | — | CAN0 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-1. Module Routing Register 0 (MODRR0)**

1. Read: Anytime
   Write: Once in normal, anytime in special mode

.

**Table 2-2. Module Routing Register0 Field Descriptions**

| Field | Description |
|---|---|
| 3<br>C0RR | **Module Routing Register** — CAN0 routing<br><br>1 TXCAN0 on PS5 and RXCAN0 on PS4<br>0 TXCAN0 on PC1 and RXCAN0 on PC0 |

### 2.3.2.2 Module Routing Register 1 (MODRR1)

Address 0x0201 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PWM6RR | PWM4RR | PWM2RR | PWM0RR |
| W | | | | | | | | |
| | — | — | — | — | PWM6 | PWM4 | PWM2 | PWM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-2. Module Routing Register 1 (MODRR1)**

1. Read: Anytime
   Write: Once in normal, anytime in special mode

**Table 2-3. MODRR1 Routing Register Field Descriptions**

| Field | Description |
|---|---|
| 3<br>PWM6RR | **Module Routing Register** — PWM6 routing<br><br>1 PWM6 to PA7<br>0 PWM6 to PP6 |
| 2<br>PWM4RR | **Module Routing Register** — PWM4 routing<br><br>1 PWM4 to PA6<br>0 PWM4 to PP4 |

**Table 2-3. MODRR1 Routing Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>PWM2RR | **Module Routing Register** — PWM2 routing<br><br>1 PWM2 to PA5<br>0 PWM2 to PP2 |
| 0<br>PWM0RR | **Module Routing Register** — PWM0 routing<br><br>1 PWM0 to PA4<br>0 PWM0 to PP0 |

### 2.3.2.3 Module Routing Register 2 (MODRR2)

Address 0x0202 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SCI1RR | IIC0RR | 0 | 0 | T1IC0RR1 | T1IC0RR0 |
| W | | | | | | | | |
| | | | SCI1 | IIC0 | | | TIM1 IC0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-3. Module Routing Register 2 (MODRR2)**

1. Read: Anytime
   Write: Once in normal, anytime in special mode

**Table 2-4. MODRR2 Routing Register Field Descriptions**

| Field | Description |
|---|---|
| 5<br>SCI1RR | **Module Routing Register** — SCI1 routing<br><br>1 TXD1 on PP7; RXD1 on PP5<br>0 TXD1 on PC7; RXD1 on PC6 |
| 4<br>IIC0RR | **Module Routing Register** — IIC0 routing<br><br>1 SCL0 on PA2; SDA0 on PA3<br>0 SCL0 on PS4; SDA0 on PS5 |
| 1-0<br>T1IC0RR1-0 | **Module Routing Register** — TIM1 IC0 routing<br><br>11 TIM1 input capture channel 0 is connected to RXD1<br>10 TIM1 input capture channel 0 is connected to RXD0<br>01 TIM1 input capture channel 0 is connected to RTC's CALCLK<br>00 TIM1 input capture channel 0 is connected to PT0 |

### 2.3.2.4 ECLK Control Register (ECLKCTL)

Address 0x0208 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NECLK | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-4. ECLK Control Register (ECLKCTL)**

1. Read: Anytime
   Write: Anytime

**Table 2-5. ECLKCTL Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>NECLK | **No ECLK** — Disable ECLK output<br><br>This bit controls the availability of a free-running clock on the ECLK pin. This clock has a fixed rate equivalent to the internal bus clock.<br><br>1 ECLK disabled<br>0 ECLK enabled |

### 2.3.2.5 IRQ Control Register (IRQCR)

Address 0x0209 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-5. IRQ Control Register (IRQCR)**

1. Read: Anytime
   Write:
   IRQE: Once in normal mode, anytime in special mode
   IRQEN: Anytime

**Table 2-6. IRQCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IRQE | **IRQ select edge sensitive only** —<br><br>1 $\overline{IRQ}$ pin configured to respond only to falling edges. Falling edges on the $\overline{IRQ}$ pin are detected anytime when IRQE=1 and will be cleared only upon a reset or the servicing of the $\overline{IRQ}$ interrupt.<br>0 $\overline{IRQ}$ configured for low level recognition |
| 6<br>IRQEN | **IRQ enable** —<br><br>1 $\overline{IRQ}$ pin is connected to interrupt logic<br>0 $\overline{IRQ}$ pin is disconnected from interrupt logic |

## 2.3.2.6 PIM Miscellaneous Register (PIMMISC)

Address 0x020A Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CALCLKEN |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-6. PIM Miscellaneous Register (PIMMISC)**

1. Read: Anytime
   Write:Anytime

**Table 2-7. PIM Miscellaneous Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>CALCLKEN | RTC_CAL output Enable — Activate the RTC CALCLK output on PT1<br><br>1 CALCLK output on PT1 enabled<br>0 CALCLK output on PT1 disabled |

### 2.3.2.7 Reserved Register

Address 0x020E Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Reset | x | x | x | x | x | x | x | x |

**Figure 2-7. Reserved Register**

1. Read: Anytime
   Write: Only in special mode

These reserved registers are designed for factory test purposes only and are not intended for general user access. Writing to these registers when in special modes can alter the modules functionality

Address 0x020F Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Reset | x | x | x | x | x | x | x | x |

**Figure 2-8. Reserved Register**

1. Read: Anytime
   Write: Only in special mode

These reserved registers are designed for factory test purposes only and are not intended for general user access. Writing to these registers when in special modes can alter the modules functionality

### 2.3.2.8 Port C Polarity Select Register

Address 0x0248 PPSC Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSx7 | PPSx6 | PPSx5 | PPSx4 | PPSx3 | PPSx2 | PPSx1 | PPSx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-9. Port C Polarity Select Register**

1. Read: Anytime
   Write: Anytime

**Table 2-8. Port C Polarity Select Register Field Descriptions**

| Field | Description |
|---|---|
| 7-1<br>PPSC | **Pull C Polarity Select** — Configure pull device on input pin<br><br>This bits select a pullup or a pulldown device if enabled on the associated port input pin.<br><br>1 pulldown device selected<br>0 pullup device selected |
| 0<br>PPSC | **Port C Pull Polarity Select** — Configure pull device on input pin<br><br>This bit selects a pullup or a pulldown device if enabled on the associated port input pin.<br><br>If MSCAN0 is active and routing to this pin, a pullup device can be activated on the RXCAN0 input; attempting to select a pulldown disables the pull-device.<br><br>1 pulldown device selected<br>0 pullup device selected |

## 2.3.2.9 Port S Polarity Select Register

Address 0x02D4 PPSS — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSx7 | PPSx6 | PPSx5 | PPSx4 | PPSx3 | PPSx2 | PPSx1 | PPSx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-10. Port S Polarity Select Register**

1. Read: Anytime
   Write: Anytime

**Table 2-9. Port S Polarity Select Register Field Descriptions**

| Field | Description |
|---|---|
| 7-5<br>PPSS | **Pull S Polarity Select** — Configure pull device and pin interrupt edge polarity on input pin<br><br>This bits select a pullup or a pulldown device if enabled on the associated port input pin.<br><br>1 pulldown device selected; rising edge selected<br>0 pullup device selected; falling edge selected |
| 4<br>PPSS | **Port S Pull Polarity Select** — Configure pull device and pin interrupt edge polarity on input pin<br><br>This bit selects a pullup or a pulldown device if enabled on the associated port input pin.<br><br>If MSCAN0 is active and routing to this pin, a pullup device can be activated on the RXCAN0 input; attempting to select a pulldown disables the pull-device.<br><br>1 pulldown device selected; rising edge selected<br>0 pullup device selected; falling edge selected |
| 3-0<br>PPSS | **Port S Pull Polarity Select** — Configure pull device and pin interrupt edge polarity on input pin<br><br>This bit selects a pullup or a pulldown device if enabled on the associated port input pin.<br>This bit also selects the polarity of the active interrupt edge.<br><br>1 pulldown device selected; rising edge selected<br>0 pullup device selected; falling edge selected |

## 2.3.2.10 Port Data Register

Address 0x0220 PTA
0x0221 PTB
0x0240 PTC
0x0241 PTD
0x0260 PTE
0x0261 PTF
0x0281 PTADL
0x02C0 PTT
0x02D0 PTS
0x02F0 PTP
0x0300 PTH
0x0310 PTJ
0x0320 PTG
0x0350 PTU

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTx7 | PTx6 | PTx5 | PTx4 | PTx3 | PTx2 | PTx1 | PTx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-11. Port Data Register**

1. Read: Anytime. The data source is depending on the data direction value.
   Write: Anytime

**Table 2-10. Port Data Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTx | **Port — General purpose input/output data**<br><br>This register holds the value driven out to the pin if the pin is used as a general purpose output.<br>When not used with the alternative function (refer to Table 2-1), these pins can be used as general purpose I/O.<br>If the associated data direction bits of these pins are set to 1, a read returns the value of the port register, otherwise the buffered pin input state is read. |

## 2.3.2.11 Port Input Register

Address 0x0222 PTIA
0x0223 PTIB
0x0242 PTIC
0x0243 PTID
0x0262 PTIE
0x0263 PTIF
0x0283 PTIADL
0x02C1 PTIT
0x02D1 PTIS
0x02F1 PTIP
0x0301 PTIH
0x0311 PTIJ
0x0321 PTIG
0x0351 PTIU

Access: User read only[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTIx7 | PTIx6 | PTIx5 | PTIx4 | PTIx3 | PTIx2 | PTIx1 | PTIx0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-12. Port Input Register**

1. Read: Anytime
   Write:Never

**Table 2-11. Port Input Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PTIx | **Port Input** — Data input<br><br>A read always returns the buffered input state of the associated pin. It can be used to detect overload or short circuit conditions on output pins. |

### 2.3.2.12 Data Direction Register

Address 0x0224 DDRA
0x0225 DDRB
0x0244 DDRC
0x0245 DDRD
0x0264 DDRE
0x0265 DDRF
0x0285 DDRADL
0x02C2 DDRT
0x02D2 DDRS
0x02F2 DDRP
0x0302 DDRH
0x0312 DDRJ
0x0322 DDRG
0x0352 DDRU

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDRx7 | DDRx6 | DDRx5 | DDRx4 | DDRx3 | DDRx2 | DDRx1 | DDRx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-13. Data Direction Register**

1. Read: Anytime
   Write: Anytime

**Table 2-12. Data Direction Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DDRx | **Data Direction** — Select general-purpose data direction<br><br>This bit determines whether the pin is a general-purpose input or output.<br><br>1 Associated pin is configured as output<br>0 Associated pin is configured as input |

**NOTE**

Due to internal synchronization circuits, it can take up to two bus clock cycles until the correct value is read on port data and port input registers, when changing the data direction register.

The general-purpose data direction configuration can be overruled by an enabled peripheral function shared on the same pin (Table 2-21). If more then one peripheral function is available and enabled at the same time, the highest ranked module according the predefined priority scheme in Table 2-1 will take precedence on the pin.

### 2.3.2.13 Digital Input Enable Register

Address 0x0299 DIENADL Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DIENx7 | DIENx6 | DIENx5 | DIENx4 | DIENx3 | DIENx2 | DIENx1 | DIENx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-14. Digital Input Enable Register**

1. Read: Anytime
   Write: Anytime

**Table 2-13. Digital Input Enable Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DIENx | Digital Input Enable — Input buffer control<br><br>This bit controls the digital input function. If set to 1 the input buffers are enabled and the pin can be used with the digital function. If the pin is used with an analog function this bit shall be cleared to avoid shoot-through current.<br><br>1 Associated pin is configured as digital input<br>0 Associated pin digital input is disabled |

### 2.3.2.14 Pull Device Enable Register

Address 0x0226 PERA
0x0227 PERB
0x0246 PERC
0x0247 PERD
0x0266 PERE
0x0267 PERF
0x0287 PERADL
0x02C3 PERT
0x02D3 PERS
0x02F3 PERP
0x0303 PERH
0x0313 PERJ
0x0323 PERG
0x0353 PERU

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PERx7 | PERx6 | PERx5 | PERx4 | PERx3 | PERx2 | PERx1 | PERx0 |
| Reset | | | | | | | | |
| Ports B, E: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Ports C, P T, ADL, J, U: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Others: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 2-15. Pull Device Enable Register**

1. Read: Anytime
Write: Anytime

**Table 2-14. Pull Device Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PERx | **Pull Enable** — Activate pull device on input pin<br><br>This bit controls whether a pull device on the associated port input or open-drain output pin is active. If a pin is used as push-pull output this bit has no effect. The polarity is selected by the related polarity select register bit. On open-drain output pins only a pullup device can be enabled.<br><br>1 Pull device enabled<br>0 Pull device disabled |

### 2.3.2.15 Polarity Select Register

Address 0x0228 PPSA
0x0229 PPSB
0x0248 PPSC
0x0249 PPSD
0x0268 PPSE
0x0269 PPSF
0x0289 PPSADL
0x02C4 PPST
0x02D4 PPSS
0x02F4 PPSP
0x0304 PPSH
0x0314 PPSJ
0x0324 PPSG
0x0354 PPSU

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSx7 | PPSx6 | PPSx5 | PPSx4 | PPSx3 | PPSx2 | PPSx1 | PPSx0 |
| Reset | | | | | | | | |
| Ports B, E: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Ports A, D, F, G, H: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Others: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-16. Polarity Select Register**

1. Read: Anytime
   Write: Anytime

**Table 2-15. Polarity Select Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PPSx | **Pull Polarity Select** — Configure pull device and pin interrupt edge polarity on input pin<br><br>This bit selects a pullup or a pulldown device if enabled on the associated port input pin.<br>If a port has interrupt functionality this bit also selects the polarity of the active edge.<br><br>If MSCAN0 is active a pullup device can be activated on the RXCAN0 input; attempting to select a pulldown disables the pull-device.<br><br>1 Pulldown device selected; rising edge selected<br>0 Pullup device selected; falling edge selected |

### 2.3.2.16 Wired-Or Mode Register

Address 0x023E WOMA
0x02DF WOMS

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WOMx7 | WOMx6 | WOMx5 | WOMx4 | WOMx3 | WOMx2 | WOMx1 | WOMx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-17. Wired-Or Mode Register**

1. Read: Anytime
   Write: Anytime

**Table 2-16. Wired-Or Mode Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>WOMx | **Wired-Or Mode** — Enable open-drain output<br><br>This bit configures the output pin as wired-or. If enabled the output is driven active low only (open-drain) while the active high drive is turned off. This allows a multipoint connection of several serial modules. These bits have no influence on pins used as inputs. Enable the IIC0, it will force the corresponding pins to be open drain output.<br><br>1 Output buffers operate as open-drain outputs<br>0 Output buffers operate as push-pull outputs |

### 2.3.2.17 Port Interrupt Enable Register

Address 0x02C6 PIET
0x02D6 PIES
0x028D PIEADL

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIEx7 | PIEx6 | PIEx5 | PIEx4 | PIEx3 | PIEx2 | PIEx1 | PIEx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-18. Port Interrupt Enable Register**

1. Read: Anytime
   Write: Anytime

**Table 2-17. Port Interrupt Enable Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIEx | **Port Interrupt Enable** — Activate pin interrupt<br><br>This bit enables or disables the edge sensitive pin interrupt on the associated pin. An interrupt can be generated if the pin is operating in input or output mode when in use with the general-purpose or related peripheral function.<br><br>1 Interrupt is enabled<br>0 Interrupt is disabled (interrupt flag masked) |

### 2.3.2.18 Port Interrupt Flag Register

Address 0x02C7 PIFT
0x02D7 PIFS
0x028F PIFADL

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PIFx7 | PIFx6 | PIFx5 | PIFx4 | PIFx3 | PIFx2 | PIFx1 | PIFx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-19. Port Interrupt Flag Register**

1. Read: Anytime
   Write: Anytime

**Table 2-18. Port Interrupt Flag Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>PIFx | **Port Interrupt Flag** — Signal pin event<br><br>The flag asserts after a valid active edge was detected on the related pin (see Section 2.4.4, “Pin interrupts and Wakeup“”). This can be a rising or a falling edge based on the state of the polarity select register.<br><br>Writing a logic “1” to the corresponding bit field clears the flag.<br><br>1 Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set)<br>0 No active edge occurred |

### 2.3.2.19 Port Slew Rate Register

Address 0x035E SRRU

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SRRx7 | SRRx6 | SRRx5 | SRRx4 | SRRx3 | SRRx2 | SRRx1 | SRRx0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-20. Port Slew Rate Register**

1. Read: Anytime
   Write: Anytime

**Table 2-19. Port Interrupt Flag Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>SRRx | **Port Slew Rate**— Slew Rate control[1]<br><br>1 Enable the slew rate control and disable the digital input buffer<br>0 Disable the slew rate control and enable the digital input buffer |

1. When change SRRx from non-zero value to zero value or vice versa, It will need to wait about 300 nanoseconds delay before the slew rate control to be real function as setting. When enter STOP, to save the power, the slew rate control will be force to off state. After wakeup from STOP, it will also need to wait about 300 nanoseconds before slew rate control to be function as setting. To enable the TIM0 ICx functions which shared with motor pads, the corresponding SRRx bit should be set to zero.

### 2.3.2.20 PIM Reserved Register

Address (any reserved) Access: User read[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 2-21. PIM Reserved Register**

1. Read: Always reads 0x00
   Write: Unimplemented

## 2.4 Functional Description

### 2.4.1 General

Each pin except BKGD can act as general-purpose I/O. In addition each pin can act as an output or input of a peripheral module.

### 2.4.2 Registers

Table 2-20 lists the configuration registers which are available on each port. These registers except the pin input registers can be written at any time, however a specific configuration might not become active.

For example selecting a pullup device: This device does not become active while the port is used as a push-pull output.

**Table 2-20. Register availability per port[1]**

| Port | Data | Input | Data Direction | Pull Enable | Polarity Select | Wired-Or Mode | Slew Rate Enable | Interrupt Enable | Interrupt Flag |
|---|---|---|---|---|---|---|---|---|---|
| A | yes | yes | yes | yes | yes | yes[2] | - | - | - |
| B | yes | yes | yes | yes | yes | - | - | - | - |
| C | yes | yes | yes | yes | yes | - | - | - | - |
| D | yes | yes | yes | yes | yes | - | - | - | - |
| E | yes | yes | yes | yes | yes | - | - | - | - |
| F | yes | yes | yes | yes | yes | - | - | - | - |
| J | yes | yes | yes | yes | yes | - | - | - | - |
| G | yes | yes | yes | yes | yes | - | - | - | - |
| H | yes | yes | yes | yes | yes | - | - | - | - |
| P | yes | yes | yes | yes | yes | - | - | - | - |

**Table 2-20. Register availability per port[1]**

| Port | Data | Input | Data Direction | Pull Enable | Polarity Select | Wired-Or Mode | Slew Rate Enable | Interrupt Enable | Interrupt Flag |
|---|---|---|---|---|---|---|---|---|---|
| S | yes | yes | yes | yes | yes | yes | - | yes | yes |
| T | yes | yes | yes | yes | yes | - | - | yes | yes |
| AD | yes | yes | yes | yes | yes | - | - | yes | yes |
| U | yes | yes | yes | yes | yes | - | yes | - | - |

1. Each cell represents one register with individual configuration bits

2. Only PA3/PA2

Table 2-21 shows the effect of enabled peripheral features on I/O state and enabled pull devices.

**Table 2-21. Effect of Enabled Features**

| Enabled Feature | Related Pin(s) | Effect on I/O state | Effect on enabled pull device |
|---|---|---|---|
| CPMU OSC | EXTAL, XTAL | CPMU takes control | Forced off |
| 32K OSC | 32K_EXTAL, 32K_XTAL | OSC takes control if CLKSRC in Section 18.4.2, “RTC Control Register 2 (RTCCTL2)“ is set | Forced off |
| LCD | FP[39:0], BP[3:0] | LCD takes control | Forced off |
| TIMx | OCx | Forced output | Forced off |
| | ICx | None (DDR maintains control) | None (PER/PPS maintain control) |
| SPIx | MISO, MOSI, SCK, $\overline{SS}$ | Controlled input/output | Forced off if output |
| SCIx | TXD | Forced output | Forced off |
| | RXD | Forced input | None (PER/PPS maintain control) |
| CANx | TXCAN | Forced output | Forced off |
| | RXCAN | Forced input | Pulldown forced off |
| IICx | SCL, SDA | Controlled input/output | Forced off if output |
| S12ZDBG | PDO, PDOCLK | Forced output | Forced off |
| SSGx | SGA, SGT | Forced output | Forced off |
| PWM channel | PWMx | Forced output | Forced off |
| MC | MxCxM, MxCxP | Forced output | Forced off |
| SSDx | MxCOSM, MxCOSP, MxSINM, MxSINP | Controlled input/output | Forced off if output |
| API | API_EXTCLK | Forced output | Forced off |
| ADCx | ANx | None (DDR maintains control[1]) | None (PER/PPS maintain control) |

1. To use the digital input function the related bit in Digital Input Enable Register (DIENADx) must be set to logic level “1”. To use the digital input function the related bit in Slew Rate Register (SRRx) must be set to logic level “0”.

### 2.4.2.1 Data register (PTx)

This register holds the value driven out to the pin if the pin is used as a general-purpose I/O.

Writing to this register has only an effect on the pin if the pin is used as general-purpose output. When reading this address, the synchronized state of the pin is returned if the associated data direction register bits are configured as input.

If the data direction register bits are configured as output, the contents of the data register is returned. This is independent of any other configuration (Figure 2-22).

### 2.4.2.2 Input register (PTIx)

This register is read-only and always returns the synchronized state of the pin (Figure 2-22).

### 2.4.2.3 Data direction register (DDRx)

This register defines whether the pin is used as an general-purpose input or an output.

If a peripheral module controls the pin the contents of the data direction register is ignored (Figure 2-22).

Independent of the pin usage with a peripheral module this register determines the source of data when reading the associated data register address (Section 2.4.2.1, “Data register (PTx)“”).

**NOTE**

Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on port data or port input registers, when changing the data direction register.

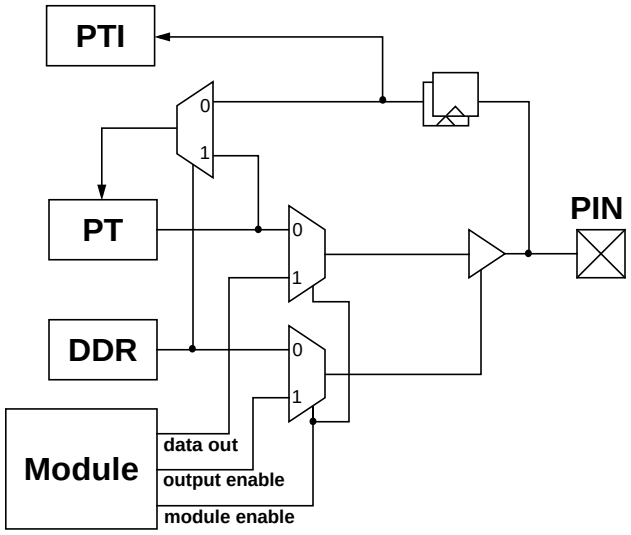



**Figure 2-22. Illustration of I/O pin functionality**

#### 2.4.2.4 Pull device enable register (PERx)

This register turns on a pullup or pulldown device on the related pins determined by the associated polarity select register (Section 2.4.2.5, “Polarity select register (PPSx)“”).

The pull device becomes active only if the pin is used as an input or as a wired-or output. Some peripheral module only allow certain configurations of pull devices to become active. Refer to the respective bit descriptions.

#### 2.4.2.5 Polarity select register (PPSx)

This register selects either a pullup or pulldown device if enabled.

It becomes only active if the pin is used as an input. A pullup device can be activated if the pin is used as a wired-or output.

On pins with interrupt functionality this register selects the active edge polarity of the input signal.

#### 2.4.2.6 Wired-or mode register (WOMx)

If the pin is used as an output this register turns off the active-high drive. This allows wired-or type connections of outputs.

#### 2.4.2.7 Interrupt enable register (PIEx)

If the pin is used as an interrupt input this register serves as a mask to the interrupt flag to enable/disable the interrupt.

#### 2.4.2.8 Interrupt flag register (PIFx)

If the pin is used as an interrupt input this register holds the interrupt flag after a valid pin event.

#### 2.4.2.9 Digital input enable register (DIENADx)

This register controls the digital input buffer. If DIENADx is set to logic level “1”, then it will enable the digital input buffer.

#### 2.4.2.10 Slew rate register (SRRx)

This register selects the slew rate function on the motor pad. It also controls the digital input buffer. If SRRx is set to logic level “1”, then it will disable the digital input buffer.

#### 2.4.2.11 Module routing register (MODRRx)

Routing registers allow software re-configuration of specific peripheral inputs and outputs:

- MODRR0 supports CAN0 rerouting
- MODRR1 supports PWM channel rerouting
- MODRR2 supports TIM1IC0, IIC0 and SCI1 rerouting

## 2.4.3 Interrupts

This section describes the interrupts generated by the PIM and their individual sources. Vector addresses and interrupt priorities are defined at MCU level.

**Table 2-22. PIM Interrupt Sources**

| Module Interrupt Sources | Local Enable |
|---|---|
| $\overline{\text{XIRQ}}$ | None |
| $\overline{\text{IRQ}}$ | IRQCR[IRQEN] |
| Port S pin interrupt | PIES[PIES7-PIES0] |
| Port T pin interrupt | PIET[PIET7-PIET0] |
| Port AD pin interrupt | PIEADL[PIEADL7-PIEADL0] |

### 2.4.3.1 XIRQ, IRQ Interrupts

The $\overline{\text{XIRQ}}$ pin allows requesting non-maskable interrupts after reset initialization. During reset, the X bit in the condition code register is set and any interrupts are masked until software enables them.

The $\overline{\text{IRQ}}$ pin allows requesting asynchronous interrupts. The interrupt input is disabled out of reset. To enable the interrupt the IRQCR[IRQEN] bit must be set and the I bit cleared in the condition code register. The interrupt can be configured for level-sensitive or falling-edge-sensitive triggering. If IRQCR[IRQEN] is cleared while an interrupt is pending, the request will deassert.

Both interrupts are capable to wake-up the device from stop mode. Means for glitch filtering are not provided on these pins.

## 2.4.4 Pin interrupts and Wakeup

Ports S, T and AD offer pin interrupt and key-wakeup capability. The related interrupt enable (PIE) as well as the sensitivity to rising or falling edges (PPS) can be individually configured on per-pin basis. All bits/pins in a port share the same interrupt vector. Interrupts can be used with the pins configured as inputs or outputs.

An interrupt is generated when a bit in the port interrupt flag (PIF) and its corresponding port interrupt enable (PIE) are both set. The pin interrupt feature is also capable to wake up the CPU when it is in stop or wait mode(key-wakeup).

A digital filter on each pin prevents short pulses from generating an interrupt. A valid edge on an input is detected if 4 consecutive samples of a passive level are followed by 4 consecutive samples of an active level. Else the sampling logic is restarted.

In run and wait mode the filters are continuously clocked by the bus clock. Pulses with a duration of $t_{PULSE} < n_{P_MASK}/f_{bus}$ are assuredly filtered out while pulses with a duration of $t_{PULSE} > n_{P_PASS}/f_{bus}$ guarantee a pin interrupt.

In stop mode the filter clock is generated by an RC-oscillator. The minimum pulse length varies over process conditions, temperature and voltage(Figure 2-23). Pulses with a duration of $t_{PULSE} < t_{P_MASK}$ are assuredly filtered out while pulses with a duration of $t_{PULSE} > t_{P_PASS}$ guarantee a wakeup event.

Please refer to the appendix table “Pin Interrupt Characteristics” for pulse length limits.

To maximize current saving the RC oscillator is active only if the following condition is true on any individual pin:

Sample count <= 4 (at active or passive level) and interrupt flag not set (PIF[x]=0).

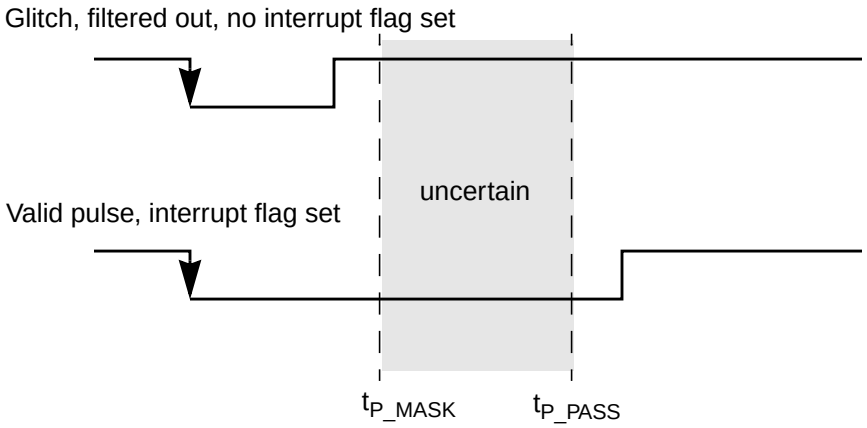



**Figure 2-23. Interrupt Glitch Filter (here: active low level selected)**

## 2.5 Initialization and Application Information

### 2.5.1 Port Data and Data Direction Register writes

It is not recommended to write PORTx/PTx and DDRx in a word access. When changing the register pins from inputs to outputs, the data may have extra transitions during the write access. Initialize the port data register before enabling the outputs.

### 2.5.2 SCI0,1 Baud Rate Detection

The baud rate for SCI0 and SCI1 is achieved by using a timer channel to measure the data rate on the RXD pin.

1. Establish the link:
   — For SCI0: Set [T1IC0RR1:T1IC0RR0]=0b10 to disconnect IOC0 from TIM1 input capture channel 0 and reroute the timer input from the RXD0 of SCI0.
   — For SCI1: Set [T1IC0RR1:T1IC0RR0]=0b11 to disconnect IOC0 from TIM1 input capture channel 0 and reroute the timer input from the RXD1 of SCI1.
2. Determine pulse width of incoming data: Configure TIM1 IC0 to measure time between incoming signal edges.

### 2.5.3 RTC on chip calibration

The on chip RTC calibration used the TIM1 IC0 and IC1 channel.

1. Establish the link:
   — Set the RTC configuration to generate the expect CALCLK frequency
   — Set [T1IC0RR1:T1IC0RR0]=0b10 to disconnect IOC0 from TIM1 input capture channel 0 and reroute the RTC CALCLK to IC0.
   — Input a standard clock on RTC_CAL pin
2. Determine pulse width of incoming data: Configure TIM1 IC0 and IC1 to measure time between incoming signal edges. Need to pay attention to timer overflow also.
3. Compare the data between IC0 and IC1 to get the RTC clock frequency value. Suppose the RTC_CAL measure value is B, and CALCLK measure value is A. Then the CALCLK frequency will be between $f_{RTC_CAL}$ * (B-2) / (A+2) and $f_{RTC_CAL}$ * (B+2) / (A-2). Depend on the calibration, user need to select the RTC_CAL, CALCLK and TIM1 frequency.

### 2.5.4 RTC off chip calibration

The off chip RTC calibration, user need to set the CALCLKEN bit in the PIMMIC register to enable the CALCLK out on RTC_CAL pin. Base on the external requirement, user can set the RTCMOD and CALS bit in RTCCTL3 registers to get the expected CALCLK output on RTC_CAL pin.

# Chapter 3
# Memory Mapping Control (S12ZMMCV1)

**Table 3-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.00 | 6 Mar 2012 | | Added footnotes to Table 3-8 |
| V01.01 | 5 Apr 2012 | | Fixed wording |
| V01.02 | 16 May 2012 | | Fixed wording |
| V01.03 | 27 Jul 2012 | | Corrected Table 3-8 |
| V01.04 | 27 Jul 2012 | | Added feature tags |
| V01.05 | 6 Aug 2012 | | Fixed wording |
| V01.06 | 12 Feb 2013 | Figure 3-8<br>3.3.2.2/3-112 | • Changed “KByte:to “KB”<br>• Corrected the description of the MMCECH/L register<br>• |
| V01.07 | 3 May 2013 | | • Fixed typos<br>• Removed PTU references |

## 3.1 Introduction

The S12ZMMC module controls the access to all internal memories and peripherals for the S12ZCPU, and the S12ZBDC module. It also provides dirct memory access for the ADC module. The S12ZMMC determines the address mapping of the on-chip resources, regulates access priorities and enforces memory protection. Figure 3-1 shows a block diagram of the S12ZMMC module.

### 3.1.1 Glossary

**Table 3-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| MCU | Microcontroller Unit |
| CPU | S12Z Central Processing Unit |
| BDC | S12Z Background Debug Controller |
| ADC | Analog-to-Digital Converter |
| unmapped address range | Address space that is not assigned to a memory |
| reserved address range | Address space that is reserved for future use cases |
| illegal access | Memory access, that is not supported or prohibited by the S12ZMMC, e.g. a data store to NVM |
| access violation | Either an illegal access or an uncorrectable ECC error |
| byte | 8-bit data |
| word | 16-bit data |

### 3.1.2 Overview

The S12ZMMC provides access to on-chip memories and peripherals for the S12ZCPU, the S12ZBDC, and the ADC. It arbitrates memory accesses and determines all of the MCU memory maps. Furthermore, the S12ZMMC is responsible for selecting the MCUs functional mode.

### 3.1.3 Features

- S12ZMMC mode operation control
- Memory mapping for S12ZCPU, S12ZBDC, and ADC
  - — Maps peripherals and memories into a 16 MByte address space for the S12ZCPU, the S12ZBDC, and the ADC
  - — Handles simultaneous accesses to different on-chip resources (NVM, RAM, and peripherals)
- Access violation detection and logging
  - — Triggers S12ZCPU machine exceptions upon detection of illegal memory accesses and uncorrectable ECC errors
  - — Logs the state of the S12ZCPU and the cause of the access error

### 3.1.4 Modes of Operation

#### 3.1.4.1 Chip configuration modes

The S12ZMMC determines the chip configuration mode of the device. It captures the state of the MODC pin at reset and provides the ability to switch from special-single chip mode to normal single chip-mode.

### 3.1.4.2 Power modes

The S12ZMMC module is only active in run and wait mode.There is no bus activity in stop mode.

## 3.1.5 Block Diagram

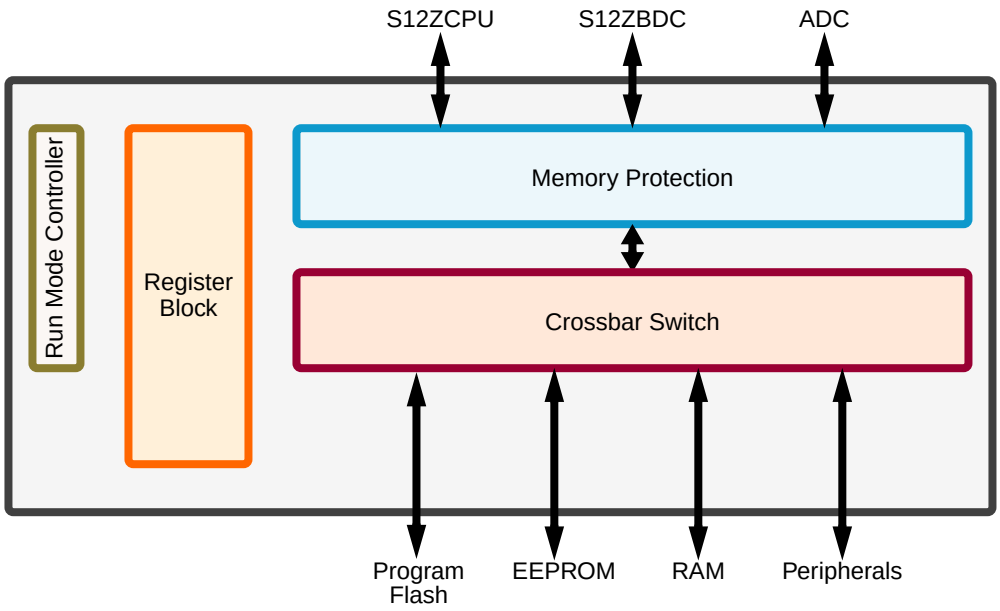



**Figure 3-1. S12ZMMC Block Diagram**

# 3.2 External Signal Description

The S12ZMMC uses two external pins to determine the devices operating mode: RESET and MODC (Table 3-3)
See device overview for the mapping of these signals to device pins.

**Table 3-3. External System Pins Associated With S12ZMMC**

| Pin Name | Description |
|---|---|
| $\overline{\text{RESET}}$ | External reset signal. The $\overline{\text{RESET}}$ signal is active low. |
| MODC | This input is captured in bit MODC of the MODE register when the external $\overline{\text{RESET}}$ pin deasserts. |

# 3.3 Memory Map and Register Definition

## 3.3.1 Memory Map

A summary of the registers associated with the MMC block is shown in Figure 3-2. Detailed descriptions of the registers and bits are given in the subsections that follow.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0070 | MODE | R | MODC | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0071-0x007F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0080 | MMCECH | R | ITR[3:0] | | | | TGT[3:0] | | | |
| | | W | | | | | | | | |
| 0x0081 | MMCECL | R | ACC[3:0] | | | | ERR[3:0] | | | |
| | | W | | | | | | | | |
| 0x0082 | MMCCCRH | R | CPUU | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0083 | MMCCCRL | R | 0 | CPUX | 0 | CPUI | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0084 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0085 | MMCPCH | R | CPUPC[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0086 | MMCPCM | R | CPUPC[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0087 | MMCPCL | R | CPUPC[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0088-0x00FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 3-2. S12ZMMC Register Summary**

## 3.3.2 Register Descriptions

This section consists of the S12ZMMC control and status register descriptions in address order.

### 3.3.2.1 Mode Register (MODE)

Address: 0x0070

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MODC | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | MODC[1] | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. External signal (see Table 3-3).

= Unimplemented or Reserved

**Figure 3-3. Mode Register (MODE)**

Read: Anytime.

Write: Only if a transition is allowed (see Figure 3-4).

The MODE register determines the operating mode of the MCU.

**Table 3-4. MODE Field Descriptions**

| Field | Description |
|---|---|
| 7<br>MODC | **Mode Select Bit** — This bit determines the current operating mode of the MCU. Its reset value is captured from the MODC pin at the rising edge of the $\overline{\text{RESET}}$ pin. Figure 3-4 illustrates the only valid mode transition from special single-chip mode to normal single chip mode. |

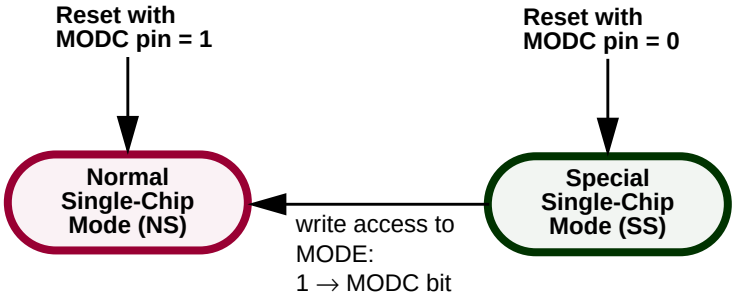



**Figure 3-4. Mode Transition Diagram**

### 3.3.2.2 Error Code Register (MMCECH, MMCECL)

Address: 0x0080 (MMCECH)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ITR[3:0] | | | | TGT[3:0] | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0x0081 (MMCECL)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ACC[3:0] | | | | ERR[3:0] | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3-5. Error Code Register (MMCEC)**

Read: Anytime

Write: Write of 0xFFFF to MMCECH:MMCECL resets both registers to 0x0000

**Table 3-5. MMCECH and MMCECL Field Descriptions**

| Field | Description |
|---|---|
| 7-4 (MMCECH)<br>ITR[3:0] | **Initiator Field** — The ITR[3:0] bits capture the initiator which caused the access violation. The initiator is captured in form of a 4 bit value which is assigned as follows:<br>0: none (no error condition detected)<br>1: S12ZCPU<br>2: reserved<br>3: ADC<br>4-15: reserved |
| 3-0 (MMCECH)<br>TGT[3:0] | **Target Field** — The TGT[3:0] bits capture the target of the faulty access. The target is captured in form of a 4 bit value which is assigned as follows:<br>0: none<br>1: register space<br>2: RAM<br>3: EEPROM<br>4: program flash<br>5: IFR<br>6-15: reserved |

| Field | Description |
|---|---|
| 7-4 (MMCECL) ACC[3:0] | **Access Type Field** — The ACC[3:0] bits capture the type of memory access, which caused the access violation. The access type is captured in form of a 4 bit value which is assigned as follows:<br>0: none (no error condition detected)<br>1: opcode fetch<br>2: vector fetch<br>3: data load<br>4: data store<br>5-15: reserved |
| 3-0 (MMCECL) ERR[3:0] | **Error Type Field** — The EC[3:0] bits capture the type of the access violation. The type is captured in form of a 4 bit value which is assigned as follows:<br>0: none (no error condition detected)<br>1: access to an illegal address<br>2: uncorrectable ECC error<br>3-15: reserved |

The MMCEC register captures debug information about access violations. It is set to a non-zero value if a S12ZCPU access violation or an uncorrectable ECC error has occurred. At the same time this register is set to a non-zero value, access information is captured in the MMCPCn and MMCCCRn registers. The MMCECn, the MMCPCn and the MMCCCRn registers are not updated if the MMCECn registers contain a non-zero value. The MMCECn registers are cleared by writing the value 0xFFFF.

### 3.3.2.3 Captured S12ZCPU Condition Code Register (MMCCCRH, MMCCCRL)

Address: 0x0082 (MMCCCRH)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CPUU | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0x0083 (MMCCCRL)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | CPUX | 0 | CPUI | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3-6. Captured S12ZCPU Condition Code Register (MMCCCRH, MMCCCRL)**

Read: Anytime

Write: Never

**Table 3-6. MMCCCRH and MMCCCRL Field Descriptions**

| Field | Description |
|---|---|
| 7 (MMCCCRH) CPUU | **S12ZCPU User State Flag** — This bit shows the state of the user/supervisor mode bit in the S12ZCPU's CCR at the time the access violation has occurred. The S12ZCPU user state flag is read-only; it will be automatically updated when the next error condition is flagged through the MMCEC register. This bit is undefined if the error code registers (MMCECn) are cleared. |
| 6 (MMCCCRL) CPUX | **S12ZCPU X-Interrupt Mask**— This bit shows the state of the X-interrupt mask in the S12ZCPU's CCR at the time the access violation has occurred. The S12ZCPU X-interrupt mask is read-only; it will be automatically updated when the next error condition is flagged through the MMCEC register. This bit is undefined if the error code registers (MMCECn) are cleared. |
| 4 (MMCCCRL) CPUI | **S12ZCPU I-Interrupt Mask**— This bit shows the state of the I-interrupt mask in the CPU's CCR at the time the access violation has occurred. The S12ZCPU I-interrupt mask is read-only; it will be automatically updated when the next error condition is flagged through the MMCEC register. This bit is undefined if the error code registers (MMCECn) are cleared. |

### 3.3.2.4 Captured S12ZCPU Program Counter (MMCPCH, MMCPCM, MMCPCL)

Address: 0x0085 (MMCPCH)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CPUPC[23:16] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0x0086 (MMCPCM)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CPUPC[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: 0x0087 (MMCPCL)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CPUPC[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3-7. Captured S12ZCPU Program Counter (MMCPCH, MMCPCM, MMCPCL)**

Read: Anytime

Write: Never

**Table 3-7. MMCPCH, MMCPCM, and MMCPCL Field Descriptions**

| Field | Description |
|---|---|
| 7–0 (MMCPCH)<br>7–0 (MMCPCM)<br>7–0 (MMCPCL)<br>CPUPC[23:0] | **S12ZCPU Program Counter Value**— The CPUPC[23:0] stores the CPU's program counter value at the time the access violation occurred. CPUPC[23:0] always points to the instruction which triggered the violation. These bits are undefined if the error code registers (MMCECn) are cleared. |

## 3.4 Functional Description

This section provides a complete functional description of the S12ZMMC module.

### 3.4.1 Global Memory Map

The S12ZMMC maps all on-chip resources into an 16MB address space, the global memory map. The exact resource mapping is shown in Figure 3-8. The global address space is used by the S12ZCPU, ADC, and the S12ZBDC module.

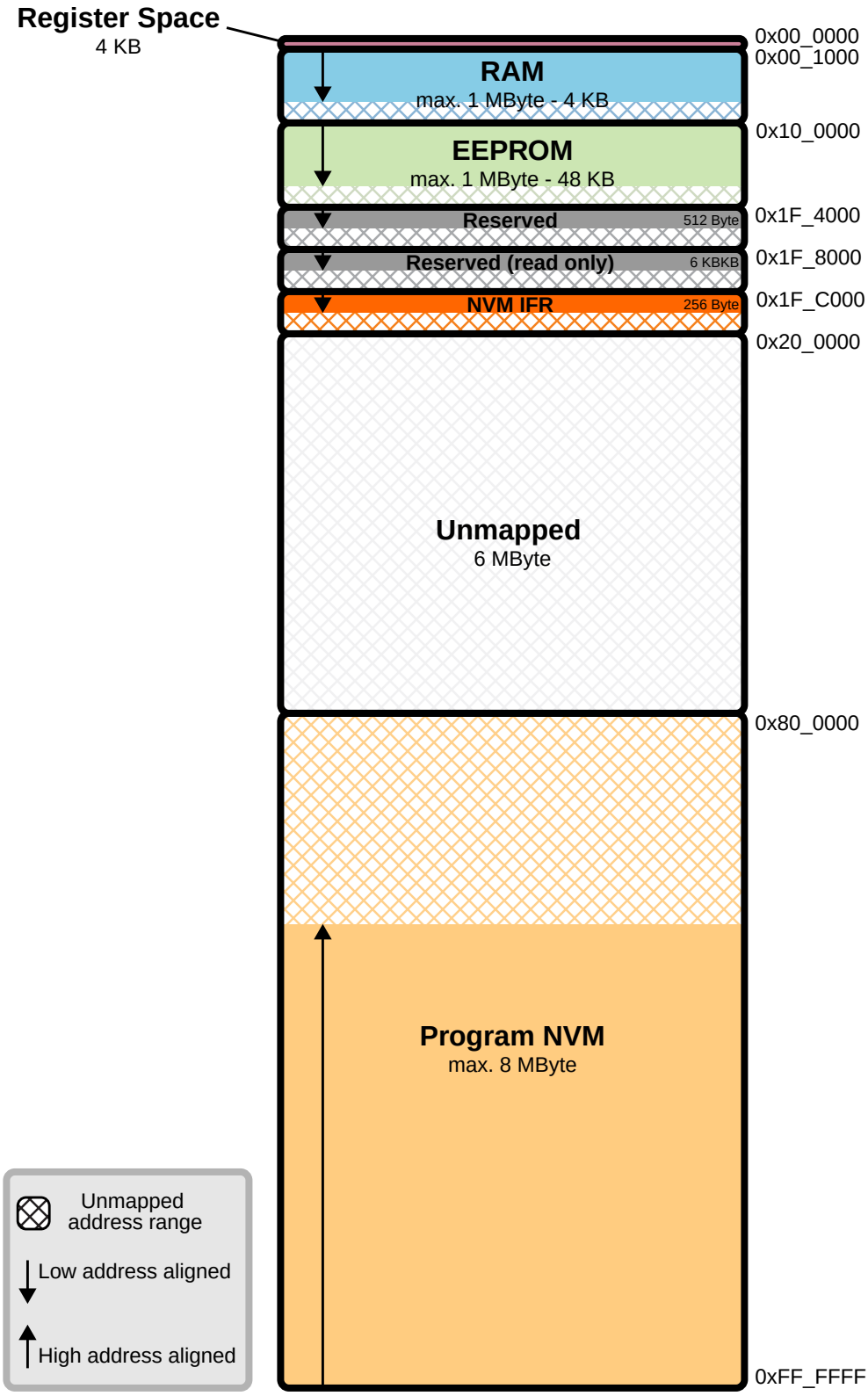



**Figure 3-8. Global Memory Map**

### 3.4.2 Illegal Accesses

The S12ZMMC module monitors all memory traffic for illegal accesses. See Table 3-8 for a complete list of all illegal accesses.

**Table 3-8. Illegal memory accesses**

| | | S12ZCPU | S12ZBDC | ADC |
|---|---|---|---|---|
| Register space | Read access | ok | ok | illegal access |
| | Write access | ok | ok | illegal access |
| | Code execution | illegal access | | |
| RAM | Read access | ok | ok | ok |
| | Write access | ok | ok | ok |
| | Code execution | ok | | |
| EEPROM | Read access | ok[1] | ok[1] | ok[1] |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | ok[1] | | |
| Reserved Space | Read access | ok | ok | illegal access |
| | Write access | only permitted in SS mode | ok | illegal access |
| | Code execution | illegal access | | |
| Reserved Read-only Space | Read access | ok | ok | illegal access |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | illegal access | | |
| NVM IFR | Read access | ok[1] | ok[1] | illegal access |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | illegal access | | |
| Program NVM | Read access | ok[1] | ok[1] | ok[1] |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | ok[1] | | |
| Unmapped Space | Read access | illegal access | illegal access | illegal access |
| | Write access | illegal access | illegal access | illegal access |
| | Code execution | illegal access | | |

1 Unsupported NVM accesses during NVM command execution (“collisions”), are treated as illegal accesses.

Illegal accesses are reported in several ways:

- All illegal accesses performed by the S12ZCPU trigger machine exceptions.
- All illegal accesses performed through the S12ZBDC interface, are captured in the ILLACC bit of the BDCCSRL register.

- All illegal accesses performed by the ADC module trigger error interrupts. See ADC section for details.

**NOTE**

Illegal accesses caused by S12ZCPU opcode prefetches will also trigger machine exceptions, even if those opcodes might not be executed in the program flow. To avoid these machine exceptions, S12ZCPU instructions must not be executed from the last (high addresses) 8 bytes of RAM, EEPROM, and Flash.

### 3.4.3 Uncorrectable ECC Faults

RAM and flash use error correction codes (ECC) to detect and correct memory corruption. Each uncorrectable memory corruption, which is detected during a S12ZCPU or ADC access triggers a machine exception. Uncorrectable memory corruptions which are detected during a S12ZBDC access, are captured in the RAMWF or the RDINV bit of the BDCCSRL register.

# Chapter 4
# Interrupt (S12ZINTV0)

**Table 4-1. Revision History**

| Version Number | Revision Date | Effective Date | Description of Changes |
|---|---|---|---|
| V00.01 | 17 Apr 2009 | all | Initial version based on S12XINT V2.06 |
| V00.02 | 14 Jul 2009 | all | Reduce RESET vectors from three to one. |
| V00.03 | 05 Oct 2009 | all | Removed dedicated ECC machine exception vector and marked vector-table entry “reserved for future use”.<br>Added a second illegal op-code vector (to distinguish between SPARE and TRAP). |
| V00.04 | 04 Jun 2010 | all | Fixed remaining descriptions of RESET vectors.<br>Split non-maskable hardware interrupts into XGATE software error and machine exception requests.<br>Replaced mentions of CCR (old name from S12X) with CCW (new name). |
| V00.05 | 12 Jan 2011 | all | Corrected wrong IRQ vector address in some descriptions. |
| V00.06 | 22 Mar 2011 | all | Added vectors for RAM ECC and NVM ECC machine exceptions. And moved position to 1E0..1E8.<br>Moved XGATE error interrupt to vector 1DC.<br>Remaining vectors accordingly.<br>Removed illegal address reset as a potential reset source. |
| V00.07 | 15 Apr 2011 | all | Removed illegal address reset as a potential reset source from Exception vector table as well. Added the other possible reset sources to the table.<br>Changed register addresses according to S12Z platform definition. |
| V00.08 | 02 May 2011 | all | Reduced machine exception vectors to one.<br>Removed XGATE error interrupt.<br>Moved Spurious interrupt vector to 1DC.<br>Moved vector base address to 010 to make room for NVM non-volatile registers. |
| V00.09 | 12 Aug 2011 | all | Added: Machine exceptions can cause wake-up from STOP or WAIT |
| V00.10 | 21 Feb 2012 | all | Corrected reset value for INT_CFADDR register |
| V00.11 | 02 Jul 2012 | all | Removed references and functions related to XGATE |
| V00.12 | 22 May 2013 | all | added footnote about availability of “Wake-up from STOP or WAIT by XIRQ with X bit set” feature |

## 4.1 Introduction

The INT module decodes the priority of all system exception requests and provides the applicable vector for processing the exception to the CPU. The INT module supports:

- I-bit and X-bit maskable interrupt requests
- One non-maskable unimplemented page1 op-code trap

- One non-maskable unimplemented page2 op-code trap
- One non-maskable software interrupt (SWI)
- One non-maskable system call interrupt (SYS)
- One non-maskable machine exception vector request
- One spurious interrupt vector request
- One system reset vector request

Each of the I-bit maskable interrupt requests can be assigned to one of seven priority levels supporting a flexible priority scheme. The priority scheme can be used to implement nested interrupt capability where interrupts from a lower level are automatically blocked if a higher level interrupt is being processed.

### 4.1.1 Glossary

The following terms and abbreviations are used in the document.

**Table 4-2. Terminology**

| Term | Meaning |
|---|---|
| CCW | Condition Code Register (in the S12Z CPU) |
| DMA | Direct Memory Access |
| INT | Interrupt |
| IPL | Interrupt Processing Level |
| ISR | Interrupt Service Routine |
| MCU | Micro-Controller Unit |
| $\overline{\text{IRQ}}$ | refers to the interrupt request associated with the $\overline{\text{IRQ}}$ pin |
| $\overline{\text{XIRQ}}$ | refers to the interrupt request associated with the $\overline{\text{XIRQ}}$ pin |

### 4.1.2 Features

- Interrupt vector base register (IVBR)
- One system reset vector (at address 0xFFFFFC).
- One non-maskable unimplemented page1 op-code trap (SPARE) vector (at address vector base[1] + 0x0001F8).
- One non-maskable unimplemented page2 op-code trap (TRAP) vector (at address vector base[1] + 0x0001F4).
- One non-maskable software interrupt request (SWI) vector (at address vector base[1] + 0x0001F0).
- One non-maskable system call interrupt request (SYS) vector (at address vector base[1] + 0x00001EC).
- One non-maskable machine exception vector request (at address vector base[1] + 0x0001E8).
- One spurious interrupt vector (at address vector base[1] + 0x0001DC).
- One X-bit maskable interrupt vector request associated with $\overline{\text{XIRQ}}$ (at address vector base[1] + 0x0001D8).

1. The vector base is a 24-bit address which is accumulated from the contents of the interrupt vector base register (IVBR, used as the upper 15 bits of the address) and 0x000 (used as the lower 9 bits of the address).

- One I-bit maskable interrupt vector request associated with $\overline{\text{IRQ}}$ (at address vector base[1] + 0x0001D4).
- up to 113 additional I-bit maskable interrupt vector requests (at addresses vector base[1] + 0x000010 .. vector base + 0x0001D0).
- Each I-bit maskable interrupt request has a configurable priority level.
- I-bit maskable interrupts can be nested, depending on their priority levels.
- Wakes up the system from stop or wait mode when an appropriate interrupt request occurs or whenever $\overline{\text{XIRQ}}$ is asserted, even if X interrupt is masked.

### 4.1.3 Modes of Operation

- Run mode

  This is the basic mode of operation.
- Wait mode

  In wait mode, the INT module is capable of waking up the CPU if an eligible CPU exception occurs. Please refer to Section 4.5.3, “Wake Up from Stop or Wait Mode” for details.
- Stop Mode

  In stop mode, the INT module is capable of waking up the CPU if an eligible CPU exception occurs. Please refer to Section 4.5.3, “Wake Up from Stop or Wait Mode” for details.

### 4.1.4 Block Diagram

Figure 4-1 shows a block diagram of the INT module.

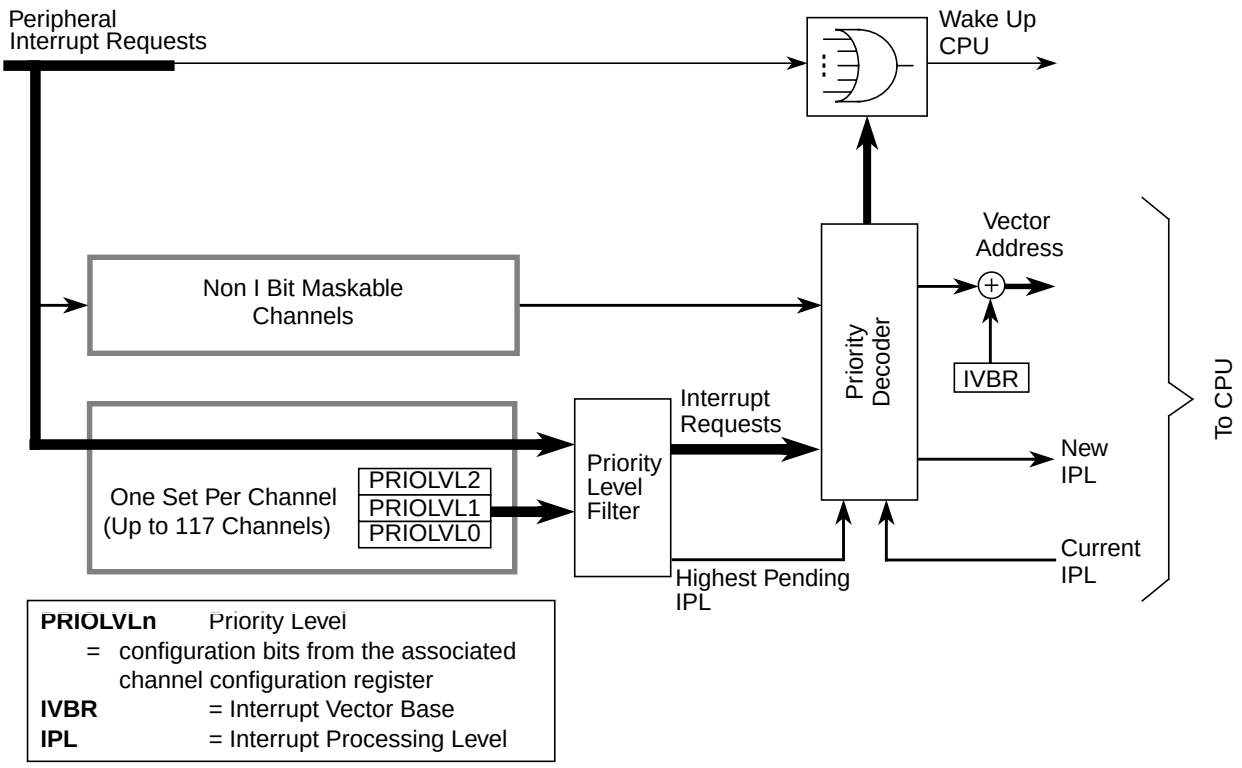



**Figure 4-1. INT Block Diagram**

## 4.2 External Signal Description

The INT module has no external signals.

## 4.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the INT module.

### 4.3.1 Module Memory Map

Table 4-3 gives an overview over all INT module registers.

**Table 4-3. INT Memory Map**

| Address | Use | Access |
|---|---|---|
| 0x000010–0x000011 | Interrupt Vector Base Register (IVBR) | R/W |
| 0x000012–0x000016 | RESERVED | — |
| 0x000017 | Interrupt Request Configuration Address Register (INT_CFADDR) | R/W |
| 0x000018 | Interrupt Request Configuration Data Register 0 (INT_CFDATA0) | R/W |

**Table 4-3. INT Memory Map**

| | | |
|---|---|---|
| 0x000019 | Interrupt Request Configuration Data Register 1 (INT_CFDATA1) | R/W |
| 0x00001A | Interrupt Request Configuration Data Register 2 (INT_CFDATA2 | R/W |
| 0x00001B | Interrupt Request Configuration Data Register 3 (INT_CFDATA3) | R/W |
| 0x00001C | Interrupt Request Configuration Data Register 4 (INT_CFDATA4) | R/W |
| 0x00001D | Interrupt Request Configuration Data Register 5 (INT_CFDATA5) | R/W |
| 0x00001E | Interrupt Request Configuration Data Register 6 (INT_CFDATA6) | R/W |
| 0x00001F | Interrupt Request Configuration Data Register 7 (INT_CFDATA7) | R/W |

## 4.3.2 Register Descriptions

This section describes in address order all the INT module registers and their individual bits.

| Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000010 | IVBR | R | IVB_ADDR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x000011 | | R | IVB_ADDR[7:1] | | | | | | | 0 |
| | | W | | | | | | | | |
| 0x000017 | INT_CFADDR | R | 0 | INT_CFADDR[6:3] | | | | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000018 | INT_CFDATA0 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x000019 | INT_CFDATA1 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x00001A | INT_CFDATA2 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x00001B | INT_CFDATA3 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x00001C | INT_CFDATA4 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 4-2. INT Register Summary**

| Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00001D | INT_CFDATA5 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x00001E | INT_CFDATA6 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x00001F | INT_CFDATA7 | R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 4-2. INT Register Summary**

### 4.3.2.1 Interrupt Vector Base Register (IVBR)

Address: 0x000010

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IVB_ADDR[15:1] | | | | | | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Figure 4-3. Interrupt Vector Base Register (IVBR)**

Read: Anytime

Write: Anytime

**Table 4-4. IVBR Field Descriptions**

| Field | Description |
|---|---|
| 15–1<br>IVB_ADDR<br>[15:1] | **Interrupt Vector Base Address Bits** — These bits represent the upper 15 bits of all vector addresses. Out of reset these bits are set to 0xFFFE (i.e., vectors are located at 0xFFFE00–0xFFFFFF).<br>**Note:** A system reset will initialize the interrupt vector base register with “0xFFFE” before it is used to determine the reset vector address. Therefore, changing the IVBR has no effect on the location of the reset vector (0xFFFFFC–0xFFFFFF). |

### 4.3.2.2 Interrupt Request Configuration Address Register (INT_CFADDR)

Address: 0x000017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | INT_CFADDR[6:3] | | | | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 4-4. Interrupt Configuration Address Register (INT_CFADDR)**

Read: Anytime

Write: Anytime

**Table 4-5. INT_CFADDR Field Descriptions**

| Field | Description |
|---|---|
| 6–3<br>INT_CFADDR[6:3] | **Interrupt Request Configuration Data Register Select Bits** — These bits determine which of the 128 configuration data registers are accessible in the 8 register window at INT_CFDATA0–7.<br>The hexadecimal value written to this register corresponds to the upper 4 bits of the vector number (multiply with 4 to get the vector address offset).<br>If, for example, the value 0x70 is written to this register, the configuration data register block for the 8 interrupt vector requests starting with vector at address (vector base + (0x70*4 = 0x0001C0)) is selected and can be accessed as INT_CFDATA0–7. |

### 4.3.2.3 Interrupt Request Configuration Data Registers (INT_CFDATA0–7)

The eight register window visible at addresses INT_CFDATA0–7 contains the configuration data for the block of eight interrupt requests (out of 128) selected by the interrupt configuration address register (INT_CFADDR) in ascending order. INT_CFDATA0 represents the interrupt configuration data register of the vector with the lowest address in this block, while INT_CFDATA7 represents the interrupt configuration data register of the vector with the highest address, respectively.

Address: 0x000018

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-5. Interrupt Request Configuration Data Register 0 (INT_CFDATA0)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x000019

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-6. Interrupt Request Configuration Data Register 1 (INT_CFDATA1)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x00001A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-7. Interrupt Request Configuration Data Register 2 (INT_CFDATA2)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x00001B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-8. Interrupt Request Configuration Data Register 3 (INT_CFDATA3)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x00001C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-9. Interrupt Request Configuration Data Register 4 (INT_CFDATA4)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x00001D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-10. Interrupt Request Configuration Data Register 5 (INT_CFDATA5)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x00001E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-11. Interrupt Request Configuration Data Register 6 (INT_CFDATA6)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x00001F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1(1) |

= Unimplemented or Reserved

**Figure 4-12. Interrupt Request Configuration Data Register 7 (INT_CFDATA7)**

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Read: Anytime

Write: Anytime

**Table 4-6. INT_CFDATA0–7 Field Descriptions**

| Field | Description |
|---|---|
| 2–0 PRIOLVL[2:0] | **Interrupt Request Priority Level Bits** — The PRIOLVL[2:0] bits configure the interrupt request priority level of the associated interrupt request. Out of reset all interrupt requests are enabled at the lowest active level (“1”). Please also refer to Table 4-7 for available interrupt request priority levels.<br>**Note:** Write accesses to configuration data registers of unused interrupt channels are ignored and read accesses return all 0s. For information about what interrupt channels are used in a specific MCU, please refer to the Device Reference Manual for that MCU.<br>**Note:** When non I-bit maskable request vectors are selected, writes to the corresponding INT_CFDATA registers are ignored and read accesses return all 0s. The corresponding vectors do not have configuration data registers associated with them.<br>**Note:** Write accesses to the configuration register for the spurious interrupt vector request (vector base + 0x0001DC) are ignored and read accesses return 0x07 (request is handled by the CPU, PRIOLVL = 7). |

**Table 4-7. Interrupt Priority Levels**

| Priority | PRIOLVL2 | PRIOLVL1 | PRIOLVL0 | Meaning |
|---|---|---|---|---|
| | 0 | 0 | 0 | Interrupt request is disabled |
| low | 0 | 0 | 1 | Priority level 1 |
| | 0 | 1 | 0 | Priority level 2 |
| | 0 | 1 | 1 | Priority level 3 |
| | 1 | 0 | 0 | Priority level 4 |
| | 1 | 0 | 1 | Priority level 5 |
| | 1 | 1 | 0 | Priority level 6 |
| high | 1 | 1 | 1 | Priority level 7 |

## 4.4 Functional Description

The INT module processes all exception requests to be serviced by the CPU module. These exceptions include interrupt vector requests and reset vector requests. Each of these exception types and their overall priority level is discussed in the subsections below.

### 4.4.1 S12Z Exception Requests

The CPU handles both reset requests and interrupt requests. The INT module contains registers to configure the priority level of each I-bit maskable interrupt request which can be used to implement an interrupt priority scheme. This also includes the possibility to nest interrupt requests. A priority decoder is used to evaluate the relative priority of pending interrupt requests.

### 4.4.2 Interrupt Prioritization

After system reset all I-bit maskable interrupt requests are configured to be enabled, are set up to be handled by the CPU and have a pre-configured priority level of 1. Exceptions to this rule are the non-maskable interrupt requests and the spurious interrupt vector request at (vector base + 0x0001DC) which cannot be disabled, are always handled by the CPU and have a fixed priority levels. A priority level of 0 effectively disables the associated I-bit maskable interrupt request.

If more than one interrupt request is configured to the same interrupt priority level the interrupt request with the higher vector address wins the prioritization.

The following conditions must be met for an I-bit maskable interrupt request to be processed.

1. The local interrupt enabled bit in the peripheral module must be set.
2. The setup in the configuration register associated with the interrupt request channel must meet the following conditions:
   a) The priority level must be set to non zero.
   b) The priority level must be greater than the current interrupt processing level in the condition code register (CCW) of the CPU (PRIOLVL[2:0] > IPL[2:0]).
3. The I-bit in the condition code register (CCW) of the CPU must be cleared.
4. There is no access violation interrupt request pending.
5. There is no SYS, SWI, SPARE, TRAP, Machine Exception or $\overline{\text{XIRQ}}$ request pending.

**NOTE**

All non I-bit maskable interrupt requests always have higher priority than I-bit maskable interrupt requests. If an I-bit maskable interrupt request is interrupted by a non I-bit maskable interrupt request, the currently active interrupt processing level (IPL) remains unaffected. It is possible to nest non I-bit maskable interrupt requests, e.g., by nesting SWI, SYS or TRAP calls.

#### 4.4.2.1 Interrupt Priority Stack

The current interrupt processing level (IPL) is stored in the condition code register (CCW) of the CPU. This way the current IPL is automatically pushed to the stack by the standard interrupt stacking procedure. The new IPL is copied to the CCW from the priority level of the highest priority active interrupt request channel which is configured to be handled by the CPU. The copying takes place when the interrupt vector is fetched. The previous IPL is automatically restored from the stack by executing the RTI instruction.

### 4.4.3 Priority Decoder

The INT module contains a priority decoder to determine the relative priority for all interrupt requests pending for the CPU.

A CPU interrupt vector is not supplied until the CPU requests it. Therefore, it is possible that a higher priority interrupt request could override the original exception which caused the CPU to request the vector. In this case, the CPU will receive the highest priority vector and the system will process this exception first instead of the original request.

If the interrupt source is unknown (for example, in the case where an interrupt request becomes inactive after the interrupt has been recognized, but prior to the vector request), the vector address supplied to the CPU defaults to that of the spurious interrupt vector.

**NOTE**

Care must be taken to ensure that all exception requests remain active until the system begins execution of the applicable service routine; otherwise, the exception request may not get processed at all or the result may be a spurious interrupt request (vector at address (vector base + 0x0001DC)).

### 4.4.4 Reset Exception Requests

The INT module supports one system reset exception request. The different reset types are mapped to this vector (for details please refer to the Clock and Power Management Unit module (CPMU)):

1. Pin reset
2. Power-on reset
3. Low-voltage reset
4. Clock monitor reset request
5. COP watchdog reset request

### 4.4.5 Exception Priority

The priority (from highest to lowest) and address of all exception vectors issued by the INT module upon request by the CPU are shown in Table 4-8. Generally, all non-maskable interrupts have higher priorities than maskable interrupts. Please note that between the four software interrupts (Unimplemented op-code trap page1/page2 requests, SWI request, SYS request) there is no real priority defined since they cannot occur simultaneously (the S12Z CPU executes one instruction at a time).

**Table 4-8. Exception Vector Map and Priority**

| Vector Address[1] | Source |
|---|---|
| 0xFFFFFC | Pin reset, power-on reset, low-voltage reset, clock monitor reset, COP watchdog reset |
| (Vector base + 0x0001F8) | Unimplemented page1 op-code trap (SPARE) vector request |
| (Vector base + 0x0001F4) | Unimplemented page2 op-code trap (TRAP) vector request |
| (Vector base + 0x0001F0) | Software interrupt instruction (SWI) vector request |
| (Vector base + 0x0001EC) | System call interrupt instruction (SYS) vector request |

**Table 4-8. Exception Vector Map and Priority**

| Vector Address[1] | Source |
|---|---|
| (Vector base + 0x0001E8) | Machine exception vector request |
| (Vector base + 0x0001E4) | Reserved |
| (Vector base + 0x0001E0) | Reserved |
| (Vector base + 0x0001DC) | Spurious interrupt |
| (Vector base + 0x0001D8) | $\overline{\text{XIRQ}}$ interrupt request |
| (Vector base + 0x0001D4) | $\overline{\text{IRQ}}$ interrupt request |
| (Vector base + 0x000010<br>..<br>Vector base + 0x0001D0) | Device specific I-bit maskable interrupt sources (priority determined by the associated configuration registers, in descending order) |

1. 24 bits vector address based

### 4.4.6 Interrupt Vector Table Layout

The interrupt vector table contains 128 entries, each 32 bits (4 bytes) wide. Each entry contains a 24-bit address (3 bytes) which is stored in the 3 low-significant bytes of the entry. The content of the most significant byte of a vector-table entry is ignored. Figure 4-13 illustrates the vector table entry format.

| Bits | [31:24] | [23:0] |
|---|---|---|
| | (unused) | ISR Address |

**Figure 4-13. Interrupt Vector Table Entry**

## 4.5 Initialization/Application Information

### 4.5.1 Initialization

After system reset, software should:

- Initialize the interrupt vector base register if the interrupt vector table is not located at the default location (0xFFFE00–0xFFFFFB).
- Initialize the interrupt processing level configuration data registers (INT_CFADDR, INT_CFDATA0–7) for all interrupt vector requests with the desired priority levels. It might be a good idea to disable unused interrupt requests.
- Enable I-bit maskable interrupts by clearing the I-bit in the CCW.
- Enable the X-bit maskable interrupt by clearing the X-bit in the CCW (if required).

### 4.5.2 Interrupt Nesting

The interrupt request priority level scheme makes it possible to implement priority based interrupt request nesting for the I-bit maskable interrupt requests.

- I-bit maskable interrupt requests can be interrupted by an interrupt request with a higher priority, so that there can be up to seven nested I-bit maskable interrupt requests at a time (refer to Figure 4-14 for an example using up to three nested interrupt requests).

I-bit maskable interrupt requests cannot be interrupted by other I-bit maskable interrupt requests per default. In order to make an interrupt service routine (ISR) interruptible, the ISR must explicitly clear the I-bit in the CCW (CLI). After clearing the I-bit, I-bit maskable interrupt requests with higher priority can interrupt the current ISR.

An ISR of an interruptible I-bit maskable interrupt request could basically look like this:

- Service interrupt, e.g., clear interrupt flags, copy data, etc.
- Clear I-bit in the CCW by executing the CPU instruction CLI (thus allowing interrupt requests with higher priority)
- Process data
- Return from interrupt by executing the instruction RTI

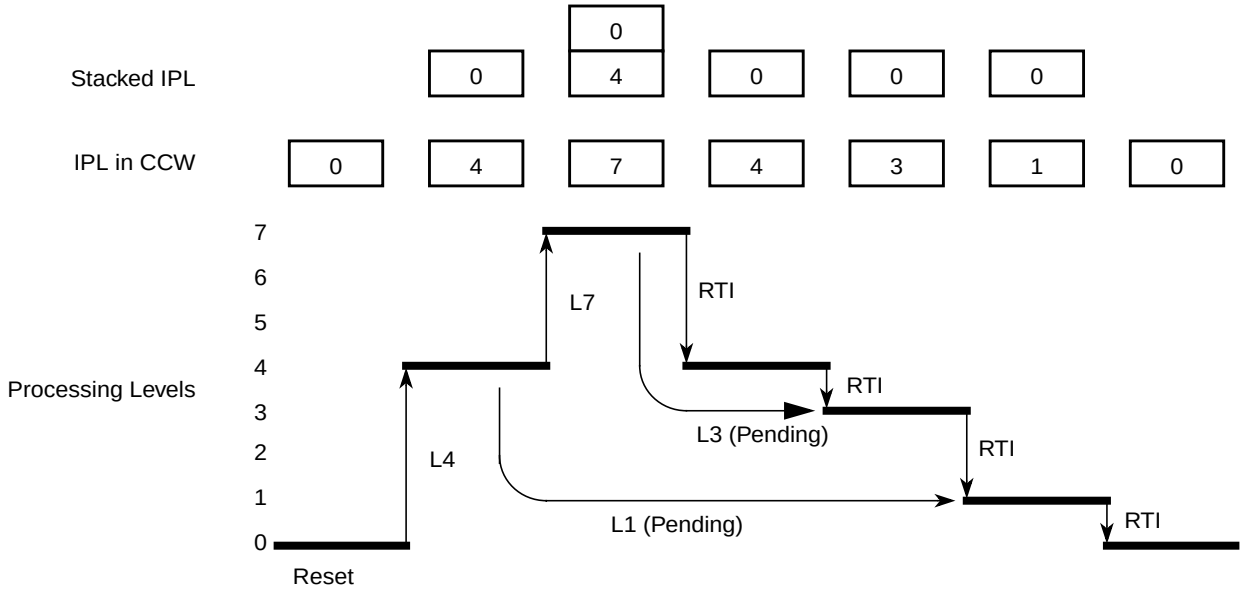



**Figure 4-14. Interrupt Processing Example**

## 4.5.3 Wake Up from Stop or Wait Mode

### 4.5.3.1 CPU Wake Up from Stop or Wait Mode

Every I-bit maskable interrupt request which is configured to be handled by the CPU is capable of waking the MCU from stop or wait mode. Additionally machine exceptions can wake-up the MCU from stop or wait mode.

To determine whether an I-bit maskable interrupts is qualified to wake up the CPU or not, the same settings as in normal run mode are applied during stop or wait mode:

- If the I-bit in the CCW is set, all I-bit maskable interrupts are masked from waking up the MCU.
- An I-bit maskable interrupt is ignored if it is configured to a priority level below or equal to the current IPL in CCW.

The X-bit maskable interrupt request can wake up the MCU from stop or wait mode at anytime, even if the X-bit in CCW is set[1]. If the X-bit maskable interrupt request is used to wake-up the MCU with the X-

bit in the CCW set, the associated ISR is not called. The CPU then resumes program execution with the instruction following the WAI or STOP instruction. This feature works following the same rules like any interrupt request, i.e. care must be taken that the X-bit maskable interrupt request used for wake-up remains active at least until the system begins execution of the instruction following the WAI or STOP instruction; otherwise, wake-up may not occur.

1. The capability of the $\overline{\text{XIRQ}}$ pin to wake-up the MCU with the X bit set may not be available if, for example, the $\overline{\text{XIRQ}}$ pin is shared with other peripheral modules on the device. Please refer to the Port Integration Module (PIM) section of the MCU reference manual for details.

# Chapter 5
# Background Debug Controller (S12ZBDCV2)

**Table 5-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V2.03 | 13.Nov.2012 | Section 5.3.2.2 | Included SYNC_PC / Stop mode dependency |
| V2.04 | 03.Dec.2012 | Section 5.1.3.3 | Included BACKGROUND/ Stop mode dependency |
| V2.05 | 22.Jan.2013 | Section 5.3.2.2 | Improved NORESP description and added STEP1/ Wait mode dependency |
| V2.06 | 22.Mar.2013 | Section 5.3.2.2 | Improved NORESP description of STEP1/ Wait mode dependency |
| V2.07 | 11.Apr..2013 | Section 5.1.3.3.1 | Improved STOP and BACKGROUND interdepency description |
| V2.08 | 31.May..2013 | Section 5.4.4.4<br>Section 5.4.7.1 | Removed misleading WAIT and BACKGROUND interdepency description<br>Added subsection dedicated to Long-ACK |
| V2.09 | 29.Aug..2013 | Section 5.4.4.12 | Noted that READ_DBGTB is only available for devices featuring a trace buffer. |

## 5.1 Introduction

The background debug controller (BDC) is a single-wire, background debug system implemented in on-chip hardware for minimal CPU intervention. The device BKGD pin interfaces directly to the BDC.

The S12ZBDC maintains the standard S12 serial interface protocol but introduces an enhanced handshake protocol and enhanced BDC command set to support the linear instruction set family of S12Z devices and offer easier, more flexible internal resource access over the BDC serial interface.

### 5.1.1 Glossary

**Table 5-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| DBG | On chip Debug Module |
| BDM | Active Background Debug Mode |
| CPU | S12Z CPU |
| SSC | Special Single Chip Mode (device operating mode |
| NSC | Normal Single Chip Mode (device operating mode) |
| BDCSI | Background Debug Controller Serial Interface. This refers to the single pin BKGD serial interface. |
| EWAIT | Optional S12 feature which allows external devices to delay external accesses until deassertion of EWAIT |

## 5.1.2 Features

The BDC includes these distinctive features:

- Single-wire communication with host development system
- SYNC command to determine communication rate
- Genuine non-intrusive handshake protocol
- Enhanced handshake protocol for error detection and stop mode recognition
- Active out of reset in special single chip mode
- Most commands not requiring active BDM, for minimal CPU intervention
- Full global memory map access without paging
- Simple flash mass erase capability

## 5.1.3 Modes of Operation

S12 devices feature power modes (run, wait, and stop) and operating modes (normal single chip, special single chip). Furthermore, the operation of the BDC is dependent on the device security status.

### 5.1.3.1 BDC Modes

The BDC features module specific modes, namely disabled, enabled and active. These modes are dependent on the device security and operating mode. In active BDM the CPU ceases execution, to allow BDC system access to all internal resources including CPU internal registers.

### 5.1.3.2 Security and Operating mode Dependency

In device run mode the BDC dependency is as follows

- Normal modes, unsecure device
  General BDC operation available. The BDC is disabled out of reset.
- Normal modes, secure device
  BDC disabled. No BDC access possible.
- Special single chip mode, unsecure
  BDM active out of reset. All BDC commands are available.
- Special single chip mode, secure
  BDM active out of reset. Restricted command set available.

When operating in secure mode, BDC operation is restricted to allow checking and clearing security by mass erasing the on-chip flash memory. Secure operation prevents BDC access to on-chip memory other than mass erase. The BDC command set is restricted to those commands classified as Always-available.

### 5.1.3.3 Low-Power Modes

#### 5.1.3.3.1 Stop Mode

The execution of the CPU STOP instruction leads to stop mode only when all bus masters (CPU, or others, depending on the device) have finished processing. The operation during stop mode depends on the ENBDC and BDCCIS bit settings as summarized in Table 5-3

**Table 5-3. BDC STOP Operation Dependencies**

| ENBDC | BDCCIS | Description Of Operation |
|---|---|---|
| 0 | 0 | BDC has no effect on STOP mode. |
| 0 | 1 | BDC has no effect on STOP mode. |
| 1 | 0 | Only BDCSI clock continues |
| 1 | 1 | All clocks continue |

A disabled BDC has no influence on stop mode operation. In this case the BDCSI clock is disabled in stop mode thus it is not possible to enable the BDC from within stop mode.

#### STOP Mode With BDC Enabled And BDCCIS Clear

If the BDC is enabled and BDCCIS is clear, then the BDC prevents the BDCCLK clock (Figure 5-5) from being disabled in stop mode. This allows BDC communication to continue throughout stop mode in order to access the BDCCSR register. All other device level clock signals are disabled on entering stop mode.

**NOTE**

This is intended for application debugging, not for fast flash programming. Thus the CLKSW bit must be clear to map the BDCSI to BDCCLK.

With the BDC enabled, an internal acknowledge delays stop mode entry and exit by 2 BDCSI clock + 2 bus clock cycles. If no other module delays stop mode entry and exit, then these additional clock cycles represent a difference between the debug and not debug cases. Furthermore if a BDC internal access is being executed when the device is entering stop mode, then the stop mode entry is delayed until the internal access is complete (typically for 1 bus clock cycle).

Accesses to the internal memory map are not possible when the internal device clocks are disabled. Thus attempted accesses to memory mapped resources are suppressed and the NORESP flag is set. Resources can be accessed again by the next command received following exit from Stop mode.

A BACKGROUND command issued whilst in stop mode remains pending internally until the device leaves stop mode. This means that subsequent active BDM commands, issued whilst BACKGROUND is pending, set the ILLCMD flag because the device is not yet in active BDM.

If ACK handshaking is enabled, then the first ACK, following a stop mode entry is long to indicate a stop exception. The BDC indicates a stop mode occurrence by setting the BDCCSR bit STOP. If the host attempts further communication before the ACK pulse generation then the OVRUN bit is set.

#### STOP Mode With BDC Enabled And BDCCIS Set

If the BDC is enabled and BDCCIS is set, then the BDC prevents core clocks being disabled in stop mode. This allows BDC communication, for access of internal memory mapped resources, but not CPU registers, to continue throughout stop mode.

A BACKGROUND command issued whilst in stop mode remains pending internally until the device leaves stop mode. This means that subsequent active BDM commands, issued whilst BACKGROUND is pending, set the ILLCMD flag because the device is not yet in active BDM.

If ACK handshaking is enabled, then the first ACK, following a stop mode entry is long to indicate a stop exception. The BDC indicates a stop mode occurrence by setting the BDCCSR bit STOP. If the host attempts further communication before the ACK pulse generation then the OVRUN bit is set.

#### 5.1.3.3.2 Wait Mode

The device enters wait mode when the CPU starts to execute the WAI instruction. The second part of the WAI instruction (return from wait mode) can only be performed when an interrupt occurs. Thus on entering wait mode the CPU is in the middle of the WAI instruction and cannot permit access to CPU internal resources, nor allow entry to active BDM. Thus only commands classified as Non-Intrusive or Always-Available are possible in wait mode.

On entering wait mode, the WAIT flag in BDCCSR is set. If the ACK handshake protocol is enabled then the first ACK generated after WAIT has been set is a long-ACK pulse. Thus the host can recognize a wait mode occurrence. The WAIT flag remains set and cannot be cleared whilst the device remains in wait mode. After the device leaves wait mode the WAIT flag can be cleared by writing a “1” to it.

A BACKGROUND command, issued whilst in wait mode with ACK disabled, sets the NORESP bit and the BDM active request remains pending internally until the CPU leaves wait mode due to an interrupt. The device then enters BDM with the PC pointing to the address of the first instruction of the ISR. Further Non-Intrusive or Always-Available commands are possible, in this pending state, but attempted Active-Background commands set NORESP and ILLCMD because the BDC is not in active BDM state.

A BACKGROUND command, issued whilst in wait mode with ACK enabled, remains pending internally until the CPU leaves wait mode due to an interrupt. A long ACK is generated on leaving wait mode. If the host attempts communication before the ACK pulse generation then the OVRUN bit is set.

Similarly the STEP1 command issued from a WAI instruction cannot be completed by the CPU until the CPU leaves wait mode due to an interrupt. The first STEP1 into wait mode sets the BDCCSR WAIT bit.

If the part is still in Wait mode and a further STEP1 is carried out then the NORESP and ILLCMD bits are set because the device is no longer in active BDM for the duration of WAI execution.

### 5.1.4 Block Diagram

A block diagram of the BDC is shown in Figure 5-1.

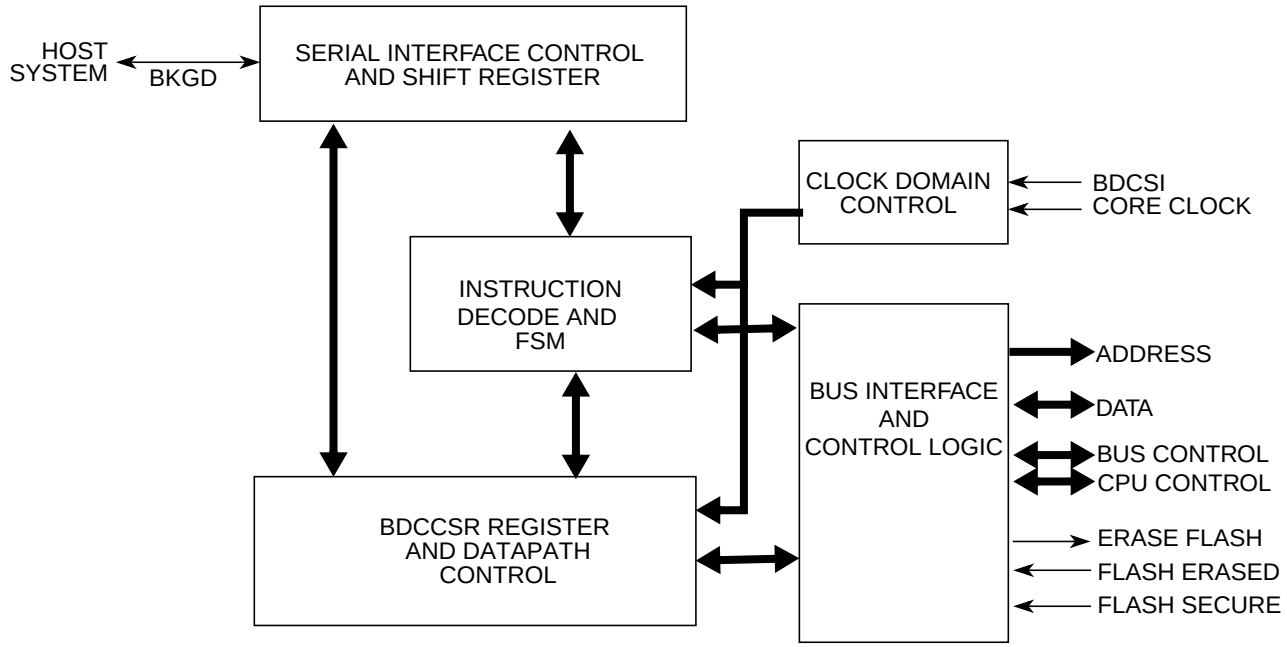



**Figure 5-1. BDC Block Diagram**

## 5.2 External Signal Description

A single-wire interface pin (BKGD) is used to communicate with the BDC system. During reset, this pin is a device mode select input. After reset, this pin becomes the dedicated serial interface pin for the BDC.

BKGD is a pseudo-open-drain pin with an on-chip pull-up. Unlike typical open-drain pins, the external RC time constant on this pin due to external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speed-up pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 5.4.6” for more details.

## 5.3 Memory Map and Register Definition

### 5.3.1 Module Memory Map

Table 5-4 shows the BDC memory map.

**Table 5-4. BDC Memory Map**

| Global Address | Module | Size (Bytes) |
|---|---|---|
| Not Applicable | BDC registers | 2 |

## 5.3.2 Register Descriptions

The BDC registers are shown in Figure 5-2. Registers are accessed only by host-driven communications to the BDC hardware using READ_BDCCSR and WRITE_BDCCSR commands. They are not accessible in the device memory map.

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Not Applicable | BDCCSRH | R | ENBDC | BDMACT | BDCCIS | 0 | STEAL | CLKSW | UNSEC | ERASE |
| | | W | | | | | | | | |
| Not Applicable | BDCCSRL | R | WAIT | STOP | RAMWF | OVRUN | NORESP | RDINV | ILLACC | ILLCMD |
| | | W | | | | | | | | |

= Unimplemented, Reserved  0 = Always read zero

**Figure 5-2. BDC Register Summary**

### 5.3.2.1 BDC Control Status Register High (BDCCSRH)

Register Address: This register is not in the device memory map. It is accessible using BDC inherent addressing commands

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ENBDC | BDMACT | BDCCIS | 0 | STEAL | CLKSW | UNSEC | ERASE |
| W | | | | | | | | |
| Reset | | | | | | | | |
| Secure AND SSC-Mode | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Unsecure AND SSC-Mode | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| Secure AND NSC-Mode | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Unsecure AND NSC-Mode | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

= Unimplemented, Reserved

0 = Always read zero

**Figure 5-3. BDC Control Status Register High (BDCCSRH)**

Read: All modes through BDC operation only.

Write: All modes through BDC operation only, when not secured, but subject to the following:

- Bits 7,3 and 2 can only be written by WRITE_BDCCSR commands.
- Bit 5 can only be written by WRITE_BDCCSR commands when the device is not in stop mode.
- Bits 6, 1 and 0 cannot be written. They can only be updated by internal hardware.

**Table 5-5. BDCCSRH Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ENBDC | **Enable BDC** — This bit controls whether the BDC is enabled or disabled. When enabled, active BDM can be entered and non-intrusive commands can be carried out. When disabled, active BDM is not possible and the valid command set is restricted. Further information is provided in Table 5-7.<br>0 BDC disabled<br>1 BDC enabled<br>**Note:** ENBDC is set out of reset in special single chip mode. |
| 6<br>BDMACT | **BDM Active Status** — This bit becomes set upon entering active BDM. BDMACT is cleared as part of the active BDM exit sequence.<br>0 BDM not active<br>1 BDM active<br>**Note:** BDMACT is set out of reset in special single chip mode. |
| 5<br>BDCCIS | **BDC Continue In Stop** — If ENBDC is set then BDCCIS selects the type of BDC operation in stop mode (as shown in Table 5-3). If ENBDC is clear, then the BDC has no effect on stop mode and no BDC communication is possible.If ACK pulse handshaking is enabled, then the first ACK pulse following stop mode entry is a long ACK. This bit cannot be written when the device is in stop mode.<br>0 Only the BDCSI clock continues in stop mode<br>1 All clocks continue in stop mode |
| 3<br>STEAL | **Steal enabled with ACK**— This bit forces immediate internal accesses with the ACK handshaking protocol enabled. If ACK handshaking is disabled then BDC accesses steal the next bus cycle.<br>0 If ACK is enabled then BDC accesses await a free cycle, with a timeout of 512 cycles<br>1 If ACK is enabled then BDC accesses are carried out in the next bus cycle |
| 2<br>CLKSW | **Clock Switch** — The CLKSW bit controls the BDCSI clock source. This bit is initialized to “0” by each reset and can be written to “1”. Once it has been set, it can only be cleared by a reset. When setting CLKSW a minimum delay of 150 cycles at the initial clock speed must elapse before the next command can be sent. This guarantees that the start of the next BDC command uses the new clock for timing subsequent BDC communications.<br>0 BDCCLK used as BDCSI clock source<br>1 Device fast clock used as BDCSI clock source<br>**Note:** Refer to the device specification to determine which clock connects to the BDCCLK and fast clock inputs. |
| 1<br>UNSEC | **Unsecure** — If the device is unsecure, the UNSEC bit is set automatically.<br>0 Device is secure.<br>1 Device is unsecure.<br>**Note:** When UNSEC is set, the device is unsecure and the state of the secure bits in the on-chip Flash EEPROM can be changed. |
| 0<br>ERASE | **Erase Flash** — This bit can only be set by the dedicated ERASE_FLASH command. ERASE is unaffected by write accesses to BDCCSR. ERASE is cleared either when the mass erase sequence is completed, independent of the actual status of the flash array or by a soft reset.<br>Reading this bit indicates the status of the requested mass erase sequence.<br>0 No flash mass erase sequence pending completion<br>1 Flash mass erase sequence pending completion. |

### 5.3.2.2 BDC Control Status Register Low (BDCCSRL)

Register Address: This register is not in the device memory map. It is accessible using BDC inherent addressing commands

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WAIT | STOP | RAMWF | OVRUN | NORESP | RDINV | ILLACC | ILLCMD |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-4. BDC Control Status Register Low (BDCCSRL)**

Read: BDC access only.

Write: Bits [7:5], [3:0] BDC access only, restricted to flag clearing by writing a “1” to the bit position.
Write: Bit 4 never. It can only be cleared by a SYNC pulse.

If ACK handshaking is enabled then BDC commands with ACK causing a BDCCSRL[3:1] flag setting condition also generate a long ACK pulse. Subsequent commands that are executed correctly generate a normal ACK pulse. Subsequent commands that are not correctly executed generate a long ACK pulse. The first ACK pulse after WAIT or STOP have been set also generates a long ACK. Subsequent ACK pulses are normal, whilst STOP and WAIT remain set.

Long ACK pulses are not immediately generated if an overrun condition is caused by the host driving the BKGD pin low whilst a target ACK is pending, because this would conflict with an attempted host transmission following the BKGD edge. When a whole byte has been received following the offending BKGD edge, the OVRUN bit is still set, forcing subsequent ACK pulses to be long.

Unimplemented BDC opcodes causing the ILLCMD bit to be set do not generate a long ACK because this could conflict with further transmission from the host. If the ILLCMD is set for another reason, then a long ACK is generated for the current command if it is a BDC command with ACK.

**Table 5-6. BDCCSRL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WAIT | **WAIT Indicator Flag** — Indicates that the device entered wait mode. Writing a “1” to this bit whilst in wait mode has no effect. Writing a “1” after exiting wait mode, clears the bit.<br>0 Device did not enter wait mode<br>1 Device entered wait mode. |
| 6<br>STOP | **STOP Indicator Flag** — Indicates that the CPU requested stop mode following a STOP instruction. Writing a “1” to this bit whilst not in stop mode clears the bit. Writing a “1” to this bit whilst in stop mode has no effect. This bit can only be set when the BDC is enabled.<br>0 Device did not enter stop mode<br>1 Device entered stop mode. |
| 5<br>RAMWF | **RAM Write Fault** — Indicates an ECC double fault during a BDC write access to RAM.<br>Writing a “1” to this bit, clears the bit.<br>0 No RAM write double fault detected.<br>1 RAM write double fault detected. |

**Table 5-6. BDCCSRL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>OVRUN | **Overrun Flag** — Indicates unexpected host activity before command completion.<br>This occurs if a new command is received before the current command completion.<br>With ACK enabled this also occurs if the host drives the BKGD pin low whilst a target ACK pulse is pending<br>To protect internal resources from misinterpreted BDC accesses following an overrun, internal accesses are suppressed until a SYNC clears this bit.<br>A SYNC clears the bit.<br>0 No overrun detected.<br>1 Overrun detected when issuing a BDC command. |
| 3<br>NORESP | **No Response Flag** — Indicates that the BDC internal action or data access did not complete. This occurs in the following scenarios:<br>a) If no free cycle for an access is found within 512 core clock cycles. This could typically happen if a code loop without free cycles is executing with ACK enabled and STEAL clear.<br>b) With ACK disabled or STEAL set, when an internal access is not complete before the host starts data/BDCCSRL retrieval or an internal write access is not complete before the host starts the next BDC command.<br>c) Attempted internal memory or SYNC_PC accesses during STOP mode set NORESP if BDCCIS is clear.<br>In the above cases, on setting NORESP, the BDC aborts the access if permitted. (For devices supporting EWAIT, BDC external accesses with EWAIT assertions, prevent a command from being aborted until EWAIT is deasserted).<br>d) If a BACKGROUND command is issued whilst the device is in wait mode the NORESP bit is set but the command is not aborted. The active BDM request is completed when the device leaves wait mode. Furthermore subsequent CPU register access commands during wait mode set the NORESP bit, should it have been cleared.<br>e) If a command is issued whilst awaiting return from Wait mode. This can happen when using STEP1 to step over a CPU WAI instruction, if the CPU has not returned from Wait mode before the next BDC command is received.<br>f) If STEP1 is issued with the BDC enabled as the device enters Wait mode regardless of the BDMACT state.<br><br>When NORESP is set a value of 0xEE is returned for each data byte associated with the current access.<br>Writing a “1” to this bit, clears the bit.<br>0 Internal action or data access completed.<br>1 Internal action or data access did not complete. |
| 2<br>RDINV | **Read Data Invalid Flag** — Indicates invalid read data due to an ECC error during a BDC initiated read access.<br>The access returns the actual data read from the location.<br>Writing a “1” to this bit, clears the bit.<br>0 No invalid read data detected.<br>1 Invalid data returned during a BDC read access. |
| 1<br>ILLACC | **Illegal Access Flag** — Indicates an attempted illegal access. This is set in the following cases:<br>When the attempted access addresses unimplemented memory<br>When the access attempts to write to the flash array<br>When a CPU register access is attempted with an invalid CRN (Section 5.4.5.1).<br>Illegal accesses return a value of 0xEE for each data byte<br>Writing a “1” to this bit, clears the bit.<br>0 No illegal access detected.<br>1 Illegal BDC access detected. |

**Table 5-6. BDCCSRL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>ILLCMD | **Illegal Command Flag** — Indicates an illegal BDC command. This bit is set in the following cases:<br>When an unimplemented BDC command opcode is received.<br>When a DUMP_MEM{_WS}, FILL_MEM{_WS} or READ_SAME{_WS} is attempted in an illegal sequence.<br>When an active BDM command is received whilst BDM is not active<br>When a non Always-available command is received whilst the BDC is disabled or a flash mass erase is ongoing.<br>When a non Always-available command is received whilst the device is secure<br>Read commands return a value of 0xEE for each data byte<br>Writing a “1” to this bit, clears the bit.<br>0 No illegal command detected.<br>1 Illegal BDC command detected. |

## 5.4 Functional Description

### 5.4.1 Security

If the device resets with the system secured, the device clears the BDCCSR UNSEC bit. In the secure state BDC access is restricted to the BDCCSR register. A mass erase can be requested using the ERASE_FLASH command. If the mass erase is completed successfully, the device programs the security bits to the unsecure state and sets the BDC UNSEC bit. If the mass erase is unsuccessful, the device remains secure and the UNSEC bit is not set.

For more information regarding security, please refer to device specific security information.

### 5.4.2 Enabling BDC And Entering Active BDM

BDM can be activated only after being enabled. BDC is enabled by setting the ENBDC bit in the BDCCSR register, via the single-wire interface, using the command WRITE_BDCCSR.

After being enabled, BDM is activated by one of the following[1]:

- The BDC BACKGROUND command
- A CPU BGND instruction
- The DBG Breakpoint mechanism

Alternatively BDM can be activated directly from reset when resetting into Special Single Chip Mode.

The BDC is ready for receiving the first command 10 core clock cycles after the deassertion of the internal reset signal. This is delayed relative to the external pin reset as specified in the device reset documentation. On S12Z devices an NVM initialization phase follows reset. During this phase the BDC commands classified as always available are carried out immediately, whereas other BDC commands are subject to delayed response due to the NVM initialization phase.

**NOTE**

After resetting into SSC mode, the initial PC address must be supplied by the host using the WRITE_Rn command before issuing the GO command.

1. BDM active immediately out of special single-chip reset.

When BDM is activated, the CPU finishes executing the current instruction. Thereafter only BDC commands can affect CPU register contents until the BDC GO command returns from active BDM to user code or a device reset occurs. When BDM is activated by a breakpoint, the type of breakpoint used determines if BDM becomes active before or after execution of the next instruction.

**NOTE**

Attempting to activate BDM using a BGND instruction whilst the BDC is disabled, the CPU requires clock cycles for the attempted BGND execution. However BACKGROUND commands issued whilst the BDC is disabled are ignored by the BDC and the CPU execution is not delayed.

### 5.4.3 Clock Source

The BDC clock source can be mapped to a constant frequency clock source or a PLL based fast clock. The clock source for the BDC is selected by the CLKSW bit as shown in Figure 5-5. The BDC internal clock is named BDCSI clock. If BDCSI clock is mapped to the BDCCLK by CLKSW then the serial interface communication is not affected by bus/core clock frequency changes. If the BDC is mapped to BDCFCLK then the clock is connected to a PLL derived source at device level (typically bus clock), thus can be subject to frequency changes in application. Debugging through frequency changes requires SYNC pulses to re-synchronize. The sources of BDCCLK and BDCFCLK are specified at device level.

BDC accesses of internal device resources always use the device core clock. Thus if the ACK handshake protocol is not enabled, the clock frequency relationship must be taken into account by the host.

When changing the clock source via the CLKSW bit a minimum delay of 150 cycles at the initial clock speed must elapse before a SYNC can be sent. This guarantees that the start of the next BDC command uses the new clock for timing subsequent BDC communications.

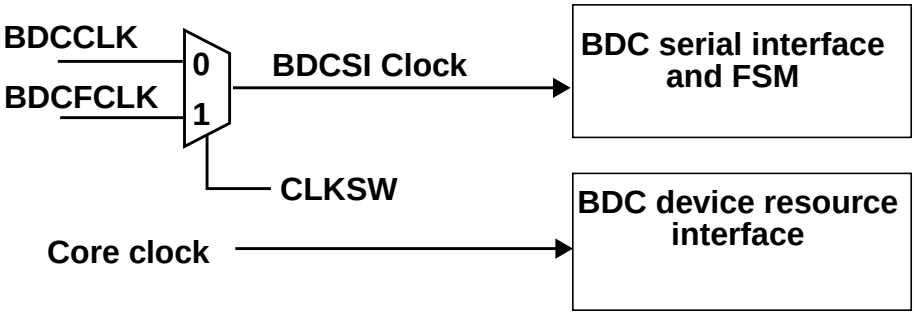



**Figure 5-5. Clock Switch**

### 5.4.4 BDC Commands

BDC commands can be classified into three types as shown in Table 5-7.

**Table 5-7. BDC Command Types**

| Command Type | Secure Status | BDC Status | CPU Status | Command Set |
|---|---|---|---|---|
| Always-available | Secure or Unsecure | Enabled or Disabled | — | • Read/write access to BDCCSR<br>• Mass erase flash memory using ERASE_FLASH<br>• SYNC<br>• ACK enable/disable |
| Non-intrusive | Unsecure | Enabled | Code execution allowed | • Read/write access to BDCCSR<br>• Memory access<br>• Memory access with status<br>• Mass erase flash memory using ERASE_FLASH<br>• Debug register access<br>• BACKGROUND<br>• SYNC<br>• ACK enable/disable |
| Active background | Unsecure | Active | Code execution halted | • Read/write access to BDCCSR<br>• Memory access<br>• Memory access with status<br>• Mass erase flash memory using ERASE_FLASH<br>• Debug register access<br>• Read or write CPU registers<br>• Single-step the application<br>• Exit active BDM to return to the application program (GO)<br>• SYNC<br>• ACK enable/disable |

Non-intrusive commands are used to read and write target system memory locations and to enter active BDM. Target system memory includes all memory and registers within the global memory map, including external memory.

Active background commands are used to read and write all memory locations and CPU resources. Furthermore they allow single stepping through application code and to exit from active BDM.

Non-intrusive commands can only be executed when the BDC is enabled and the device unsecure. Active background commands can only be executed when the system is not secure and is in active BDM.

Non-intrusive commands do not require the system to be in active BDM for execution, although, they can still be executed in this mode. When executing a non-intrusive command with the ACK pulse handshake protocol disabled, the BDC steals the next bus cycle for the access. If an operation requires multiple cycles, then multiple cycles can be stolen. Thus if stolen cycles are not free cycles, the application code execution is delayed. The delay is negligible because the BDC serial transfer rate dictates that such accesses occur infrequently.

For data read commands, the external host must wait at least 16 BDCSI clock cycles after sending the address before attempting to obtain the read data. This is to be certain that valid data is available in the BDC shift register, ready to be shifted out. For write commands, the external host must wait 16 bdcsi cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDC shift register before the write has been completed. The external host must wait at least for 16 bdcsi cycles after a control command before starting any new serial command.

If the ACK pulse handshake protocol is enabled and STEAL is cleared, then the BDC waits for the first free bus cycle to make a non-intrusive access. If no free bus cycle occurs within 512 core clock cycles then the BDC aborts the access, sets the NORESP bit and uses a long ACK pulse to indicate an error condition to the host.

Table 5-8 summarizes the BDC command set. The subsequent sections describe each command in detail and illustrate the command structure in a series of packets, each consisting of eight bit times starting with a falling edge. The bar across the top of the blocks indicates that the BKGD line idles in the high state. The time for an 8-bit command is 8 × 16 target BDCSI clock cycles.

The nomenclature below is used to describe the structure of the BDC commands. Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

```
/       =       separates parts of the command
d       =       delay 16 target BDCSI clock cycles (DLY)
dack    =       delay (16 cycles) no ACK; or delay (=> 32 cycles) then ACK.(DACK)
ad24    =       24-bit memory address in the host-to-target direction
rd8     =       8 bits of read data in the target-to-host direction
rd16    =       16 bits of read data in the target-to-host direction
rd24    =       24 bits of read data in the target-to-host direction
rd32    =       32 bits of read data in the target-to-host direction
rd64    =       64 bits of read data in the target-to-host direction
rd.sz   =       read data, size defined by sz, in the target-to-host direction
wd8     =       8 bits of write data in the host-to-target direction
wd16    =       16 bits of write data in the host-to-target direction
wd32    =       32 bits of write data in the host-to-target direction
wd.sz   =       write data, size defined by sz, in the host-to-target direction
ss      =       the contents of BDCCSRL in the target-to-host direction
sz      =       memory operand size (00 = byte, 01 = word, 10 = long)
                (sz = 11 is reserved and currently defaults to long)
crn     =       core register number, 32-bit data width
WS      =       command suffix signaling the operation is with status
```

**Table 5-8. BDC Command Summary**

| Command Mnemonic | Command Classification | ACK | Command Structure | Description |
|---|---|---|---|---|
| SYNC | Always Available | N/A | N/A[1] | Request a timed reference pulse to determine the target BDC communication speed |
| ACK_DISABLE | Always Available | No | 0x03/d | Disable the communication handshake. This command does not issue an ACK pulse. |
| ACK_ENABLE | Always Available | Yes | 0x02/dack | Enable the communication handshake. Issues an ACK pulse after the command is executed. |
| BACKGROUND | Non-Intrusive | Yes | 0x04/dack | Halt the CPU if ENBDC is set. Otherwise, ignore as illegal command. |

**Table 5-8. BDC Command Summary (continued)**

| Command Mnemonic | Command Classification | ACK | Command Structure | Description |
|---|---|---|---|---|
| DUMP_MEM.sz | Non-Intrusive | Yes | (0x32+4 x sz)/dack/rd.sz | Dump (read) memory based on operand size (sz). Used with READ_MEM to dump large blocks of memory. An initial READ_MEM is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM commands retrieve sequential operands. |
| DUMP_MEM.sz_WS | Non-Intrusive | No | (0x33+4 x sz)/d/ss/rd.sz | Dump (read) memory based on operand size (sz) and report status. Used with READ_MEM{_WS} to dump large blocks of memory. An initial READ_MEM{_WS} is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM{_WS} commands retrieve sequential operands. |
| FILL_MEM.sz | Non-Intrusive | Yes | (0x12+4 x sz)/wd.sz/dack | Fill (write) memory based on operand size (sz). Used with WRITE_MEM to fill large blocks of memory. An initial WRITE_MEM is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM commands write sequential operands. |
| FILL_MEM.sz_WS | Non-Intrusive | No | (0x13+4 x sz)/wd.sz/d/ss | Fill (write) memory based on operand size (sz) and report status. Used with WRITE_MEM{_WS} to fill large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM{_WS} commands write sequential operands. |
| GO | Active Background | Yes | 0x08/dack | Resume CPU user code execution |
| GO_UNTIL[2] | Active Background | Yes | 0x0C/dack | Go to user program. ACK is driven upon returning to active background mode. |
| NOP | Non-Intrusive | Yes | 0x00/dack | No operation |
| READ_Rn | Active Background | Yes | (0x60+CRN)/dack/rd32 | Read the requested CPU register |
| READ_MEM.sz | Non-Intrusive | Yes | (0x30+4 x sz)/ad24/dack/rd.sz | Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address |
| READ_MEM.sz_WS | Non-Intrusive | No | (0x31+4 x sz)/ad24/d/ss/rd.sz | Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address and report status |
| READ_DBGTB | Non-Intrusive | Yes | (0x07)/dack/rd32/dack/rd32 | Read 64-bits of DBG trace buffer |

**Table 5-8. BDC Command Summary (continued)**

| Command Mnemonic | Command Classification | ACK | Command Structure | Description |
|---|---|---|---|---|
| READ_SAME.sz | Non-Intrusive | Yes | (0x50+4 x sz)/dack/rd.sz | Read from location. An initial READ_MEM defines the address, subsequent READ_SAME reads return content of same address |
| READ_SAME.sz_WS | Non-Intrusive | No | (0x51+4 x sz)/d/ss/rd.sz | Read from location. An initial READ_MEM defines the address, subsequent READ_SAME reads return content of same address |
| READ_BDCCSR | Always Available | No | 0x2D/rd16 | Read the BDCCSR register |
| SYNC_PC | Non-Intrusive | Yes | 0x01/dack/rd24 | Read current PC |
| WRITE_MEM.sz | Non-Intrusive | Yes | (0x10+4 x sz)/ad24/wd.sz/dack | Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address |
| WRITE_MEM.sz_WS | Non-Intrusive | No | (0x11+4 x sz)/ad24/wd.sz/d/ss | Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address and report status |
| WRITE_Rn | Active Background | Yes | (0x40+CRN)/wd32/dack | Write the requested CPU register |
| WRITE_BDCCSR | Always Available | No | 0x0D/wd16 | Write the BDCCSR register |
| ERASE_FLASH | Always Available | No | 0x95/d | Mass erase internal flash |
| STEP1 (TRACE1) | Active Background | Yes | 0x09/dack | Execute one CPU command. |

1 The SYNC command is a special operation which does not have a command code.

2 The GO_UNTIL command is identical to the GO command if ACK is not enabled.

### 5.4.4.1 SYNC

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct speed to use for serial communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

1. Ensures that the BKGD pin is high for at least 4 cycles of the slowest possible BDCSI clock without reset asserted.
2. Drives the BKGD pin low for at least 128 cycles of the slowest possible BDCSI clock.
3. Drives BKGD high for a brief speed-up pulse to get a fast rise time. (This speedup pulse is typically one cycle of the host clock which is as fast as the maximum target BDCSI clock).
4. Removes all drive to the BKGD pin so it reverts to high impedance.
5. Listens to the BKGD pin for the sync response pulse.

Upon detecting the sync request from the host (which is a much longer low time than would ever occur during normal BDC communications), the target:

1. Discards any incomplete command
2. Waits for BKGD to return to a logic high.
3. Delays 16 cycles to allow the host to stop driving the high speed-up pulse.
4. Drives BKGD low for 128 BDCSI clock cycles.
5. Drives a 1-cycle high speed-up pulse to force a fast rise time on BKGD.
6. Removes all drive to the BKGD pin so it reverts to high impedance.
7. Clears the OVRRUN flag (if set).

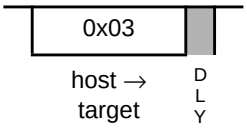
The host measures the low time of this 128-cycle SYNC response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the serial protocol can easily tolerate this speed error.

If the SYNC request is detected by the target, any partially executed command is discarded. This is referred to as a soft-reset, equivalent to a timeout in the serial communication. After the SYNC response, the target interprets the next negative edge (issued by the host) as the start of a new BDC command or the start of new SYNC request.

A SYNC command can also be used to abort a pending ACK pulse. This is explained in Section 5.4.8.

### 5.4.4.2 ACK_DISABLE

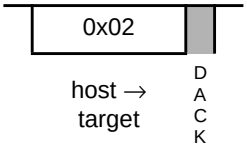
**Disable host/target handshake protocol** **Always Available**




Disables the serial communication handshake protocol. The subsequent commands, issued after the ACK_DISABLE command, do not execute the hardware handshake protocol. This command is not followed by an ACK pulse.

### 5.4.4.3 ACK_ENABLE

**Enable host/target handshake protocol** **Always Available**



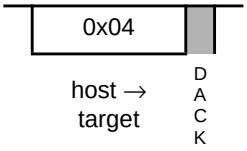

Enables the hardware handshake protocol in the serial communication. The hardware handshake is implemented by an acknowledge (ACK) pulse issued by the target MCU in response to a host command. The ACK_ENABLE command is interpreted and executed in the BDC logic without the need to interface

with the CPU. An ACK pulse is issued by the target device after this command is executed. This command can be used by the host to evaluate if the target supports the hardware handshake protocol. If the target supports the hardware handshake protocol, subsequent commands are enabled to execute the hardware handshake protocol, otherwise this command is ignored by the target. Table 5-8 indicates which commands support the ACK hardware handshake protocol.

For additional information about the hardware handshake protocol, refer to Section 5.4.7," and Section 5.4.8."

### 5.4.4.4 BACKGROUND

**Enter active background mode (if enabled)** **Non-intrusive**




Provided ENBDC is set, the BACKGROUND command causes the target MCU to enter active BDM as soon as the current CPU instruction finishes. If ENBDC is cleared, the BACKGROUND command is ignored.

A delay of 16 BDCSI clock cycles is required after the BACKGROUND command to allow the target MCU to finish its current CPU instruction and enter active background mode before a new BDC command can be accepted.

The host debugger must set ENBDC before attempting to send the BACKGROUND command the first time. Normally the host sets ENBDC once at the beginning of a debug session or after a target system reset. During debugging, the host uses GO commands to move from active BDM to application program execution and uses the BACKGROUND command or DBG breakpoints to return to active BDM.

A BACKGROUND command issued during stop or wait modes cannot immediately force active BDM because the WAI instruction does not end until an interrupt occurs. For the detailed mode dependency description refer to Section 5.1.3.3.

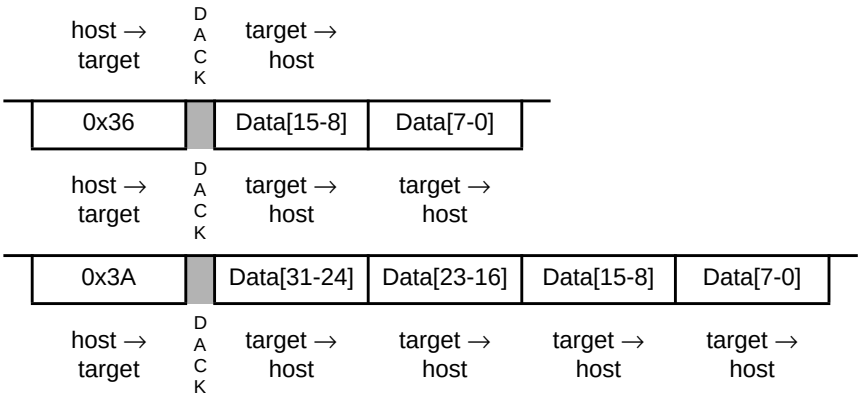
The host can recognize this pending BDM request condition because both NORESP and WAIT are set, but BDMACT is clear. Whilst in wait mode, with the pending BDM request, non-intrusive BDC commands are allowed.

### 5.4.4.5 DUMP_MEM.sz, DUMP_MEM.sz_WS

**DUMP_MEM.sz**

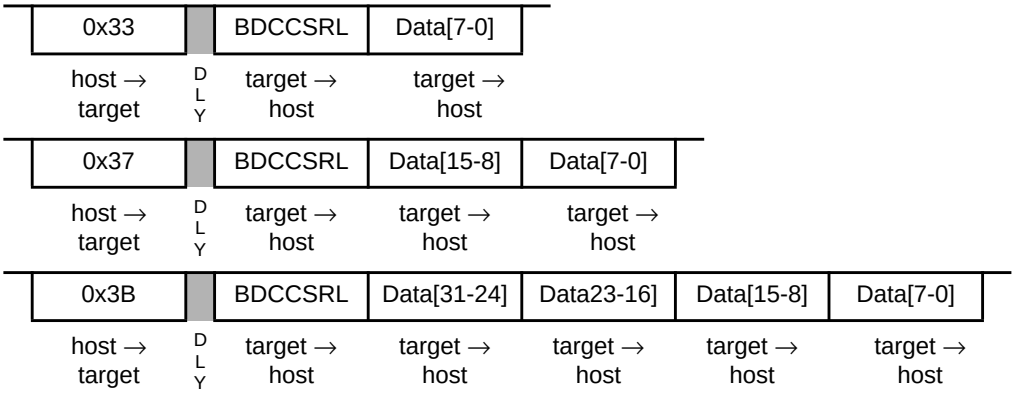
**Read memory specified by debug address register, then increment address** **Non-intrusive**

**DUMP_MEM.sz**




**DUMP_MEM.sz_WS**

**Read memory specified by debug address register with status, then increment address** **Non-intrusive**




DUMP_MEM{_WS} is used with the READ_MEM{_WS} command to access large blocks of memory. An initial READ_MEM{_WS} is executed to set-up the starting address of the block and to retrieve the first result. The DUMP_MEM{_WS} command retrieves subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP_MEM{_WS} commands use this address, perform the memory read, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the BDCCSRL status byte is returned before the read data. This status byte reflects the state after the memory read was performed. If enabled, an ACK pulse is driven before the data bytes are transmitted. The effect of the access size and alignment on the next address to be accessed is explained in more detail in Section 5.4.5.2".

**NOTE**

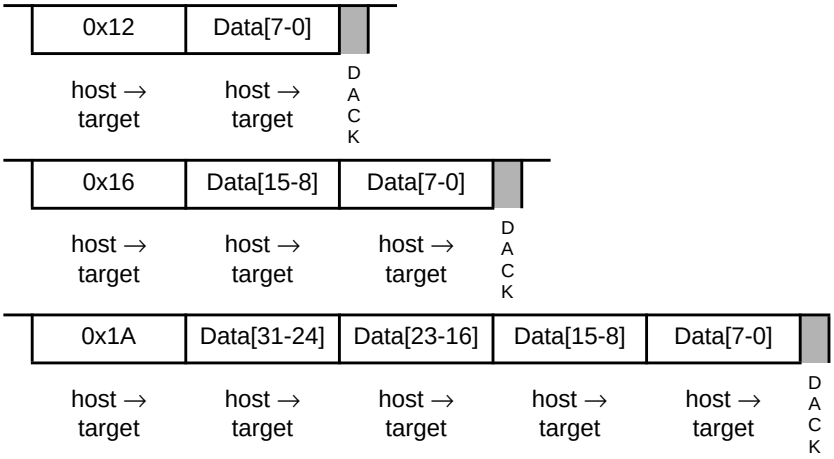
DUMP_MEM{_WS} is a valid command only when preceded by SYNC, NOP, READ_MEM{_WS}, or another DUMP_MEM{_WS} command. Otherwise, an illegal command response is returned, setting the ILLCMD bit. NOP can be used for inter-command padding without corrupting the address pointer.

The size field (sz) is examined each time a DUMP_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the DUMP_MEM.B{_WS}, DUMP_MEM.W{_WS} and DUMP_MEM.L{_WS} commands.

### 5.4.4.6 FILL_MEM.sz, FILL_MEM.sz_WS

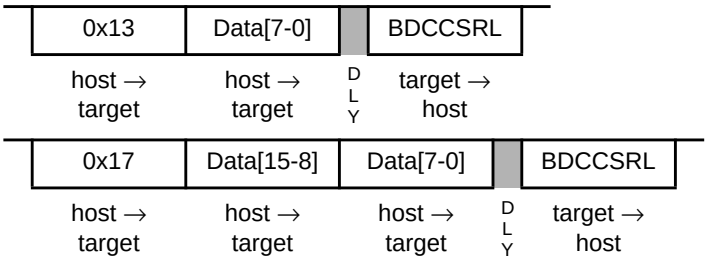
**FILL_MEM.sz**

**Write memory specified by debug address register, then increment address** **Non-intrusive**

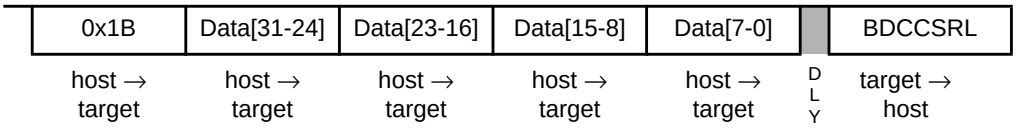



**FILL_MEM.sz_WS**

**Write memory specified by debug address register with status, then increment address** **Non-intrusive**



**FILL_MEM.sz_WS**




FILL_MEM{_WS} is used with the WRITE_MEM{_WS} command to access large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and write the first datum. If an initial WRITE_MEM{_WS} is not executed before the first FILL_MEM{_WS}, an illegal command response is returned. The FILL_MEM{_WS} command stores subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent FILL_MEM{_WS} commands use this address, perform the memory write, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the BDCCSRL status byte is returned after the write data. This status byte reflects the state after the memory write was performed. If enabled an ACK pulse is generated after the internal write access has been completed or aborted. The effect of the access size and alignment on the next address to be accessed is explained in more detail in Section 5.4.5.2”

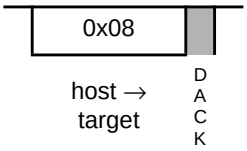
**NOTE**

FILL_MEM{_WS} is a valid command only when preceded by SYNC, NOP, WRITE_MEM{_WS}, or another FILL_MEM{_WS} command. Otherwise, an illegal command response is returned, setting the ILLCMD bit. NOP can be used for inter command padding without corrupting the address pointer.

The size field (sz) is examined each time a FILL_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the FILL_MEM.B{_WS}, FILL_MEM.W{_WS} and FILL_MEM.L{_WS} commands.

### 5.4.4.7 GO

**Go** **Non-intrusive**

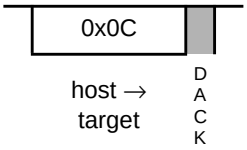



This command is used to exit active BDM and begin (or resume) execution of CPU application code. The CPU pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC. If any register (such as the PC) is altered by a BDC command whilst in BDM, the updated value is used when prefetching resumes. If enabled, an ACK is driven on exiting active BDM.

If a GO command is issued whilst the BDM is inactive, an illegal command response is returned and the ILLCMD bit is set.

### 5.4.4.8 GO_UNTIL

**Go Until** **Active Background**




This command is used to exit active BDM and begin (or resume) execution of application code. The CPU pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC. If any register (such as the PC) is altered by a BDC command whilst in BDM, the updated value is used when prefetching resumes.

After resuming application code execution, if ACK is enabled, the BDC awaits a return to active BDM before driving an ACK pulse. timeouts do not apply when awaiting a GO_UNTIL command ACK.

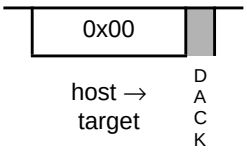
If a GO_UNTIL is not acknowledged then a SYNC command must be issued to end the pending GO_UNTIL.

If a GO_UNTIL command is issued whilst BDM is inactive, an illegal command response is returned and the ILLCMD bit is set.

If ACK handshaking is disabled, the GO_UNTIL command is identical to the GO command.

### 5.4.4.9 NOP

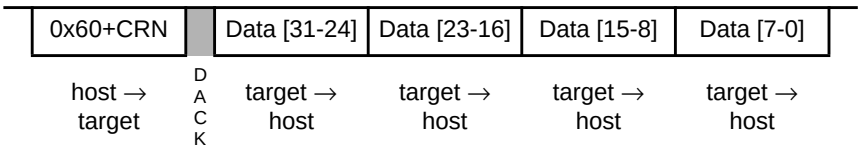
**No operation** **Active Background**




NOP performs no operation and may be used as a null command where required.

### 5.4.4.10 READ_Rn

**Read CPU register** **Active Background**




This command reads the selected CPU registers and returns the 32-bit result. Accesses to CPU registers are always 32-bits wide, regardless of implemented register width. Bytes that are not implemented return

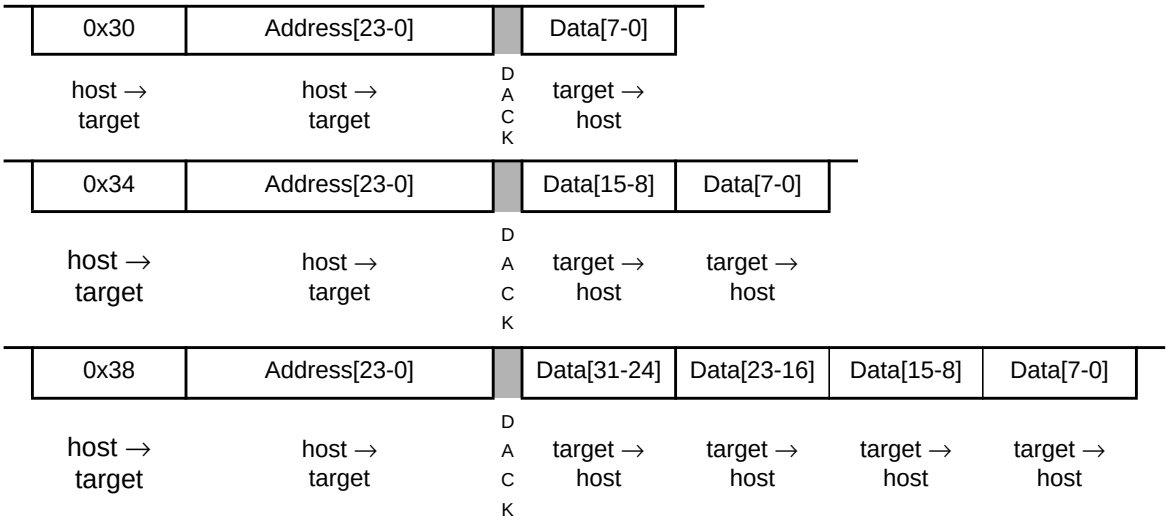
zero. The register is addressed through the CPU register number (CRN). See Section 5.4.5.1 for the CRN address decoding. If enabled, an ACK pulse is driven before the data bytes are transmitted.

If the device is not in active BDM, this command is illegal, the ILLCMD bit is set and no access is performed.

### 5.4.4.11 READ_MEM.sz, READ_MEM.sz_WS

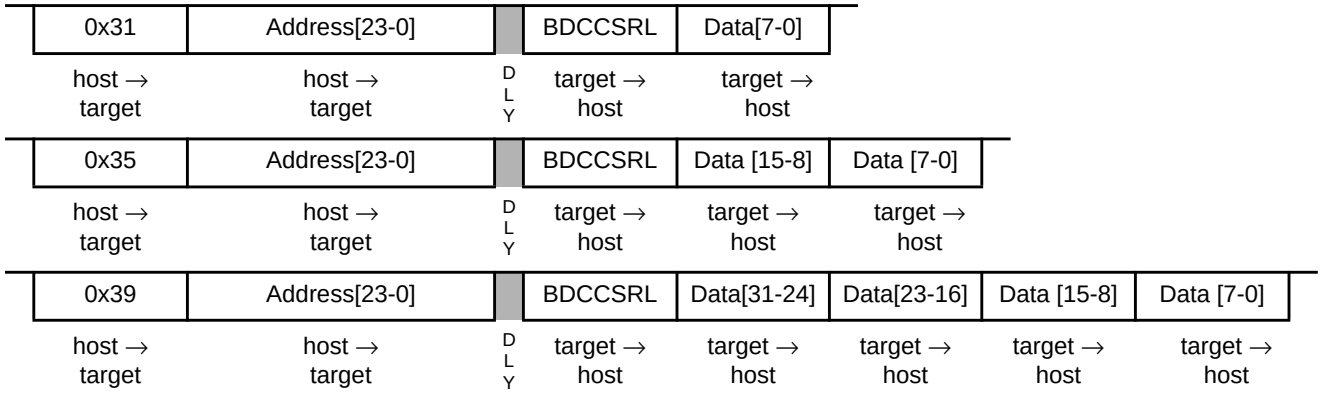
**READ_MEM.sz**

**Read memory at the specified address** **Non-intrusive**




**READ_MEM.sz_WS**

**Read memory at the specified address with status** **Non-intrusive**



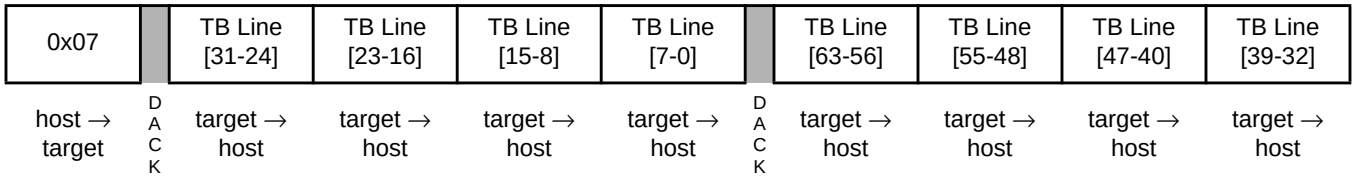

Read data at the specified memory address. The address is transmitted as three 8-bit packets (msb to lsb) immediately after the command.

The hardware forces low-order address bits to zero longword accesses to ensure these accesses are on 0-modulo-size alignments. Byte alignment details are described in Section 5.4.5.2". If the with-status option is specified, the BDCCSR status byte is returned before the read data. This status byte reflects the state after the memory read was performed. If enabled, an ACK pulse is driven before the data bytes are transmitted.

The examples show the READ_MEM.B{_WS}, READ_MEM.W{_WS} and READ_MEM.L{_WS} commands.

### 5.4.4.12 READ_DBGTB

**Read DBG trace buffer** **Non-intrusive**



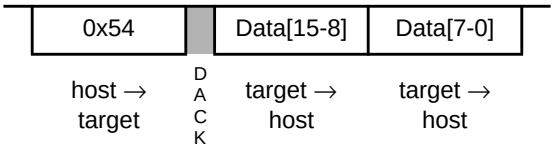

This command is only available on devices, where the DBG module includes a trace buffer. Attempted use of this command on devices without a traace buffer return 0x00.

Read 64 bits from the DBG trace buffer. Refer to the DBG module description for more detailed information. If enabled an ACK pulse is generated before each 32-bit longword is ready to be read by the host. After issuing the first ACK a timeout is still possible whilst accessing the second 32-bit longword, since this requires separate internal accesses. The first 32-bit longword corresponds to trace buffer line bits[31:0]; the second to trace buffer line bits[63:32]. If ACK handshaking is disabled, the host must wait 16 clock cycles (DLY) after completing the first 32-bit read before starting the second 32-bit read.

### 5.4.4.13 READ_SAME.sz, READ_SAME.sz_WS

**READ_SAME**

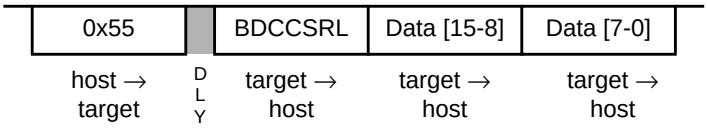
**Read same location specified by previous READ_MEM{_WS}** **Non-intrusive**




**READ_SAME_WS**

**Read same location specified by previous READ_MEM{_WS}** **Non-intrusive**

**READ_SAME_WS**



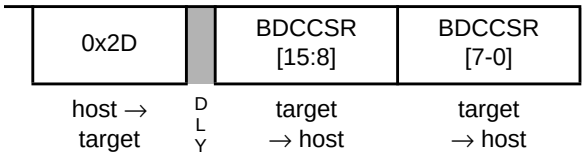

Read from location defined by the previous READ_MEM. The previous READ_MEM command defines the address, subsequent READ_SAME commands return contents of same address. The example shows the sequence for reading a 16-bit word size. Byte alignment details are described in Section 5.4.5.2". If enabled, an ACK pulse is driven before the data bytes are transmitted.

**NOTE**

READ_SAME{_WS} is a valid command only when preceded by SYNC, NOP, READ_MEM{_WS}, or another READ_SAME{_WS} command. Otherwise, an illegal command response is returned, setting the ILLCMD bit. NOP can be used for inter-command padding without corrupting the address pointer.

### 5.4.4.14 READ_BDCCSR

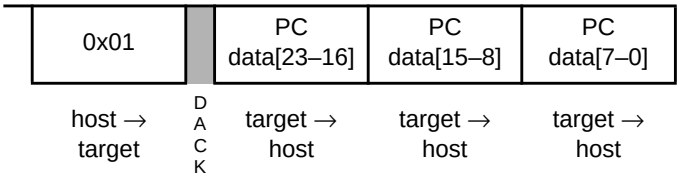
**Read BDCCSR Status Register** **Always Available**




Read the BDCCSR status register. This command can be executed in any mode.

### 5.4.4.15 SYNC_PC

**Sample current PC** **Non-intrusive**



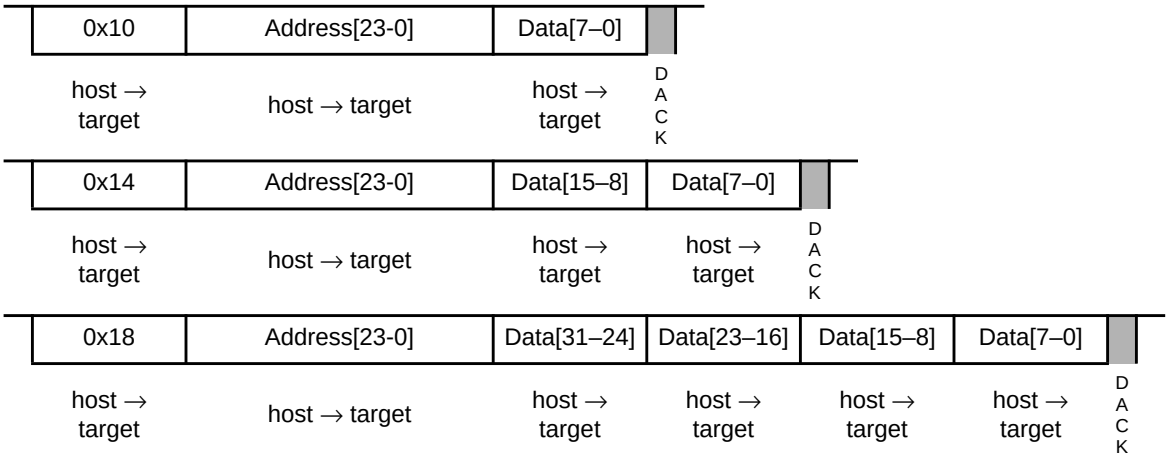

This command returns the 24-bit CPU PC value to the host. Unsuccessful SYNC_PC accesses return 0xEE for each byte. If enabled, an ACK pulse is driven before the data bytes are transmitted. The value of 0xEE is returned if a timeout occurs, whereby NORESP is set. This can occur if the CPU is executing the WAI instruction, or the STOP instruction with BDCCIS clear, or if a CPU access is delayed by EWAIT. If the CPU is executing the STOP instruction and BDCCIS is set, then SYNC_PC returns the PC address of the instruction following STOP in the code listing.

This command can be used to dynamically access the PC for performance monitoring as the execution of this command is considerably less intrusive to the real-time operation of an application than a BACKGROUND/read-PC/GO command sequence. Whilst the BDC is not in active BDM, SYNC_PC returns the PC address of the instruction currently being executed by the CPU. In active BDM, SYNC_PC returns the address of the next instruction to be executed on returning from active BDM. Thus following a write to the PC in active BDM, a SYNC_PC returns that written value.

### 5.4.4.16 WRITE_MEM.sz, WRITE_MEM.sz_WS

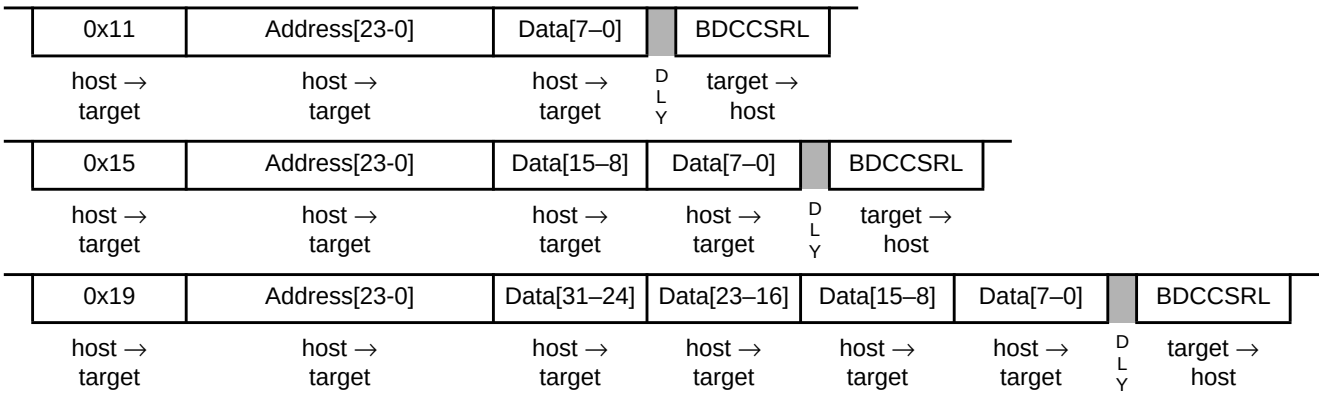
**WRITE_MEM.sz**

**Write memory at the specified address** **Non-intrusive**




**WRITE_MEM.sz_WS**

**Write memory at the specified address with status** **Non-intrusive**




Write data to the specified memory address. The address is transmitted as three 8-bit packets (msb to lsb) immediately after the command.

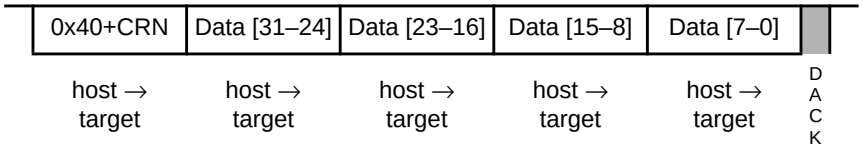
If the with-status option is specified, the status byte contained in BDCCSRL is returned after the write data. This status byte reflects the state after the memory write was performed. The examples show the

WRITE_MEM.B{_WS}, WRITE_MEM.W{_WS}, and WRITE_MEM.L{_WS} commands. If enabled an ACK pulse is generated after the internal write access has been completed or aborted.

The hardware forces low-order address bits to zero longword accesses to ensure these accesses are on 0-modulo-size alignments. Byte alignment details are described in Section 5.4.5.2".

### 5.4.4.17 WRITE_Rn

**Write general-purpose CPU register** **Active Background**

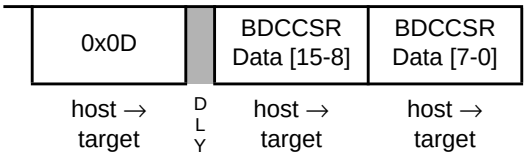



If the device is in active BDM, this command writes the 32-bit operand to the selected CPU general-purpose register. See Section 5.4.5.1 for the CRN details. Accesses to CPU registers are always 32-bits wide, regardless of implemented register width. If enabled an ACK pulse is generated after the internal write access has been completed or aborted.

If the device is not in active BDM, this command is rejected as an illegal operation, the ILLCMD bit is set and no operation is performed.

### 5.4.4.18 WRITE_BDCCSR

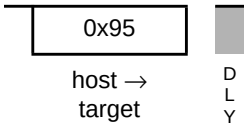
**Write BDCCSR** **Always Available**




16-bit write to the BDCCSR register. No ACK pulse is generated. Writing to this register can be used to configure control bits or clear flag bits. Refer to the register bit descriptions.

### 5.4.4.19 ERASE_FLASH

**Erase FLASH** **Always Available**




Mass erase the internal flash. This command can always be issued. On receiving this command twice in succession, the BDC sets the ERASE bit in BDCCSR and requests a flash mass erase. Any other BDC

command following a single ERASE_FLASH initializes the sequence, such that thereafter the ERASE_FLASH must be applied twice in succession to request a mass erase. If 512 BDCSI clock cycles elapse between the consecutive ERASE_FLASH commands then a timeout occurs, which forces a soft reset and initializes the sequence. The ERASE bit is cleared when the mass erase sequence has been completed. No ACK is driven.

During the mass erase operation, which takes many clock cycles, the command status is indicated by the ERASE bit in BDCCSR. Whilst a mass erase operation is ongoing, Always-available commands can be issued. This allows the status of the erase operation to be polled by reading BDCCSR to determine when the operation is finished.

The status of the flash array can be verified by subsequently reading the flash error flags to determine if the erase completed successfully.

ERASE_FLASH can be aborted by a SYNC pulse forcing a soft reset.

**NOTE: Device Bus Frequency Considerations**

The ERASE_FLASH command requires the default device bus clock frequency after reset. Thus the bus clock frequency must not be changed following reset before issuing an ERASE_FLASH command.

#### 5.4.4.20 STEP1

This command is used to step through application code. In active BDM this command executes the next CPU instruction in application code. If enabled an ACK is driven.

If a STEP1 command is issued and the CPU is not halted, the command is ignored.

Using STEP1 to step through a CPU WAI instruction is explained in Section 5.1.3.3.2.

### 5.4.5 BDC Access Of Internal Resources

Unsuccessful read accesses of internal resources return a value of 0xEE for each data byte. This enables a debugger to recognize a potential error, even if neither the ACK handshaking protocol nor a status command is currently being executed. The value of 0xEE is returned in the following cases.

- Illegal address access, whereby ILLACC is set
- Invalid READ_SAME or DUMP_MEM sequence
- Invalid READ_Rn command (BDM inactive or CRN incorrect)
- Internal resource read with timeout, whereby NORESP is set

### 5.4.5.1 BDC Access Of CPU Registers

The CRN field of the READ_Rn and WRITE_Rn commands contains a pointer to the CPU registers. The mapping of CRN to CPU registers is shown in Table 5-9. Accesses to CPU registers are always 32-bits wide, regardless of implemented register width. This means that the BDC data transmission for these commands is 32-bits long. The valid bits of the transfer are listed in the Valid Data Bits column. The other bits of the transmission are redundant.

Attempted accesses of CPU registers using a CRN of 0xD,0xE or 0xF is invalid, returning the value 0xEE for each byte and setting the ILLACC bit.

**Table 5-9. CPU Register Number (CRN) Mapping**

| CPU Register | Valid Data Bits | Command | Opcode | Command | Opcode |
|---|---|---|---|---|---|
| D0 | [7:0] | WRITE_D0 | 0x40 | READ_D0 | 0x60 |
| D1 | [7:0] | WRITE_D1 | 0x41 | READ_D1 | 0x61 |
| D2 | [15:0] | WRITE_D2 | 0x42 | READ_D2 | 0x62 |
| D3 | [15:0] | WRITE_D3 | 0x43 | READ_D3 | 0x63 |
| D4 | [15:0] | WRITE_D4 | 0x44 | READ_D4 | 0x64 |
| D5 | [15:0] | WRITE_D5 | 0x45 | READ_D5 | 0x65 |
| D6 | [31:0] | WRITE_D6 | 0x46 | READ_D6 | 0x66 |
| D7 | [31:0] | WRITE_D7 | 0x47 | READ_D7 | 0x67 |
| X | [23:0] | WRITE_X | 0x48 | READ_X | 0x68 |
| Y | [23:0] | WRITE_Y | 0x49 | READ_Y | 0x69 |
| SP | [23:0] | WRITE_SP | 0x4A | READ_SP | 0x6A |
| PC | [23:0] | WRITE_PC | 0x4B | READ_PC | 0x6B |
| CCR | [15:0] | WRITE_CCR | 0x4C | READ_CCR | 0x6C |

### 5.4.5.2 BDC Access Of Device Memory Mapped Resources

The device memory map is accessed using READ_MEM, DUMP_MEM, WRITE_MEM, FILL_MEM and READ_SAME, which support different access sizes, as explained in the command descriptions.

When an unimplemented command occurs during a DUMP_MEM, FILL_MEM or READ_SAME sequence, then that sequence is ended.

Illegal read accesses return a value of 0xEE for each byte. After an illegal access FILL_MEM and READ_SAME commands are not valid, and it is necessary to restart the internal access sequence with READ_MEM or WRITE_MEM. An illegal access does not break a DUMP_MEM sequence. After read accesses that cause the RDINV bit to be set, DUMP_MEM and READ_SAME commands are valid, it is not necessary to restart the access sequence with a READ_MEM.

The hardware forces low-order address bits to zero for longword accesses to ensure these accesses are realigned to 0-modulo-size alignments.

Word accesses map to 2-bytes from within a 4-byte field as shown in Table 5-10. Thus if address bits [1:0] are both logic “1” the access is realigned so that it does not straddle the 4-byte boundary but accesses data from within the addressed 4-byte field.

**Table 5-10. Field Location to Byte Access Mapping**

| Address[1:0] | Access Size | 00 | 01 | 10 | 11 | Note |
|---|---|---|---|---|---|---|
| 00 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | |
| 01 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | Realigned |
| 10 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | Realigned |
| 11 | 32-bit | Data[31:24] | Data[23:16] | Data [15:8] | Data [7:0] | Realigned |
| 00 | 16-bit | Data [15:8] | Data [7:0] | | | |
| 01 | 16-bit | | Data [15:8] | Data [7:0] | | |
| 10 | 16-bit | | | Data [15:8] | Data [7:0] | |
| 11 | 16-bit | | | Data [15:8] | Data [7:0] | Realigned |
| 00 | 8-bit | Data [7:0] | | | | |
| 01 | 8-bit | | Data [7:0] | | | |
| 10 | 8-bit | | | Data [7:0] | | |
| 11 | 8-bit | | | | Data [7:0] | |
| | | | Denotes byte that is not transmitted | | | |

#### 5.4.5.2.1 FILL_MEM and DUMP_MEM Increments and Alignment

FILL_MEM and DUMP_MEM increment the previously accessed address by the previous access size to calculate the address of the current access. On misaligned longword accesses, the address bits [1:0] are forced to zero, therefore the following FILL_MEM or DUMP_MEM increment to the first address in the next 4-byte field. This is shown in Table 5-11, the address of the first DUMP_MEM.32 following READ_MEM.32 being calculated from 0x004000+4.

When misaligned word accesses are realigned, then the original address (not the realigned address) is incremented for the following FILL_MEM, DUMP_MEM command.

Misaligned word accesses can cause the same locations to be read twice as shown in rows 6 and 7. The hardware ensures alignment at an attempted misaligned word access across a 4-byte boundary, as shown in row 7. The following word access in row 8 continues from the realigned address of row 7.

**Table 5-11. Consecutive Accesses With Variable Size**

| Row | Command | Address | Address[1:0] | 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|---|---|
| 1 | READ_MEM.32 | 0x004003 | 11 | Accessed | Accessed | Accessed | Accessed |
| 2 | DUMP_MEM.32 | 0x004004 | 00 | Accessed | Accessed | Accessed | Accessed |
| 3 | DUMP_MEM.16 | 0x004008 | 00 | Accessed | Accessed | | |
| 4 | DUMP_MEM.16 | 0x00400A | 10 | | | Accessed | Accessed |
| 5 | DUMP_MEM.08 | 0x00400C | 00 | Accessed | | | |
| 6 | DUMP_MEM.16 | 0x00400D | 01 | | Accessed | Accessed | |
| 7 | DUMP_MEM.16 | 0x00400E | 10 | | | Accessed | Accessed |
| 8 | DUMP_MEM.16 | 0x004010 | 01 | Accessed | Accessed | | |

#### 5.4.5.2.2 READ_SAME Effects Of Variable Access Size

READ_SAME uses the unadjusted address given in the previous READ_MEM command as a base address for subsequent READ_SAME commands. When the READ_MEM and READ_SAME size parameters differ then READ_SAME uses the original base address buts aligns 32-bit and 16-bit accesses, where those accesses would otherwise cross the aligned 4-byte boundary. Table 5-12 shows some examples of this.

**Table 5-12. Consecutive READ_SAME Accesses With Variable Size**

| Row | Command | Base Address | 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|---|
| 1 | READ_MEM.32 | 0x004003 | Accessed | Accessed | Accessed | Accessed |
| 2 | READ_SAME.32 | — | Accessed | Accessed | Accessed | Accessed |
| 3 | READ_SAME.16 | — | | | Accessed | Accessed |
| 4 | READ_SAME.08 | — | | | | Accessed |
| 5 | READ_MEM.08 | 0x004000 | Accessed | | | |
| 6 | READ_SAME.08 | — | Accessed | | | |
| 7 | READ_SAME.16 | — | Accessed | Accessed | | |
| 8 | READ_SAME.32 | — | Accessed | Accessed | Accessed | Accessed |
| 9 | READ_MEM.08 | 0x004002 | | | Accessed | |
| 10 | READ_SAME.08 | — | | | Accessed | |
| 11 | READ_SAME.16 | — | | | Accessed | Accessed |
| 12 | READ_SAME.32 | — | Accessed | Accessed | Accessed | Accessed |
| 13 | READ_MEM.08 | 0x004003 | | | | Accessed |
| 14 | READ_SAME.08 | — | | | | Accessed |
| 15 | READ_SAME.16 | — | | | Accessed | Accessed |
| 16 | READ_SAME.32 | — | Accessed | Accessed | Accessed | Accessed |
| 17 | READ_MEM.16 | 0x004001 | | Accessed | Accessed | |
| 18 | READ_SAME.08 | — | | Accessed | | |
| 19 | READ_SAME.16 | — | | Accessed | Accessed | |
| 20 | READ_SAME.32 | — | Accessed | Accessed | Accessed | Accessed |
| 21 | READ_MEM.16 | 0x004003 | | | Accessed | Accessed |
| 22 | READ_SAME.08 | — | | | | Accessed |
| 23 | READ_SAME.16 | — | | | Accessed | Accessed |
| 24 | READ_SAME.32 | — | Accessed | Accessed | Accessed | Accessed |

## 5.4.6 BDC Serial Interface

The BDC communicates with external devices serially via the BKGD pin. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the BDC.

The BDC serial interface uses an internal clock source, selected by the CLKSW bit in the BDCCSR register. This clock is referred to as the target clock in the following explanation.

The BDC serial interface uses a clocking scheme in which the external host generates a falling edge on the BKGD pin to indicate the start of each bit time. This falling edge is sent for every bit whether data is transmitted or received. Data is transferred most significant bit (MSB) first at 16 target clock cycles per bit. The interface times out if during a command 512 clock cycles occur between falling edges from the host. The timeout forces the current command to be discarded.

The BKGD pin is a pseudo open-drain pin and has a weak on-chip active pull-up that is enabled at all times. It is assumed that there is an external pull-up and that drivers connected to BKGD do not typically drive the high level. Since R-C rise time could be unacceptably long, the target system and host provide brief drive-high (speedup) pulses to drive BKGD to a logic 1. The source of this speedup pulse is the host for transmit cases and the target for receive cases.

The timing for host-to-target is shown in Figure 5-6 and that of target-to-host in Figure 5-7 and Figure 5-8. All cases begin when the host drives the BKGD pin low to generate a falling edge. Since the host and target operate from separate clocks, it can take the target up to one full clock cycle to recognize this edge; this synchronization uncertainty is illustrated in Figure 5-6. The target measures delays from this perceived start of the bit time while the host measures delays from the point it actually drove BKGD low to start the bit up to one target clock cycle earlier. Synchronization between the host and target is established in this manner at the start of every bit time.

Figure 5-6 shows an external host transmitting a logic 1 and transmitting a logic 0 to the BKGD pin of a target system. The host is asynchronous to the target, so there is up to a one clock-cycle delay from the host-generated falling edge to where the target recognizes this edge as the beginning of the bit time. Ten target clock cycles later, the target senses the bit level on the BKGD pin. Internal glitch detect logic requires the pin be driven high no later than eight target clock cycles after the falling edge for a logic 1 transmission.

Since the host drives the high speedup pulses in these two cases, the rising edges look like digitally driven signals.

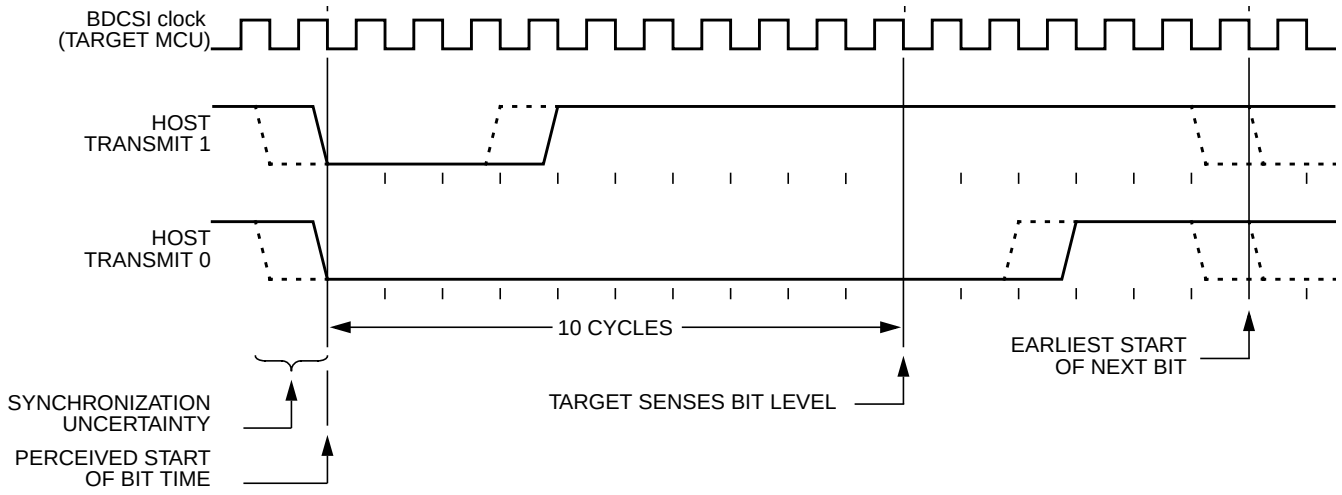



**Figure 5-6. BDC Host-to-Target Serial Bit Timing**

Figure 5-7 shows the host receiving a logic 1 from the target system. The host holds the BKGD pin low long enough for the target to recognize it (at least two target clock cycles). The host must release the low

drive at the latest after 6 clock cycles, before the target drives a brief high speedup pulse seven target clock cycles after the perceived start of the bit time. The host should sample the bit level about 10 target clock cycles after it started the bit time.

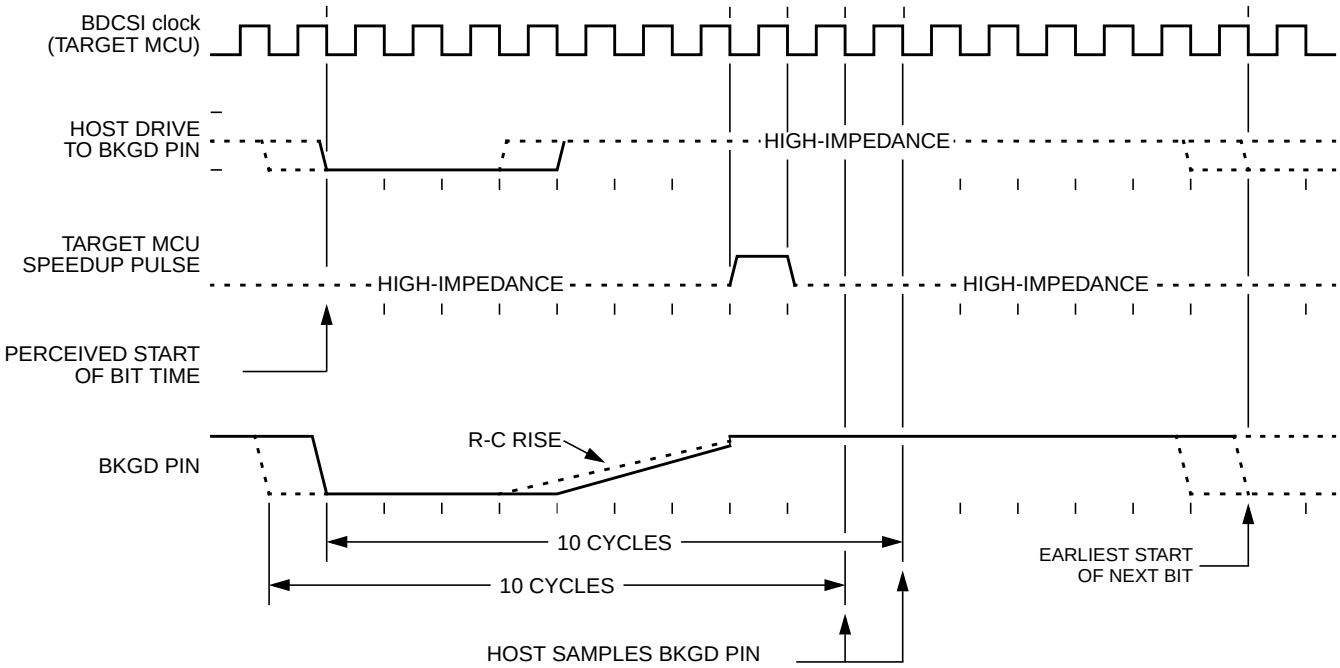



**Figure 5-7. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 5-8 shows the host receiving a logic 0 from the target. The host initiates the bit time but the target finishes it. Since the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 target clock cycles then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 target clock cycles after starting the bit time.

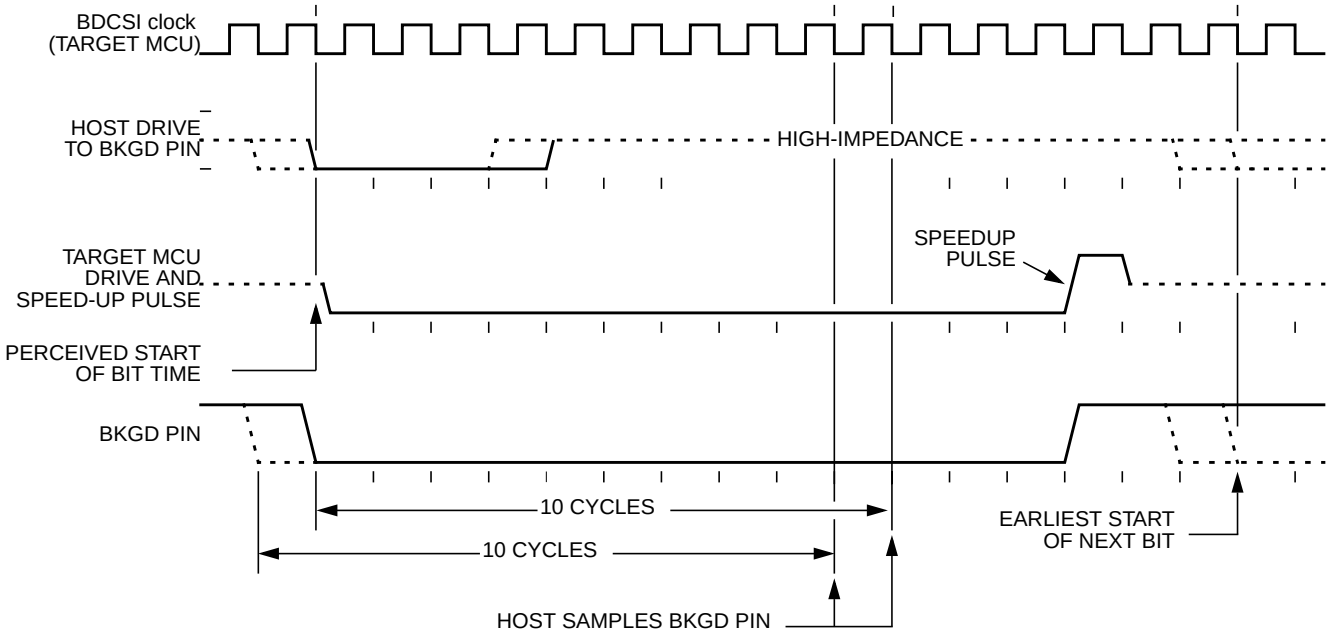



**Figure 5-8. BDC Target-to-Host Serial Bit Timing (Logic 0)**

## 5.4.7 Serial Interface Hardware Handshake (ACK Pulse) Protocol

BDC commands are processed internally at the device core clock rate. Since the BDCSI clock can be asynchronous relative to the bus frequency, a handshake protocol is provided so the host can determine when an issued command has been executed. This section describes the hardware handshake protocol.

The hardware handshake protocol signals to the host controller when a BDC command has been executed by the target. This protocol is implemented by a low pulse (16 BDCSI clock cycles) followed by a brief speedup pulse on the BKGD pin, generated by the target MCU when a command, issued by the host, has been successfully executed (see Figure 5-9). This pulse is referred to as the ACK pulse. After the ACK pulse has finished, the host can start the bit retrieval if the last issued command was a read command, or start a new command if the last command was a write command or a control command.

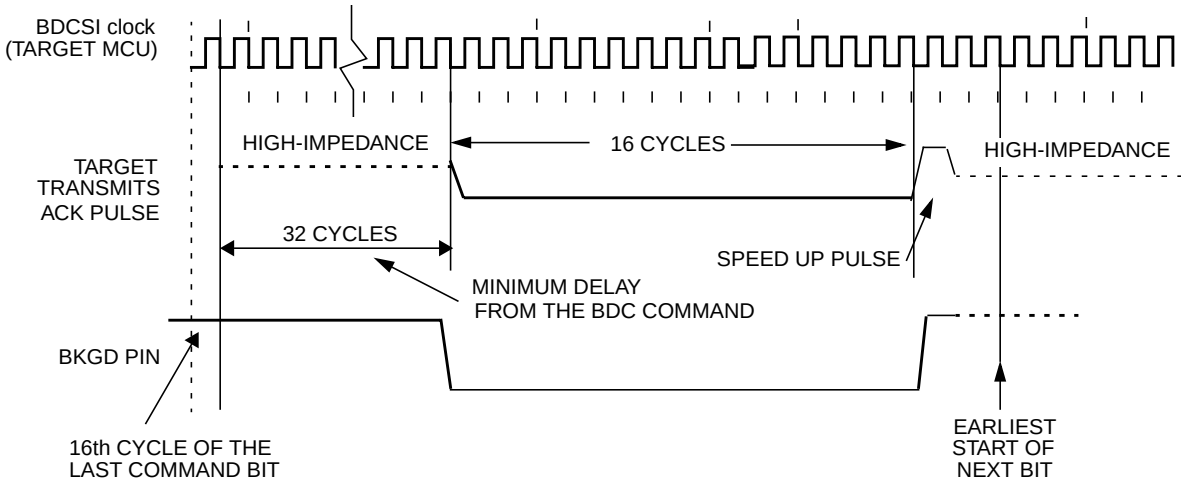



**Figure 5-9. Target Acknowledge Pulse (ACK)**

The handshake protocol is enabled by the ACK_ENABLE command. The BDC sends an ACK pulse when the ACK_ENABLE command has been completed. This feature can be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol.

Unlike the normal bit transfer, where the host initiates the transmission by issuing a negative edge on the BKGD pin, the serial interface ACK handshake pulse is initiated by the target MCU by issuing a negative edge on the BKGD pin. Figure 5-9 specifies the timing when the BKGD pin is being driven. The host must follow this timing constraint in order to avoid the risk of an electrical conflict at the BKGD pin.

When the handshake protocol is enabled, the STEAL bit in BDCCSR selects if bus cycle stealing is used to gain immediate access. If STEAL is cleared, the BDC is configured for low priority bus access using free cycles, without stealing cycles. This guarantees that BDC accesses remain truly non-intrusive to not affect the system timing during debugging. If STEAL is set, the BDC gains immediate access, if necessary stealing an internal bus cycle.

**NOTE**

If bus steals are disabled then a loop with no free cycles cannot allow access. In this case the host must recognize repeated NORESP messages and then issue a BACKGROUND command to stop the target and access the data.

Figure 5-10 shows the ACK handshake protocol without steal in a command level timing diagram. The READ_MEM.B command is used as an example. First, the 8-bit command code is sent by the host, followed by the address of the memory location to be read. The target BDC decodes the command. Then an internal access is requested by the BDC. When a free bus cycle occurs the READ_MEM.B operation is carried out. If no free cycle occurs within 512 core clock cycles then the access is aborted, the NORESP flag is set and the target generates a Long-ACK pulse.

Having retrieved the data, the BDC issues an ACK pulse to the host controller, indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the data read part of the command.

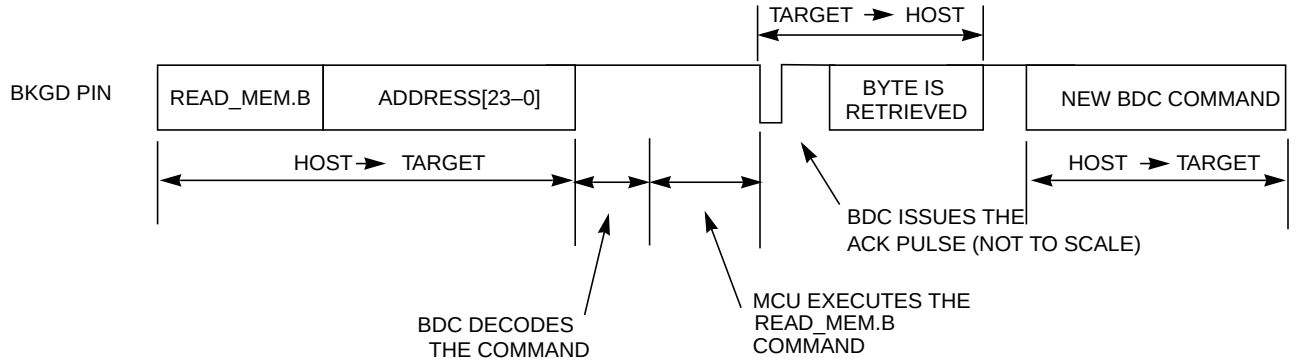



**Figure 5-10. Handshake Protocol at Command Level**

Alternatively, setting the STEAL bit configures the handshake protocol to make an immediate internal access, independent of free bus cycles.

The ACK handshake protocol does not support nested ACK pulses. If a BDC command is not acknowledged by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDC command. The host can decide to abort any possible pending ACK pulse in order to be sure a new command can be issued. Therefore, the protocol provides a mechanism in which a command, and its corresponding ACK, can be aborted.

Commands With-Status do not generate an ACK, thus if ACK is enabled and a With-Status command is issued, the host must use the 512 cycle timeout to calculate when the data is ready for retrieval.

#### 5.4.7.1 Long-ACK Hardware Handshake Protocol

If a command results in an error condition, whereby a BDCCSRL flag is set, then the target generates a “Long-ACK” low pulse of 64 BDCSI clock cycles, followed by a brief speed pulse. This indicates to the host that an error has occurred. The host can subsequently read BDCCSR to determine the type of error. Whether normal ACK or Long-ACK, the ACK pulse is not issued earlier than 32 BDCSI clock cycles after the BDC command was issued. The end of the BDC command is assumed to be the 16th BDCSI clock cycle of the last bit. The 32 cycle minimum delay differs from the 16 cycle delay time with ACK disabled.

If a BDC access request does not gain access within 512 core clock cycles, the request is aborted, the NORESP flag is set and a Long-ACK pulse is transmitted to indicate an error case.

Following a STOP or WAI instruction, if the BDC is enabled, the first ACK, following stop or wait mode entry is a long ACK to indicate an exception.

### 5.4.8 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. To abort a command that has not responded with an ACK pulse, the host controller generates a sync request (by driving BKGD low for at least 128 BDCSI clock cycles and then driving it high for one BDCSI clock cycle as a speedup pulse). By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, see Section 5.4.4.1”, and assumes that the pending command and therefore the related ACK pulse are being aborted. After the SYNC protocol has been completed the host is free to issue new BDC commands.

The host can issue a SYNC close to the 128 clock cycles length, providing a small overhead on the pulse length to assure the sync pulse is not misinterpreted by the target. See Section 5.4.4.1”.

Figure 5-11 shows a SYNC command being issued after a READ_MEM, which aborts the READ_MEM command. Note that, after the command is aborted a new command is issued by the host.

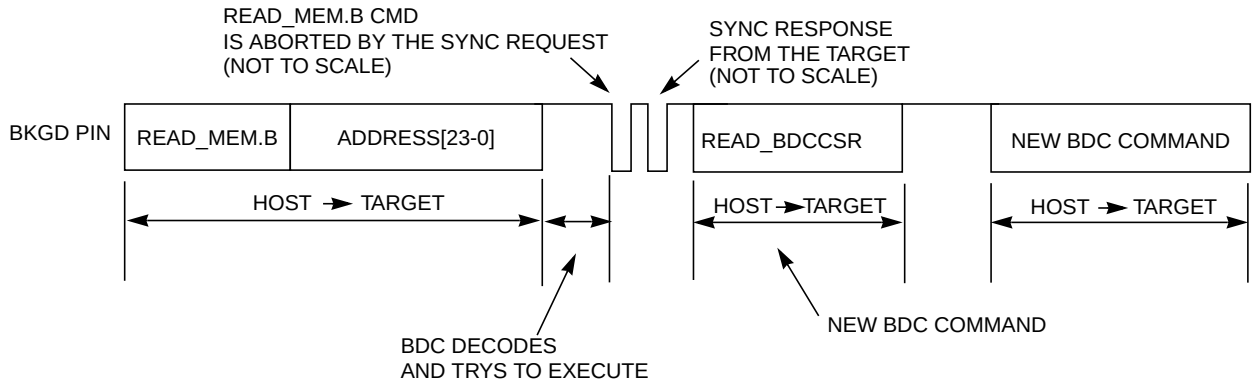



**Figure 5-11. ACK Abort Procedure at the Command Level (Not To Scale)**

Figure 5-12 shows a conflict between the ACK pulse and the SYNC request pulse. The target is executing a pending BDC command at the exact moment the host is being connected to the BKGD pin. In this case, an ACK pulse is issued simultaneously to the SYNC command. Thus there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. As this is not a probable situation, the protocol does not prevent this conflict from happening.

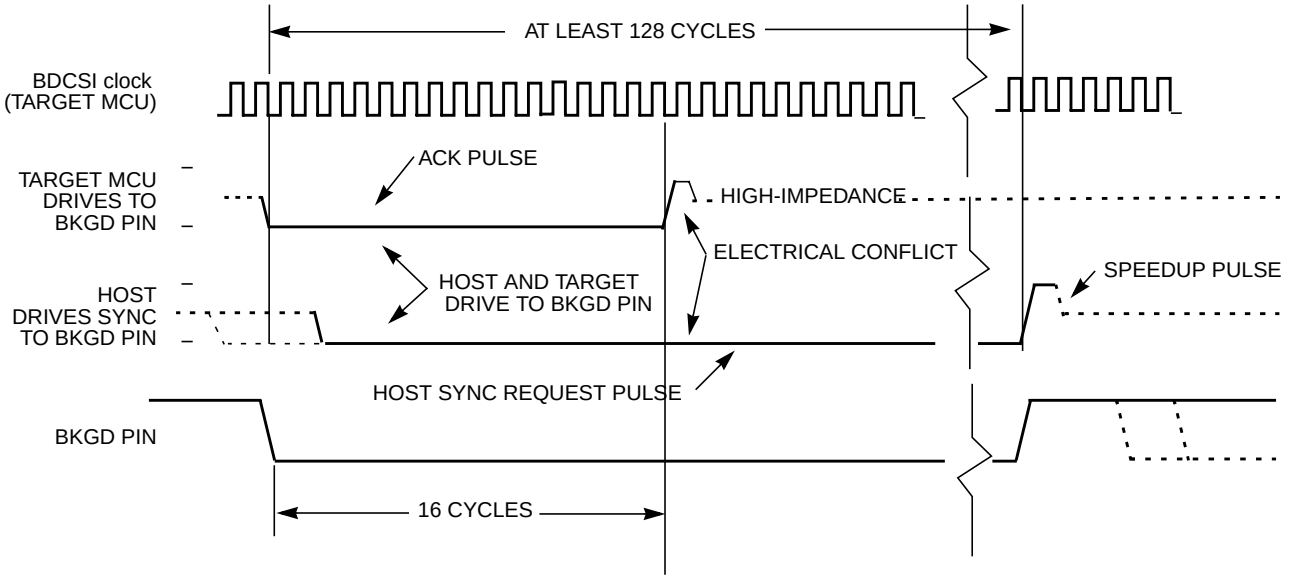



**Figure 5-12. ACK Pulse and SYNC Request Conflict**

## 5.4.9 Hardware Handshake Disabled (ACK Pulse Disabled)

The default state of the BDC after reset is hardware handshake protocol disabled. It can also be disabled by the ACK_DISABLE BDC command. This provides backwards compatibility with the existing host devices which are not able to execute the hardware handshake protocol. For host devices that support the hardware handshake protocol, true non-intrusive debugging and error flagging is offered.

If the ACK pulse protocol is disabled, the host needs to use the worst case delay time at the appropriate places in the protocol.

If the handshake protocol is disabled, the access is always independent of free cycles, whereby BDC has higher priority than CPU. Since at least 2 bytes (command byte + data byte) are transferred over BKGD the maximum intrusiveness is only once every few hundred cycles.

After decoding an internal access command, the BDC then awaits the next internal core clock cycle. The relationship between BDCSI clock and core clock must be considered. If the host retrieves the data immediately, then the BDCSI clock frequency must not be more than 4 times the core clock frequency, in order to guarantee that the BDC gains bus access within 16 the BDCSI cycle DLY period following an access command. If the BDCSI clock frequency is more than 4 times the core clock frequency, then the host must use a suitable delay time before retrieving data (see 5.5.1/5-170). Furthermore, for stretched read accesses to external resources via a device expanded bus (if implemented) the potential extra stretch cycles must be taken into consideration before attempting to obtain read data.

If the access does not succeed before the host starts data retrieval then the NORESP flag is set but the access is not aborted. The NORESP state can be used by the host to recognize an unexpected access conflict due to stretched expanded bus accesses. Although the NORESP bit is set when an access does not succeed before the start of data retrieval, the access may succeed in following bus cycles if the internal access has already been initiated.

### 5.4.10 Single Stepping

When a STEP1 command is issued to the BDC in active BDM, the CPU executes a single instruction in the user code and returns to active BDM. The STEP1 command can be issued repeatedly to step through the user code one instruction at a time.

If an interrupt is pending when a STEP1 command is issued, the interrupt stacking operation occurs but no user instruction is executed. In this case the stacking counts as one instruction. The device re-enters active BDM with the program counter pointing to the first instruction in the interrupt service routine.

When stepping through the user code, the execution of the user code is done step by step but peripherals are free running. Some peripheral modules include a freeze feature, whereby their clocks are halted when the device enters active BDM. Timer modules typically include the freeze feature. Serial interface modules typically do not include the freeze feature. Hence possible timing relations between CPU code execution and occurrence of events of peripherals no longer exist.

If the handshake protocol is enabled and BDCCIS is set then stepping over the STOP instruction causes the Long-ACK pulse to be generated and the BDCCSR STOP flag to be set. When stop mode is exited due to an interrupt the device enters active BDM and the PC points to the start of the corresponding interrupt service routine. Stepping can be continued.

Stepping over a WAI instruction, the STEP1 command cannot be finished because active BDM cannot be entered after CPU starts to execute the WAI instruction.

Stepping over the WAI instruction causes the BDCCSR WAIT and NORESP flags to be set and, if the handshake protocol is enabled, then the Long-ACK pulse is generated. Then the device enters wait mode, clears the BDMACT bit and awaits an interrupt to leave wait mode. In this time non-intrusive BDC commands are possible, although the STEP1 has actually not finished. When an interrupt occurs the device leaves wait mode, enters active BDM and the PC points to the start of the corresponding interrupt service routine. A further ACK related to stepping over the WAI is not generated.

### 5.4.11 Serial Communication Timeout

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target waits for a rising edge on BKGD in order to answer the SYNC request pulse. When the BDC detects the rising edge a soft reset is generated, whereby the current BDC command is discarded. If the rising edge is not detected, the target keeps waiting forever without any timeout limit.

If a falling edge is not detected by the target within 512 clock cycles since the last falling edge, a timeout occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset. This timeout also applies if 512 cycles elapse between 2 consecutive ERASE_FLASH commands. The soft reset is disabled whilst the internal flash mass erase operation is pending completion.

timeouts are also possible if a BDC command is partially issued, or data partially retrieved. Thus if a time greater than 512 BDCSI clock cycles is observed between two consecutive negative edges, a soft-reset occurs causing the partially received command or data retrieved to be discarded. The next negative edge at the BKGD pin, after a soft-reset has occurred, is considered by the target as the start of a new BDC command, or the start of a SYNC request pulse.

## 5.5 Application Information

### 5.5.1 Clock Frequency Considerations

Read commands without status and without ACK must consider the frequency relationship between BDCSI and the internal core clock. If the core clock is slow, then the internal access may not have been carried out within the standard 16 BDCSI cycle delay period (DLY). The host must then extend the DLY period or clock frequencies accordingly. Taking internal clock domain synchronizers into account, the minimum number of BDCSI periods required for the DLY is expressed by:

$\#DLY > 3(f_{(BDCSI\ clock)} / f_{(core\ clock)}) + 4$

and the minimum core clock frequency with respect to BDCSI clock frequency is expressed by

Minimum $f_{(core\ clock)} = (3/(\#DLY\ cycles\ -4))f_{(BDCSI\ clock)}$

For the standard 16 period DLY this yields $f_{(core\ clock)} >= (1/4)f_{(BDCSI\ clock)}$

# Chapter 6
# S12Z Debug (S12ZDBGV2) Module

**Table 6-1. Revision History Table**

| Revision Number | Revision Date | Sections Affected | Description Of Changes |
|---|---|---|---|
| 2.04 | 19.APR.2012 | Section 6.4.5.2.1 | Documented DBGTB read dependency on PROFILE bit |
| 2.05 | 23.MAY.2012 | General | Formatting changes to support DBGV3 from single source |
| 2.06 | 10.SEP.2012 | Section 6.4.5.3 | Added NOTE about PC trace buffer entries for Comp D timestamps |
| 2.07 | 18.OCT.2012 | General | Formatting corrections |
| 2.08 | 16.NOV.2012 | Section 6.5.1 | Modified step over breakpoint information |
| 2.09 | 19.DEC.2012 | General | Formatting corrections |
| 2.10 | 28.JUN.2013 | General<br>Section 6.3.2.21<br>Section 6.3.2.1<br>Section 6.3.2.5 | Emphasized need to set TSOURCE for tracing or profiling<br>Corrected DBGCDM write access dependency<br>Corrrected ARM versus PTACT dependency<br>Modified DBGTBH read access dependencies |
| 2.11 | 15.JUL.2013 | Section 6.3.2 | Added explicit names to state control register bit fields |

## 6.1 Introduction

The DBG module provides on-chip breakpoints and trace buffer with flexible triggering capability to allow non-intrusive debug of application software. The DBG module is optimized for the S12Z architecture and allows debugging of CPU module operations.

Typically the DBG module is used in conjunction with the BDC module, whereby the user configures the DBG module for a debugging session over the BDC interface. Once configured the DBG module is armed and the device leaves active BDM returning control to the user program, which is then monitored by the DBG module. Alternatively the DBG module can be configured over a serial interface using SWI routines.

### 6.1.1 Glossary

**Table 6-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| COF | Change Of Flow.<br>Change in the program flow due to a conditional branch, indexed jump or interrupt |
| PC | Program Counter |

**Table 6-2. Glossary Of Terms**

| Term | Definition |
|---|---|
| BDM | Background Debug Mode.<br>In this mode CPU application code execution is halted.<br>Execution of BDC “active BDM” commands is possible. |
| BDC | Background Debug Controller |
| WORD | 16-bit data entity |
| Data Line | 64-bit data entity |
| CPU | S12Z CPU module |
| Trigger | A trace buffer input that triggers tracing start, end or mid point |

## 6.1.2 Overview

The comparators monitor the bus activity of the CPU. A single comparator match or a series of matches can trigger bus tracing and/or generate breakpoints. A state sequencer determines if the correct series of matches occurs. Similarly an external event can trigger bus tracing and/or generate breakpoints.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads.

## 6.1.3 Features

- Four comparators (A, B, C, and D)
  - Comparators A and C compare the full address bus and full 32-bit data bus
  - Comparators A and C feature a data bus mask register
  - Comparators B and D compare the full address bus only
  - Each comparator can be configured to monitor PC addresses or addresses of data accesses
  - Each comparator can select either read or write access cycles
  - Comparator matches can force state sequencer state transitions
- Three comparator modes
  - Simple address/data comparator match mode
  - Inside address range mode, Addmin ≤ Address ≤ Addmax
  - Outside address range match mode, Address < Addmin or Address > Addmax
- State sequencer control
  - State transitions forced by comparator matches
  - State transitions forced by software write to TRIG
  - State transitions forced by an external event
- The following types of breakpoints
  - CPU breakpoint entering active BDM on breakpoint (BDM)
  - CPU breakpoint executing SWI on breakpoint (SWI)
- Trace control

  - Tracing session triggered by state sequencer
  - Begin, End, and Mid alignment of tracing to trigger
- Four trace modes
  - Normal: change of flow (COF) PC information is stored (see Section 6.4.5.2.1) for change of flow definition.
  - Loop1: same as Normal but inhibits consecutive duplicate source address entries
  - Detail: address and data for all read/write access cycles are stored
  - Pure PC: All program counter addresses are stored.
- 2 Pin (data and clock) profiling interface
  - Output of code flow information

## 6.1.4 Modes of Operation

The DBG module can be used in all MCU functional modes.

The DBG module can issue breakpoint requests to force the device to enter active BDM or an SWI ISR. The BDC BACKGROUND command is also handled by the DBG to force the device to enter active BDM. When the device enters active BDM through a BACKGROUND command with the DBG module armed, the DBG remains armed.

## 6.1.5 Block Diagram

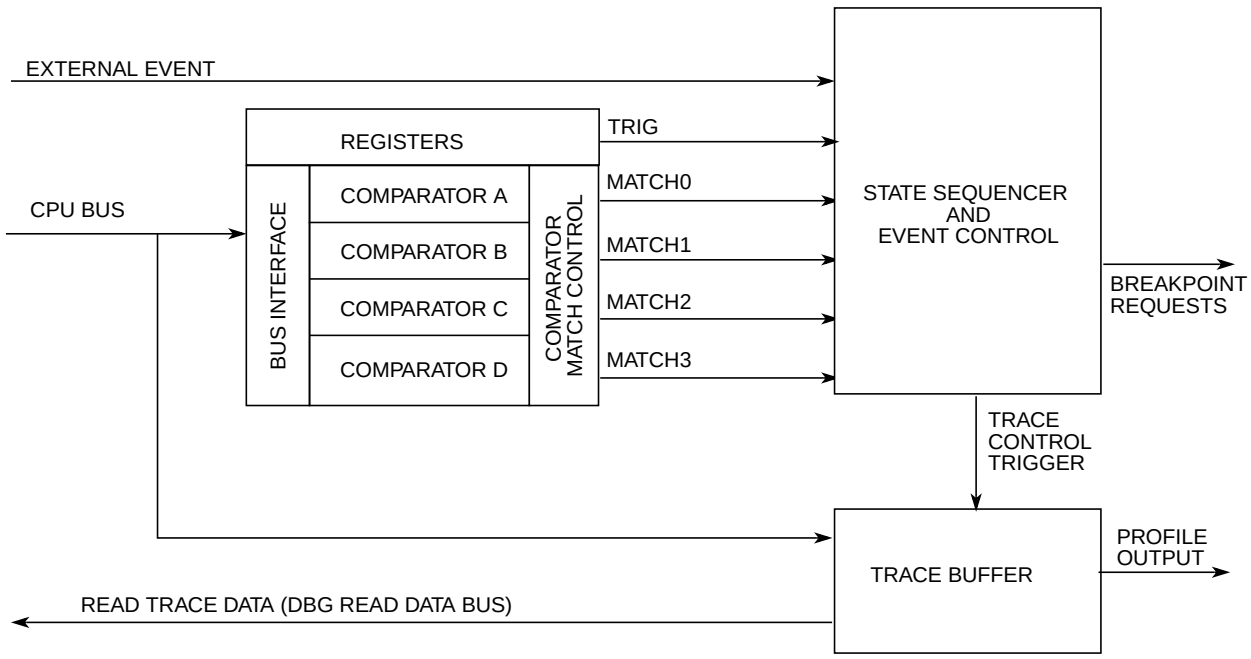



**Figure 6-1. Debug Module Block Diagram**

## 6.2 External Signal Description

### 6.2.1 External Event Input

The DBG module features an external event input signal, DBGEEV. The mapping of this signal to a device pin is specified in the device specific documentation. This function can be enabled and configured by the EEVE field in the DBGC1 control register. This signal is input only and allows an external event to force a state sequencer transition, or trace buffer entry, or to gate trace buffer entries. With the external event function enabled, a falling edge at the external event pin constitutes an event. Rising edges have no effect. If configured for gating trace buffer entries, then a low level at the pin allows entries, but a high level suppresses entries. The maximum frequency of events is half the internal core bus frequency. The function is explained in the EEVE field description.

**NOTE**

Due to input pin synchronization circuitry, the DBG module sees external events 2 bus cycles after they occur at the pin. Thus an external event occurring less than 2 bus cycles before arming the DBG module is perceived to occur whilst the DBG is armed.

When the device is in stop mode the synchronizer clocks are disabled and the external events are ignored.

### 6.2.2 Profiling Output

The DBG module features a profiling data output signal PDO. The mapping of this signal to a device pin is specified in the device specific documentation. The device pin is enabled for profiling by setting the PDOE bit. The profiling function can be enabled by the PROFILE bit in the DBGTCRL control register. This signal is output only and provides a serial, encoded data stream that can be used by external development tools to reconstruct the internal CPU code flow, as specified in Section 6.4.6. During code profiling the device PDOCLK output is used as a clock signal.

## 6.3 Memory Map and Registers

### 6.3.1 Module Memory Map

A summary of the registers associated with the DBG module is shown in Figure 6-2. Detailed descriptions of the registers and bits are given in the subsections that follow.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0100 | DBGC1 | R | ARM | 0 | reserved | BDMBP | BRKCPU | reserved | EEVE | |
| | | W | | TRIG | | | | | | |
| 0x0101 | DBGC2 | R | 0 | 0 | 0 | 0 | CDCM | | ABCM | |
| | | W | | | | | | | | |

**Figure 6-2. Quick Reference to DBG Registers**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0102 | DBGTCRH | R | reserved | TSOURCE | TRANGE | | TRCMOD | | TALIGN | |
| | | W | | | | | | | | |
| 0x0103 | DBGTCRL | R | 0 | 0 | 0 | 0 | DSTAMP | PDOE | PROFILE | STAMP |
| | | W | | | | | | | | |
| 0x0104 | DBGTB | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0105 | DBGTB | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0106 | DBGCNT | R | 0 | CNT | | | | | | |
| | | W | | | | | | | | |
| 0x0107 | DBGSCR1 | R | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x0108 | DBGSCR2 | R | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x0109 | DBGSCR3 | R | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x010A | DBGEFR | R | PTBOVF | TRIGF | 0 | EEVF | ME3 | ME2 | ME1 | ME0 |
| | | W | | | | | | | | |
| 0x010B | DBGSR | R | TBF | 0 | 0 | PTACT | 0 | SSF2 | SSF1 | SSF0 |
| | | W | | | | | | | | |
| 0x010C-0x010F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0110 | DBGACTL | R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| | | W | | | | | | | | |
| 0x0111-0x0114 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0115 | DBGAAH | R | DBGAA[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0116 | DBGAAM | R | DBGAA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0117 | DBGAAL | R | DBGAA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0118 | DBGAD0 | R | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| | | W | | | | | | | | |
| 0x0119 | DBGAD1 | R | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |
| 0x011A | DBGAD2 | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |

**Figure 6-2. Quick Reference to DBG Registers**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x011B | DBGAD3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x011C | DBGADM0 | R<br>W | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| 0x011D | DBGADM1 | R<br>W | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| 0x011E | DBGADM2 | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x011F | DBGADM3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0120 | DBGBCTL | R<br>W | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| 0x0121-<br>0x0124 | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0125 | DBGBAH | R<br>W | DBGBA[23:16] | | | | | | | |
| 0x0126 | DBGBAM | R<br>W | DBGBA[15:8] | | | | | | | |
| 0x0127 | DBGBAL | R<br>W | DBGBA[7:0] | | | | | | | |
| 0x0128-<br>0x012F | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0130 | DBGCCTL | R<br>W | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| 0x0131-<br>0x0134 | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0135 | DBGCAH | R<br>W | DBGCA[23:16] | | | | | | | |
| 0x0136 | DBGCAM | R<br>W | DBGCA[15:8] | | | | | | | |
| 0x0137 | DBGCAL | R<br>W | DBGCA[7:0] | | | | | | | |
| 0x0138 | DBGCD0 | R<br>W | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| 0x0139 | DBGCD1 | R<br>W | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| 0x013A | DBGCD2 | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |

**Figure 6-2. Quick Reference to DBG Registers**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x013B | DBGCD3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x013C | DBGCDM0 | R<br>W | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| 0x013D | DBGCDM1 | R<br>W | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| 0x013E | DBGCDM2 | R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x013F | DBGCDM3 | R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0140 | DBGDCTL | R<br>W | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| 0x0141-<br>0x0144 | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0145 | DBGDAH | R<br>W | DBGDA[23:16] | | | | | | | |
| 0x0146 | DBGDAM | R<br>W | DBGDA[15:8] | | | | | | | |
| 0x0147 | DBGDAL | R<br>W | DBGDA[7:0] | | | | | | | |
| 0x0148-<br>0x017F | Reserved | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-2. Quick Reference to DBG Registers**

## 6.3.2 Register Descriptions

This section consists of the DBG register descriptions in address order. When ARM is set in DBGC1, the only bits in the DBG module registers that can be written are ARM, and TRIG

### 6.3.2.1 Debug Control Register 1 (DBGC1)

Address: 0x0100

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x0100 | ARM | 0<br>TRIG | reserved | BDMBP | BRKCPU | reserved | EEVE | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-3. Debug Control Register (DBGC1)**

Read: Anytime

Write: Bit 7 Anytime with the exception that it cannot be set if PTACT is set. An ongoing profiling session must be finished before DBG can be armed again.

Bit 6 can be written anytime but always reads back as 0.

Bits 5:0 anytime DBG is not armed and PTACT is clear.

**NOTE**

On a write access to DBGC1 and simultaneous hardware disarm from an internal event, the hardware disarm has highest priority, clearing the ARM bit and generating a breakpoint, if enabled.

**NOTE**

When disarming the DBG by clearing ARM with software, the contents of bits[5:0] are not affected by the write, since up until the write operation, ARM = 1 preventing these bits from being written. These bits must be cleared using a second write if required.

**Table 6-3. DBGC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ARM | **Arm Bit** — The ARM bit controls whether the DBG module is armed. This bit can be set and cleared by register writes and is automatically cleared when the state sequencer returns to State0 on completing a debugging session. On setting this bit the state sequencer enters State1.<br>0 Debugger disarmed. No breakpoint is generated when clearing this bit by software register writes.<br>1 Debugger armed |
| 6<br>TRIG | **Immediate Trigger Request Bit** — This bit when written to 1 requests an immediate transition to final state independent of comparator status. This bit always reads back a 0. Writing a 0 to this bit has no effect.<br>0 No effect.<br>1 Force state sequencer immediately to final state. |
| 4<br>BDMBP | **Background Debug Mode Enable** — This bit determines if a CPU breakpoint causes the system to enter Background Debug Mode (BDM) or initiate a Software Interrupt (SWI). If this bit is set but the BDC is not enabled, then no breakpoints are generated.<br>0 Breakpoint to Software Interrupt if BDM inactive. Otherwise no breakpoint.<br>1 Breakpoint to BDM, if BDC enabled. Otherwise no breakpoint. |
| 3<br>BRKCPU | **CPU Breakpoint Enable** — The BRKCPU bit controls whether the debugger requests a breakpoint to CPU upon transitions to State0. If tracing is enabled, the breakpoint is generated on completion of the tracing session. If tracing is not enabled, the breakpoint is generated immediately. Please refer to Section 6.4.7 for further details.<br>0 Breakpoints disabled<br>1 Breakpoints enabled |
| 1–0<br>EEVE | **External Event Enable** — The EEVE bits configure the external event function. Table 6-4 explains the bit encoding. |

**Table 6-4. EEVE Bit Encoding**

| EEVE | Description |
|---|---|
| 00 | External event function disabled |
| 01 | External event forces a trace buffer entry if tracing is enabled |
| 10 | External event is mapped to the state sequencer, replacing comparator channel 3 |
| 11 | External event pin gates trace buffer entries |

### 6.3.2.2 Debug Control Register2 (DBGC2)

Address: 0x0101

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CDCM | | ABCM | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-4. Debug Control Register2 (DBGC2)**

Read: Anytime.

Write: Anytime the module is disarmed and PTACT is clear.

This register configures the comparators for range matching.

**Table 6-5. DBGC2 Field Descriptions**

| Field | Description |
|---|---|
| 3–2<br>CDCM[1:0] | **C and D Comparator Match Control** — These bits determine the C and D comparator match mapping as described in Table 6-6. |
| 1–0<br>ABCM[1:0] | **A and B Comparator Match Control** — These bits determine the A and B comparator match mapping as described in Table 6-7. |

**Table 6-6. CDCM Encoding**

| CDCM | Description |
|---|---|
| 00 | Match2 mapped to comparator C match..... Match3 mapped to comparator D match. |
| 01 | Match2 mapped to comparator C/D inside range..... Match3 disabled. |
| 10 | Match2 mapped to comparator C/D outside range..... Match3 disabled. |
| 11 | Reserved[1] |

1 Currently defaults to Match2 mapped to inside range: Match3 disabled.

**Table 6-7. ABCM Encoding**

| ABCM | Description |
|---|---|
| 00 | Match0 mapped to comparator A match..... Match1 mapped to comparator B match. |
| 01 | Match0 mapped to comparator A/B inside range..... Match1 disabled. |
| 10 | Match0 mapped to comparator A/B outside range..... Match1 disabled. |
| 11 | Reserved[1] |

1 Currently defaults to Match0 mapped to inside range: Match1 disabled

### 6.3.2.3 Debug Trace Control Register High (DBGTCRH)

Address: 0x0102

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | reserved | TSOURCE | TRANGE | | TRCMOD | | TALIGN | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-5. Debug Trace Control Register (DBGTCRH)**

Read: Anytime.

Write: Anytime the module is disarmed and PTACT is clear.

**WARNING**

**DBGTCR[7] is reserved. Setting this bit maps the tracing to an unimplemented bus, thus preventing proper operation.**

This register configures the trace buffer for tracing and profiling.

**Table 6-8. DBGTCRH Field Descriptions**

| Field | Description |
|---|---|
| 6<br>TSOURCE | **Trace Control Bits** — The TSOURCE enables the tracing session.<br>0 No CPU tracing/profiling selected<br>1 CPU tracing/profiling selected |
| 5–4<br>TRANGE | **Trace Range Bits** — The TRANGE bits allow filtering of trace information from a selected address range when tracing from the CPU in Detail mode. These bits have no effect in other tracing modes. To use a comparator for range filtering, the corresponding COMPE bit must remain cleared. If the COMPE bit is set then the comparator is used to generate events and the TRANGE bits have no effect. See Table 6-9 for range boundary definition. |
| 3–2<br>TRCMOD | **Trace Mode Bits** — See Section 6.4.5.2 for detailed Trace Mode descriptions. In Normal Mode, change of flow information is stored. In Loop1 Mode, change of flow information is stored but redundant entries into trace memory are inhibited. In Detail Mode, address and data for all memory and register accesses is stored. See Table 6-10. |
| 1–0<br>TALIGN | **Trigger Align Bits** — These bits control whether the trigger is aligned to the beginning, end or the middle of a tracing or profiling session. See Table 6-11. |

**Table 6-9. TRANGE Trace Range Encoding**

| TRANGE | Tracing Range |
|---|---|
| 00 | Trace from all addresses (No filter) |
| 01 | Trace only in address range from $00000 to Comparator D |
| 10 | Trace only in address range from Comparator C to $FFFFFF |
| 11 | Trace only in range from Comparator C to Comparator D |

**Table 6-10. TRCMOD Trace Mode Bit Encoding**

| TRCMOD | Description |
|---|---|
| 00 | Normal |
| 01 | Loop1 |

**Table 6-10. TRCMOD Trace Mode Bit Encoding**

| TRCMOD | Description |
|---|---|
| 10 | Detail |
| 11 | Pure PC |

**Table 6-11. TALIGN Trace Alignment Encoding**

| TALIGN | Description |
|---|---|
| 00 | Trigger ends data trace |
| 01 | Trigger starts data trace |
| 10 | 32 lines of data trace follow trigger |
| 11[1] | Reserved |

[1] Tracing/Profiling disabled.

### 6.3.2.4 Debug Trace Control Register Low (DBGTCRL)

Address: 0x0103

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | DSTAMP | PDOE | PROFILE | STAMP |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-6. Debug Trace Control Register Low (DBGTCRL)**

Read: Anytime.

Write: Anytime the module is disarmed and PTACT is clear.

This register configures the profiling and timestamp features

**Table 6-12. DBGTCRL Field Descriptions**

| Field | Description |
|---|---|
| 3<br>DSTAMP | **Comparator D Timestamp Enable** — This bit, when set, enables Comparator D matches to generate timestamps in Detail, Normal and Loop1 trace modes.<br>0 Comparator D match does not generate timestamp<br>1 Comparator D match generates timestamp if timestamp function is enabled |
| 2<br>PDOE | **Profile Data Out Enable** — This bit, when set, configures the device profiling pins for profiling.<br>0 Device pins not configured for profiling<br>1 Device pins configured for profiling |
| 1<br>PROFILE | **Profile Enable** — This bit, when set, enables the profile function, whereby a subsequent arming of the DBG activates profiling.<br>When PROFILE is set, the TRCMOD bits are ignored.<br>0 Profile function disabled<br>1 Profile function enabled |

**Table 6-12. DBGTCRL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 0<br>STAMP | **Timestamp Enable** — This bit, when set, enables the timestamp function. The timestamp function adds a timestamp to each trace buffer entry in Detail, Normal and Loop1 trace modes.<br>0 Timestamp function disabled<br>1 Timestamp function enabled |

### 6.3.2.5 Debug Trace Buffer Register (DBGTB)

Address: 0x0104, 0x0105

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| POR | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Other Resets | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

**Figure 6-7. Debug Trace Buffer Register (DBGTB)**

Read: Only when unlocked AND not armed and the TSOURCE bit is set, otherwise an error code (0xEE) is returned. Only aligned word read operations are supported. Misaligned word reads or byte reads return the error code 0xEE for each byte. The PROFILE bit must be clear to read profiling data,

Write: Aligned word writes when the DBG is disarmed and both PTACT and PROFILE are clear unlock the trace buffer for reading but do not affect trace buffer contents.

**Table 6-13. DBGTB Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>Bit[15:0] | **Trace Buffer Data Bits** — The Trace Buffer Register is a window through which the lines of the trace buffer may be read 16 bits at a time. Each valid read of DBGTB increments an internal trace buffer pointer which points to the next address to be read. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by writing to DBGTB with an aligned word write when the module is disarmed. The DBGTB register can be read only as an aligned word. Byte reads or misaligned access of these registers returns 0xEE and does not increment the trace buffer pointer. Similarly word reads while the debugger is armed or trace buffer is locked return 0xEEEE. The POR state is undefined Other resets do not affect the trace buffer contents. |

### 6.3.2.6 Debug Count Register (DBGCNT)

Address: 0x0106

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | CNT | | | | | | |
| W | | | | | | | | |
| Reset | 0 | — | — | — | — | — | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-8. Debug Count Register (DBGCNT)**

Read: Anytime.

Write: Never.

**Table 6-14. DBGCNT Field Descriptions**

| Field | Description |
|---|---|
| 6–0<br>CNT[6:0] | **Count Value** — The CNT bits [6:0] indicate the number of valid data lines stored in the trace buffer. Table 6-15 shows the correlation between the CNT bits and the number of valid data lines in the trace buffer. When the CNT rolls over to zero, the TBF bit in DBGSR is set. Thereafter incrementing of CNT continues if configured for end-alignment or mid-alignment.<br>The DBGCNT register is cleared when ARM in DBGC1 is written to a one. The DBGCNT register is cleared by power-on-reset initialization but is not cleared by other system resets. If a reset occurs during a debug session, the DBGCNT register still indicates after the reset, the number of valid trace buffer entries stored before the reset occurred. The DBGCNT register is not decremented when reading from the trace buffer. |

**Table 6-15. CNT Decoding Table**

| TBF (DBGSR) | CNT[6:0] | Description |
|---|---|---|
| 0 | 0000000 | No data valid |
| 0 | 0000001 | 32 bits of one line valid |
| 0 | 0000010<br>0000100<br>0000110<br>..<br>1111100 | 1 line valid<br>2 lines valid<br>3 lines valid<br>..<br>62 lines valid |
| 0 | 1111110 | 63 lines valid |
| 1 | 0000000 | 64 lines valid; if using Begin trigger alignment,<br>ARM bit is cleared and the tracing session ends. |
| 1 | 0000010<br>..<br>1111110 | 64 lines valid,<br>oldest data has been overwritten by most recent data |

## 6.3.2.7 Debug State Control Register 1 (DBGSCR1)

Address: 0x0107

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-9. Debug State Control Register 1 (DBGSCR1)**

Read: Anytime.

Write: If DBG is not armed and PTACT is clear.

The state control register 1 selects the targeted next state whilst in State1. The matches refer to the outputs of the comparator match control logic as depicted in Figure 6-1 and described in Section 6.3.2.12”. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

**Table 6-16. DBGSCR1 Field Descriptions**

| Field | Description |
|---|---|
| 1–0<br>C0SC[1:0] | Channel 0 State Control.<br>These bits select the targeted next state whilst in State1 following a match0. |
| 3–2<br>C1SC[1:0] | Channel 1 State Control.<br>These bits select the targeted next state whilst in State1 following a match1. |
| 5–4<br>C2SC[1:0] | Channel 2 State Control.<br>These bits select the targeted next state whilst in State1 following a match2. |
| 7–6<br>C3SC[1:0] | Channel 3 State Control.<br>If EEVE !=10, these bits select the targeted next state whilst in State1 following a match3.<br>If EEVE = 10, these bits select the targeted next state whilst in State1 following an external event. |

**Table 6-17. State1 Match State Sequencer Transitions**

| CxSC[1:0] | Function |
|---|---|
| 00 | Match has no effect |
| 01 | Match forces sequencer to State2 |
| 10 | Match forces sequencer to State3 |
| 11 | Match forces sequencer to Final State |

In the case of simultaneous matches, the match on the higher channel number (3...0) has priority.

### 6.3.2.8 Debug State Control Register 2 (DBGSCR2)

Address: 0x0108

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-10. Debug State Control Register 2 (DBGSCR2)**

Read: Anytime.

Write: If DBG is not armed and PTACT is clear.

The state control register 2 selects the targeted next state whilst in State2. The matches refer to the outputs of the comparator match control logic as depicted in Figure 6-1 and described in Section 6.3.2.12”. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

**Table 6-18. DBGSCR2 Field Descriptions**

| Field | Description |
|---|---|
| 1–0<br>C0SC[1:0] | Channel 0 State Control.<br>These bits select the targeted next state whilst in State2 following a match0. |
| 3–2<br>C1SC[1:0] | Channel 1 State Control.<br>These bits select the targeted next state whilst in State2 following a match1. |

**Table 6-18. DBGSCR2 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5–4<br>C2SC[1:0] | Channel 2 State Control.<br>These bits select the targeted next state whilst in State2 following a match2. |
| 7–6<br>C3SC[1:0] | Channel 3 State Control.<br>If EEVE !=10, these bits select the targeted next state whilst in State2 following a match3.<br>If EEVE =10, these bits select the targeted next state whilst in State2 following an external event. |

**Table 6-19. State2 Match State Sequencer Transitions**

| CxSC[1:0] | Function |
|---|---|
| 00 | Match has no effect |
| 01 | Match forces sequencer to State1 |
| 10 | Match forces sequencer to State3 |
| 11 | Match forces sequencer to Final State |

In the case of simultaneous matches, the match on the higher channel number (3...0) has priority.

### 6.3.2.9 Debug State Control Register 3 (DBGSCR3)

Address: 0x0109

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-11. Debug State Control Register 3 (DBGSCR3)**

Read: Anytime.

Write: If DBG is not armed and PTACT is clear.

The state control register three selects the targeted next state whilst in State3. The matches refer to the outputs of the comparator match control logic as depicted in Figure 6-1 and described in Section 6.3.2.12”. Comparators must be enabled by setting the comparator enable bit in the associated DBGxCTL control register.

**Table 6-20. DBGSCR3 Field Descriptions**

| Field | Description |
|---|---|
| 1–0<br>C0SC[1:0] | Channel 0 State Control.<br>These bits select the targeted next state whilst in State3 following a match0. |
| 3–2<br>C1SC[1:0] | Channel 1 State Control.<br>These bits select the targeted next state whilst in State3 following a match1. |
| 5–4<br>C2SC[1:0] | Channel 2 State Control.<br>These bits select the targeted next state whilst in State3 following a match2. |

**Table 6-20. DBGSCR3 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 7–6<br>C3SC[1:0] | Channel 3 State Control.<br>If EEVE !=10, these bits select the targeted next state whilst in State3 following a match3.<br>If EEVE =10, these bits select the targeted next state whilst in State3 following an external event. |

**Table 6-21. State3 Match State Sequencer Transitions**

| CxSC[1:0] | Function |
|---|---|
| 00 | Match has no effect |
| 01 | Match forces sequencer to State1 |
| 10 | Match forces sequencer to State2 |
| 11 | Match forces sequencer to Final State |

In the case of simultaneous matches, the match on the higher channel number (3....0) has priority.

### 6.3.2.10 Debug Event Flag Register (DBGEFR)

Address: 0x010A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PTBOVF | TRIGF | 0 | EEVF | ME3 | ME2 | ME1 | ME0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-12. Debug Event Flag Register (DBGEFR)**

Read: Anytime.

Write: Never

DBGEFR contains flag bits each mapped to events whilst armed. Should an event occur, then the corresponding flag is set. With the exception of TRIGF, the bits can only be set when the ARM bit is set. The TRIGF bit is set if a TRIG event occurs when ARM is already set, or if the TRIG event occurs simultaneous to setting the ARM bit.All other flags can only be cleared by arming the DBG module. Thus the contents are retained after a debug session for evaluation purposes.

A set flag does not inhibit the setting of other flags.

**Table 6-22. DBGEFR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PTBOVF | **Profiling Trace Buffer Overflow Flag** — Indicates the occurrence of a trace buffer overflow event during a profiling session.<br>0 No trace buffer overflow event<br>1 Trace buffer overflow event |
| 6<br>TRIGF | **TRIG Flag** — Indicates the occurrence of a TRIG event during the debug session.<br>0 No TRIG event<br>1 TRIG event |

**Table 6-22. DBGEFR Field Descriptions**

| Field | Description |
|---|---|
| 4<br>EEVF | **External Event Flag** — Indicates the occurrence of an external event during the debug session.<br>0 No external event<br>1 External event |
| 3–0<br>ME[3:0] | **Match Event[3:0]**— Indicates a comparator match event on the corresponding comparator channel**.** |

## 6.3.2.11 Debug Status Register (DBGSR)

Address: 0x010B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TBF | 0 | 0 | PTACT | 0 | SSF2 | SSF1 | SSF0 |
| W | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-13. Debug Status Register (DBGSR)**

Read: Anytime.

Write: Never.

**Table 6-23. DBGSR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TBF | **Trace Buffer Full** — The TBF bit indicates that the trace buffer has been filled with data since it was last armed. If this bit is set, then all trace buffer lines contain valid data, regardless of the value of DBGCNT bits CNT[6:0]. The TBF bit is cleared when ARM in DBGC1 is written to a one. The TBF is cleared by the power on reset initialization. Other system generated resets have no affect on this bit |
| 4<br>PTACT | **Profiling Transmission Active** — The PTACT bit, when set, indicates that the profiling transmission is still active. When clear, PTACT then profiling transmission is not active. The PTACT bit is set when profiling begins with the first PTS format entry to the trace buffer. The PTACT bit is cleared when the profiling transmission ends. |
| 2–0<br>SSF[2:0] | **State Sequencer Flag Bits** — The SSF bits indicate the current State Sequencer state. During a debug session on each transition to a new state these bits are updated. If the debug session is ended by software clearing the ARM bit, then these bits retain their value to reflect the last state of the state sequencer before disarming. If a debug session is ended by an internal event, then the state sequencer returns to State0 and these bits are cleared to indicate that State0 was entered during the session. On arming the module the state sequencer enters State1 and these bits are forced to SSF[2:0] = 001. See Table 6-24**.** |

**Table 6-24. SSF[2:0] — State Sequence Flag Bit Encoding**

| SSF[2:0] | Current State |
|---|---|
| 000 | State0 (disarmed) |
| 001 | State1 |
| 010 | State2 |
| 011 | State3 |
| 100 | Final State |

**Table 6-24. SSF[2:0] — State Sequence Flag Bit Encoding**

| SSF[2:0] | Current State |
|---|---|
| 101,110,111 | Reserved |

### 6.3.2.12 Debug Comparator A Control Register (DBGACTL)

Address: 0x0110

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-14. Debug Comparator A Control Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-25. DBGACTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>NDB | **Not Data Bus** — The NDB bit controls whether the match occurs when the data bus matches the comparator register value or when the data bus differs from the register value. This bit is ignored if the INST bit in the same register is set.<br>0 Match on data bus equivalence to comparator register contents<br>1 Match on data bus difference to comparator register contents |
| 5<br>INST | **Instruction Select** — This bit configures the comparator to compare PC or data access addresses.<br>0 Comparator compares addresses of data accesses<br>1 Comparator compares PC address |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is ignored if RWE is clear or INST is set.<br>0 Write cycle is matched<br>1 Read cycle is matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is ignored when INST is set.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 0<br>COMPE | **Enable Bit** — Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled |

Table 6-26 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, because matches based on opcodes reaching the execution stage are data independent.

**Table 6-26. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |
| 1 | 1 | 1 | Read match |

### 6.3.2.13 Debug Comparator A Address Register (DBGAAH, DBGAAM, DBGAAL)

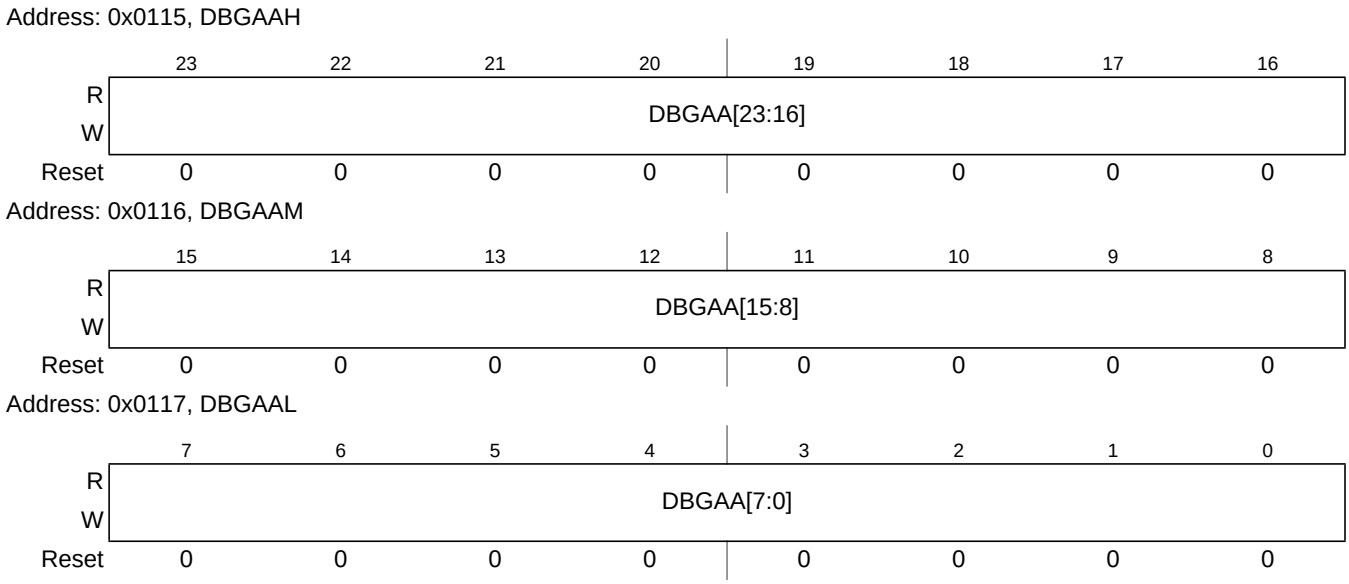



**Figure 6-15. Debug Comparator A Address Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-27. DBGAAH, DBGAAM, DBGAAL Field Descriptions**

| Field | Description |
|---|---|
| 23–16<br>DBGAA<br>[23:16] | **Comparator Address Bits [23:16]**— These comparator address bits control whether the comparator compares the address bus bits [23:16] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |
| 15–0<br>DBGAA<br>[15:0] | **Comparator Address Bits [15:0]**— These comparator address bits control whether the comparator compares the address bus bits [15:0] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

### 6.3.2.14 Debug Comparator A Data Register (DBGAD)

Address: 0x0118, 0x0119, 0x011A, 0x011B

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-16. Debug Comparator A Data Register (DBGAD)**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

This register can be accessed with a byte resolution, whereby DBGAD0, DBGAD1, DBGAD2, DBGAD3 map to DBGAD[31:0] respectively.

**Table 6-28. DBGAD Field Descriptions**

| Field | Description |
|---|---|
| 31–16<br>Bits[31:16]<br>(DBGAD0,<br>DBGAD1) | **Comparator Data Bits** — These bits control whether the comparator compares the data bus bits to a logic one or logic zero. The comparator data bits are only used in comparison if the corresponding data mask bit is logic 1.<br>0 Compare corresponding data bit to a logic zero<br>1 Compare corresponding data bit to a logic one |
| 15–0<br>Bits[15:0]<br>(DBGAD2,<br>DBGAD3) | **Comparator Data Bits** — These bits control whether the comparator compares the data bus bits to a logic one or logic zero. The comparator data bits are only used in comparison if the corresponding data mask bit is logic 1.<br>0 Compare corresponding data bit to a logic zero<br>1 Compare corresponding data bit to a logic one |

### 6.3.2.15 Debug Comparator A Data Mask Register (DBGADM)

Address: 0x011C, 0x011D, 0x011E, 0x011F

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-17. Debug Comparator A Data Mask Register (DBGADM)**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

This register can be accessed with a byte resolution, whereby DBGADM0, DBGADM1, DBGADM2, DBGADM3 map to DBGADM[31:0] respectively.

**Table 6-29. DBGADM Field Descriptions**

| Field | Description |
|---|---|
| 31–16<br>Bits[31:16]<br>(DBGADM0, DBGADM1) | **Comparator Data Mask Bits** — These bits control whether the comparator compares the data bus bits to the corresponding comparator data compare bits.<br>0 Do not compare corresponding data bit<br>1 Compare corresponding data bit |
| 15-0<br>Bits[15:0]<br>(DBGADM2, DBGADM3) | **Comparator Data Mask Bits** — These bits control whether the comparator compares the data bus bits to the corresponding comparator data compare bits.<br>0 Do not compare corresponding data bit<br>1 Compare corresponding data bit |

## 6.3.2.16 Debug Comparator B Control Register (DBGBCTL)

Address: 0x0120

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-18. Debug Comparator B Control Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-30. DBGBCTL Field Descriptions**

| Field[1] | Description |
|---|---|
| 5<br>INST | **Instruction Select** — This bit configures the comparator to compare PC or data access addresses.<br>0 Comparator compares addresses of data accesses<br>1 Comparator compares PC address |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is ignored if RWE is clear or INST is set.<br>0 Write cycle is matched<br>1 Read cycle is matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is ignored when INST is set.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 0<br>COMPE | **Enable Bit** — Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled |

[1] If the ABCM field selects range mode comparisons, then DBGACTL bits configure the comparison, DBGBCTL is ignored.

Table 6-31 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, as matches based on instructions reaching the execution stage are data independent.

**Table 6-31. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |
| 1 | 1 | 1 | Read match |

### 6.3.2.17 Debug Comparator B Address Register (DBGBAH, DBGBAM, DBGBAL)

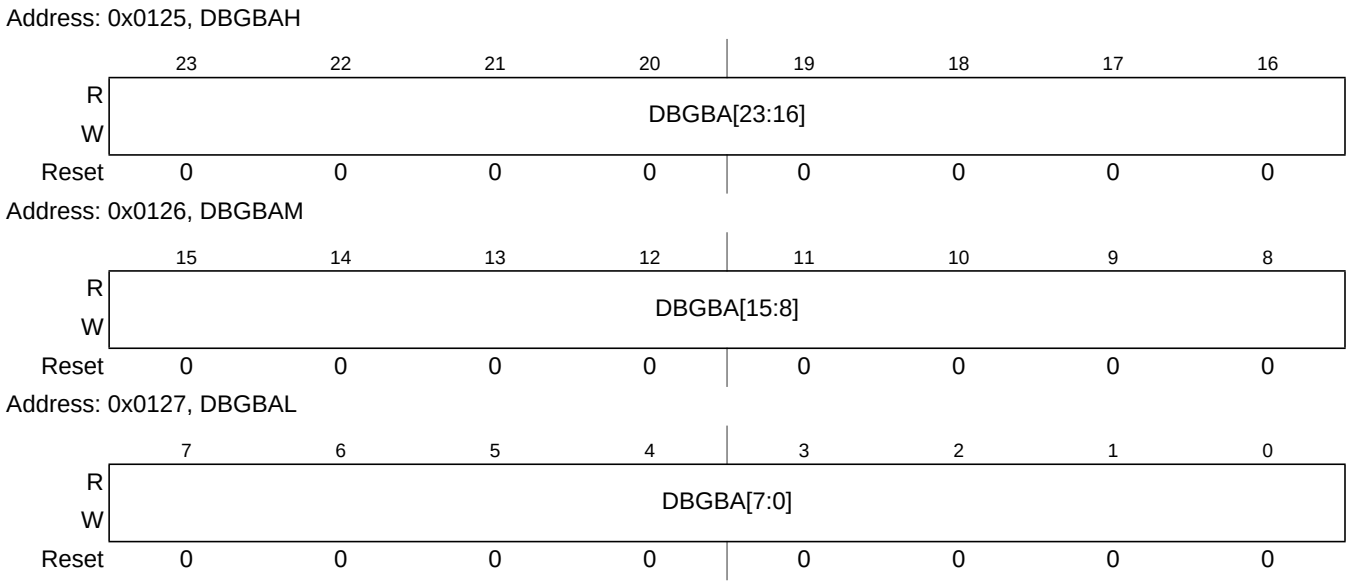



**Figure 6-19. Debug Comparator B Address Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-32. DBGBAH, DBGBAM, DBGBAL Field Descriptions**

| Field | Description |
|---|---|
| 23–16<br>DBGBA<br>[23:16] | **Comparator Address Bits [23:16]**— These comparator address bits control whether the comparator compares the address bus bits [23:16] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |
| 15–0<br>DBGBA<br>[15:0] | **Comparator Address Bits**[15:0]— These comparator address bits control whether the comparator compares the address bus bits [15:0] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

### 6.3.2.18 Debug Comparator C Control Register (DBGCCTL)

Address: 0x0130

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-20. Debug Comparator C Control Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-33. DBGCCTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>NDB | **Not Data Bus** — The NDB bit controls whether the match occurs when the data bus matches the comparator register value or when the data bus differs from the register value. This bit is ignored if the INST bit in the same register is set.<br>0 Match on data bus equivalence to comparator register contents<br>1 Match on data bus difference to comparator register contents |
| 5<br>INST | **Instruction Select** — This bit configures the comparator to compare PC or data access addresses.<br>0 Comparator compares addresses of data accesses<br>1 Comparator compares PC address |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is ignored if RWE is clear or INST is set.<br>0 Write cycle is matched<br>1 Read cycle is matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is not used if INST is set.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 0<br>COMPE | **Enable Bit** — Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled |

Table 6-34 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, because matches based on opcodes reaching the execution stage are data independent.

**Table 6-34. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |

**Table 6-34. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 1 | 1 | 1 | Read match |

### 6.3.2.19 Debug Comparator C Address Register (DBGCAH, DBGCAM, DBGCAL)

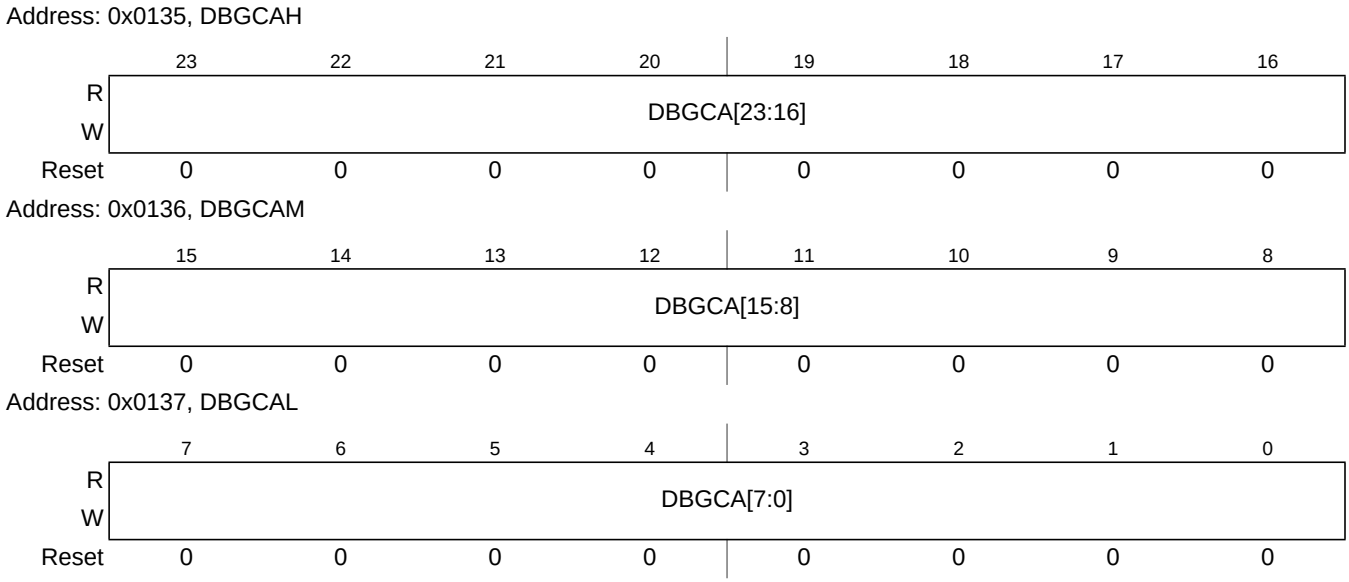



**Figure 6-21. Debug Comparator C Address Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-35. DBGCAH, DBGCAM, DBGCAL Field Descriptions**

| Field | Description |
|---|---|
| 23–16<br>DBGCA<br>[23:16] | **Comparator Address Bits [23:16]**— These comparator address bits control whether the comparator compares the address bus bits [23:16] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |
| 15–0<br>DBGCA<br>[15:0] | **Comparator Address Bits**[15:0]— These comparator address bits control whether the comparator compares the address bus bits [15:0] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

### 6.3.2.20 Debug Comparator C Data Register (DBGCD)

Address: 0x0138, 0x0139, 0x013A, 0x013B

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-22. Debug Comparator C Data Register (DBGCD)**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

This register can be accessed with a byte resolution, whereby DBGCD0, DBGCD1, DBGCD2, DBGCD3 map to DBGCD[31:0] respectively.

XGATE data accesses have a maximum width of 16-bits and are mapped to DBGCD[15:0].

**Table 6-36. DBGCD Field Descriptions**

| Field | Description |
|---|---|
| 31–16<br>Bits[31:16]<br>(DBGCD0, DBGCD1) | **Comparator Data Bits** — These bits control whether the comparator compares the data bus bits to a logic one or logic zero. The comparator data bits are only used in comparison if the corresponding data mask bit is logic 1.<br>0 Compare corresponding data bit to a logic zero<br>1 Compare corresponding data bit to a logic one |
| 15–0<br>Bits[15:0]<br>(DBGCD2, DBGCD3) | **Comparator Data Bits** — These bits control whether the comparator compares the data bus bits to a logic one or logic zero. The comparator data bits are only used in comparison if the corresponding data mask bit is logic 1.<br>0 Compare corresponding data bit to a logic zero<br>1 Compare corresponding data bit to a logic one |

### 6.3.2.21 Debug Comparator C Data Mask Register (DBGCDM)

Address: 0x013C, 0x013D, 0x013E, 0x013F

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-23. Debug Comparator C Data Mask Register (DBGCDM)**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

This register can be accessed with a byte resolution, whereby DBGCDM0, DBGCDM1, DBGCDM2, DBGCDM3 map to DBGCDM[31:0] respectively.

XGATE data accesses have a maximum width of 16-bits and are mapped to DBGCDM[15:0].

**Table 6-37. DBGCDM Field Descriptions**

| Field | Description |
|---|---|
| 31–16<br>Bits[31:16]<br>(DBGCDM0, DBGCDM1) | **Comparator Data Mask Bits** — These bits control whether the comparator compares the data bus bits to the corresponding comparator data compare bits.<br>0 Do not compare corresponding data bit<br>1 Compare corresponding data bit |
| 15–0<br>Bits[15:0]<br>(DBGCDM2, DBGCDM3) | **Comparator Data Mask Bits** — These bits control whether the comparator compares the data bus bits to the corresponding comparator data compare bits.<br>0 Do not compare corresponding data bit<br>1 Compare corresponding data bit |

## 6.3.2.22 Debug Comparator D Control Register (DBGDCTL)

Address: 0x0140

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-24. Debug Comparator D Control Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-38. DBGDCTL Field Descriptions**

| Field[1] | Description |
|---|---|
| 5<br>INST | **Instruction Select** — This bit configures the comparator to compare PC or data access addresses.<br>0 Comparator compares addresses of data accesses<br>1 Comparator compares PC address |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is ignored if RWE is clear or INST is set.<br>0 Write cycle is matched<br>1 Read cycle is matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is ignored if INST is set.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 0<br>COMPE | **Enable Bit** — Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled |

1 If the CDCM field selects range mode comparisons, then DBGCCTL bits configure the comparison, DBGDCTL is ignored.

Table 6-39 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, because matches based on opcodes reaching the execution stage are data independent.

**Table 6-39. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |
| 1 | 1 | 1 | Read match |

### 6.3.2.23 Debug Comparator D Address Register (DBGDAH, DBGDAM, DBGDAL)

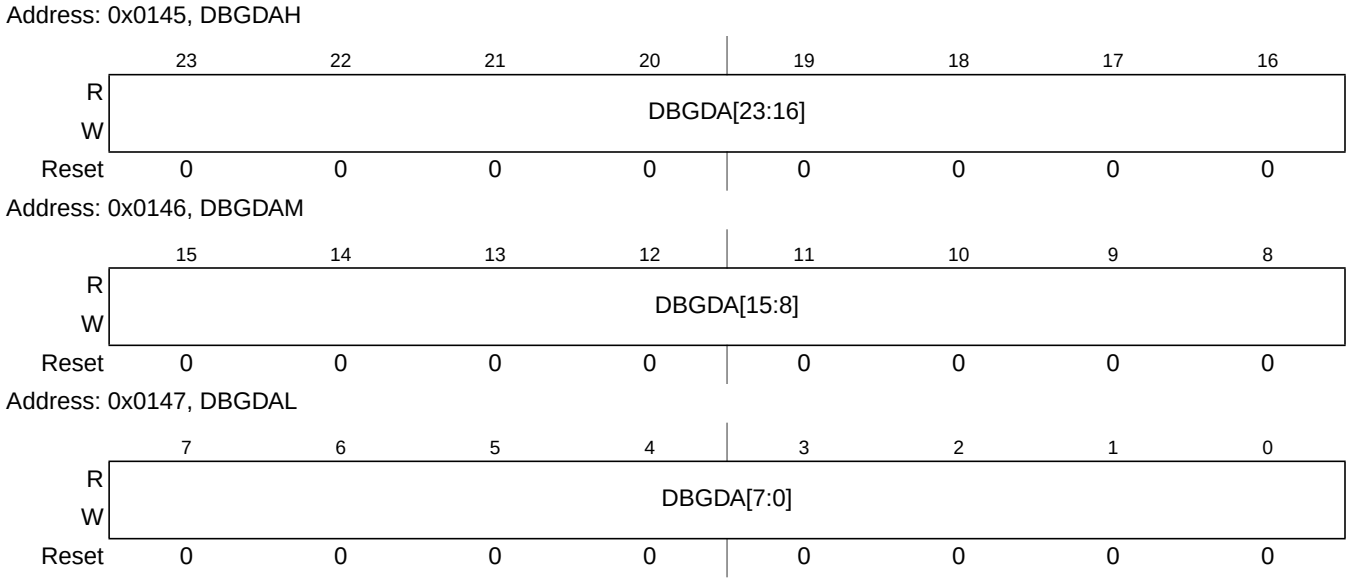



**Figure 6-25. Debug Comparator D Address Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-40. DBGDAH, DBGDAM, DBGDAL Field Descriptions**

| Field | Description |
|---|---|
| 23–16<br>DBGDA<br>[23:16] | **Comparator Address Bits [23:16]**— These comparator address bits control whether the comparator compares the address bus bits [23:16] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

**Table 6-40. DBGDAH, DBGDAM, DBGDAL Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>DBGDA<br>[15:0] | **Comparator Address Bits**[15:0]— These comparator address bits control whether the comparator compares the address bus bits [15:0] to a logic one or logic zero.<br>0 Compare corresponding address bit to a logic zero<br>1 Compare corresponding address bit to a logic one |

## 6.4 Functional Description

This section provides a complete functional description of the DBG module.

### 6.4.1 DBG Operation

The DBG module operation is enabled by setting ARM in DBGC1. When armed it supports storing of data in the trace buffer and can be used to generate breakpoints to the CPU. The DBG module is made up of comparators, control logic, the trace buffer, and the state sequencer, Figure 6-1.

The comparators monitor the bus activity of the CPU. Comparators can be configured to monitor opcode addresses (effectively the PC address) or data accesses. Comparators can be configured during data accesses to mask out individual data bus bits and to use R/W access qualification in the comparison. Comparators can be configured to monitor a range of addresses.

When configured for data access comparisons, the match is generated if the address (and optionally data) of a data access matches the comparator value.

Configured for monitoring opcode addresses, the match is generated when the associated opcode reaches the execution stage of the instruction queue, but before execution of that opcode.

When a match with a comparator register value occurs, the associated control logic can force the state sequencer to another state (see Figure 6-26).

The state sequencer can transition freely between the states 1, 2 and 3. On transition to Final State bus tracing can be triggered. On completion of tracing the state sequencer enters State0. If tracing is disabled or End aligned tracing is enabled then the state sequencer transitions immediately from Final State to State0. The transition to State0 generates breakpoints if breakpoints are enabled.

Independent of the comparators, state sequencer transitions can be forced by the external event input or by writing to the TRIG bit in the DBGC1 control register.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads.

### 6.4.2 Comparator Modes

The DBG contains four comparators, A, B, C, and D. Each comparator compares the address stored in DBGXAH, DBGXAM, and DBGXAL with the PC (opcode addresses) or selected address bus (data

accesses). Furthermore, comparators A and C can compare the data buses to values stored in DBGXD3-0 and allow data bit masking.

The comparators can monitor the buses for an exact address or an address range. The comparator configuration is controlled by the control register contents and the range control by the DBGC2 contents.

The comparator control register also allows the type of data access to be included in the comparison through the use of the RWE and RW bits. The RWE bit controls whether the access type is compared for the associated comparator and the RW bit selects either a read or write access for a valid match.

The INST bit in each comparator control register is used to determine the matching condition. By setting INST, the comparator matches opcode addresses, whereby the databus, data mask, RW and RWE bits are ignored. The comparator register must be loaded with the exact opcode address.

The comparator can be configured to match memory access addresses by clearing the INST bit.

Each comparator match can force a transition to another state sequencer state (see Section 6.4.3”).

Once a successful comparator match has occurred, the condition that caused the original match is not verified again on subsequent matches. Thus if a particular data value is matched at a given address, this address may not contain that data value when a subsequent match occurs.

Comparators C and D can also be used to select an address range to trace from, when tracing CPU accesses in Detail mode. This is determined by the TRANGE bits in the DBGTCRH register. The TRANGE encoding is shown in Table 6-9. If the TRANGE bits select a range definition using comparator D and the COMPE bit is clear, then comparator D is configured for trace range definition. By setting the COMPE bit the comparator is configured for address bus comparisons, the TRANGE bits are ignored and the tracing range function is disabled. Similarly if the TRANGE bits select a range definition using comparator C and the COMPE bit is clear, then comparator C is configured for trace range definition.

Match[0, 1, 2, 3] map directly to Comparators [A, B, C, D] respectively, except in range modes (see Section 6.3.2.2”). Comparator priority rules are described in the event priority section (Section 6.4.3.5”).

### 6.4.2.1 Exact Address Comparator Match

With range comparisons disabled, the match condition is an exact equivalence of address bus with the value stored in the comparator address registers. Qualification of the type of access (R/W) is also possible.

Code may contain various access forms of the same address, for example a 16-bit access of ADDR[n] or byte access of ADDR[n+1] both access n+1. The comparators ensure that any access of the address defined by the comparator address register generates a match, as shown in the example of Table 6-41. Thus if the comparator address register contains ADDR[n+1] any access of ADDR[n+1] matches. This means that a 16-bit access of ADDR[n] or 32-bit access of ADDR[n-1] also match because they also access ADDR[n+1]. The right hand columns show the contents of DBGxA that would match for each access.

**Table 6-41. Comparator Address Bus Matches**

| Access | Address | ADDR[n] | ADDR[n+1] | ADDR[n+2] | ADDR[n+3] |
|---|---|---|---|---|---|
| 32-bit | ADDR[n] | Match | Match | Match | Match |
| 16-bit | ADDR[n] | Match | Match | No Match | No Match |
| 16-bit | ADDR[n+1] | No Match | Match | Match | No Match |

**Table 6-41. Comparator Address Bus Matches**

| Access | Address | ADDR[n] | ADDR[n+1] | ADDR[n+2] | ADDR[n+3] |
|---|---|---|---|---|---|
| 8-bit | ADDR[n] | Match | No Match | No Match | No Match |

If the comparator INST bit is set, the comparator address register contents are compared with the PC, the data register contents and access type bits are ignored. The comparator address register must be loaded with the address of the first opcode byte.

### 6.4.2.2 Address and Data Comparator Match

Comparators A and C feature data comparators, for data access comparisons. The comparators do not evaluate if accessed data is valid. Accesses across aligned 32-bit boundaries are split internally into consecutive accesses. The data comparator mapping to accessed addresses for the CPU is shown in Table 6-42, whereby the Address column refers to the lowest 2 bits of the lowest accessed address. This corresponds to the most significant data byte.

**Table 6-42. Comparator Data Byte Alignment**

| Address[1:0] | Data Comparator |
|---|---|
| 00 | DBGxD0 |
| 01 | DBGxD1 |
| 10 | DBGxD2 |
| 11 | DBGxD3 |

The fixed mapping of data comparator bytes to addresses within a 32-bit data field ensures data matches independent of access size. To compare a single data byte within the 32-bit field, the other bytes within that field must be masked using the corresponding data mask registers. This ensures that any access of that byte (32-bit,16-bit or 8-bit) with matching data causes a match. If no bytes are masked then the data comparator always compares all 32-bits and can only generate a match on a 32-bit access with correct 32-bit data value. In this case, 8-bit or 16-bit accesses within the 32-bit field cannot generate a match even if the contents of the addressed bytes match because all 32-bits must match. In Table 6-43 the Access Address column refers to the address bits[1:0] of the lowest accessed address (most significant data byte).

**Table 6-43. Data Register Use Dependency On CPU Access Type**

| | | | Memory Address[2:0] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Case | Access Address | Access Size | 000 | 001 | 010 | 011 | 100 | 101 | 110 |
| 1 | 00 | 32-bit | DBGxD0 | DBGxD1 | DBGxD2 | DBGxD3 | | | |
| 2 | 01 | 32-bit | | DBGxD1 | DBGxD2 | DBGxD3 | DBGxD0 | | |
| 3 | 10 | 32-bit | | | DBGxD2 | DBGxD3 | DBGxD0 | DBGxD1 | |
| 4 | 11 | 32-bit | | | | DBGxD3 | DBGxD0 | DBGxD1 | DBGxD2 |
| 5 | 00 | 16-bit | DBGxD0 | DBGxD1 | | | | | |
| 6 | 01 | 16-bit | | DBGxD1 | DBGxD2 | | | | |
| 7 | 10 | 16-bit | | | DBGxD2 | DBGxD3 | | | |
| 8 | 11 | 16-bit | | | | DBGxD3 | DBGxD0 | | |

| | | | Memory Address[2:0] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Case | Access Address | Access Size | 000 | 001 | 010 | 011 | 100 | 101 | 110 |
| 9 | 00 | 8-bit | DBGxD0 | | | | | | |
| 10 | 01 | 8-bit | | DBGxD1 | | | | | |
| 11 | 10 | 8-bit | | | DBGxD2 | | | | |
| 12 | 11 | 8-bit | | | | DBGxD3 | | | |
| 13 | 00 | 8-bit | | | | | DBGxD0 | | |
| | | | | Denotes byte that is not accessed. | | | | | |

For a match of a 32-bit access with data compare, the address comparator must be loaded with the address of the lowest accessed byte. For Case1 Table 6-43 this corresponds to 000, for Case2 it corresponds to 001. To compare all 32-bits, it is required that no bits are masked.

### 6.4.2.3 Data Bus Comparison NDB Dependency

The NDB control bit allows data bus comparators to be configured to either match on equivalence or on difference. This allows monitoring of a difference in the contents of an address location from an expected value.

When matching on an equivalence (NDB=0), each individual data bus bit position can be masked out by clearing the corresponding mask bit, so that it is ignored in the comparison. A match occurs when all data bus bits with corresponding mask bits set are equivalent. If all mask register bits are clear, then a match is based on the address bus only, the data bus is ignored.

When matching on a difference, mask bits can be cleared to ignore bit positions. A match occurs when any data bus bit with corresponding mask bit set is different. Clearing all mask bits, causes all bits to be ignored and prevents a match because no difference can be detected. In this case address bus equivalence does not cause a match. Bytes that are not accessed are ignored. Thus when monitoring a multi byte field for a difference, partial accesses of the field only return a match if a difference is detected in the accessed bytes.

**Table 6-44. NDB and MASK bit dependency**

| NDB | DBGADM | Comment |
|---|---|---|
| 0 | 0 | Do not compare data bus bit. |
| 0 | 1 | Compare data bus bit. Match on equivalence. |
| 1 | 0 | Do not compare data bus bit. |
| 1 | 1 | Compare data bus bit. Match on difference. |

### 6.4.2.4 Range Comparisons

Range comparisons are accurate to byte boundaries. Thus for data access comparisons a match occurs if at least one byte of the access is in the range (inside range) or outside the range (outside range). For opcode comparisons only the address of the first opcode byte is compared with the range.

When using the AB comparator pair for a range comparison, the data bus can be used for qualification by using the comparator A data and data mask registers. Similarly when using the CD comparator pair for a range comparison, the data bus can be used for qualification by using the comparator C data and data mask registers. The DBGACTL/DBGCCTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access. The corresponding DBGBCTL/DBGDCTL bits are ignored. The DBGACTL/DBGCCTL COMPE/INST bits are used for range comparisons. The DBGBCTL/DBGDCTL COMPE/INST bits are ignored in range modes.

#### 6.4.2.4.1 Inside Range (CompAC_Addr ≤ address ≤ CompBD_Addr)

In the Inside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons by the control register (DBGC2). The match condition requires a simultaneous valid match for both comparators. A match condition on only one comparator is not valid.

#### 6.4.2.4.2 Outside Range (address < CompAC_Addr or address > CompBD_Addr)

In the Outside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons. A single match condition on either of the comparators is recognized as valid. Outside range mode in combination with opcode address matches can be used to detect if opcodes are from an unexpected range.

**NOTE**

When configured for data access matches, an outside range match would typically occur at any interrupt vector fetch or register access. This can be avoided by setting the upper or lower range limit to $FFFFFF or $000000 respectively. Interrupt vector fetches do not cause opcode address matches.

### 6.4.3 Events

Events are used as qualifiers for a state sequencer change of state. The state control register for the current state determines the next state for each event. An event can immediately initiate a transition to the next state sequencer state whereby the corresponding flag in DBGSR is set.

#### 6.4.3.1 Comparator Match Events

#### 6.4.3.1.1 Opcode Address Comparator Match

The comparator is loaded with the address of the selected instruction and the comparator control register INST bit is set. When the opcode reaches the execution stage of the instruction queue a match occurs just before the instruction executes, allowing a breakpoint immediately before the instruction boundary. The comparator address register must contain the address of the first opcode byte for the match to occur. Opcode address matches are data independent thus the RWE and RW bits are ignored. CPU compares are disabled when BDM becomes active.

#### 6.4.3.1.2 Data Access Comparator Match

Data access matches are generated when an access occurs at the address contained in the comparator address register. The match can be qualified by the access data and by the access type (read/write). The breakpoint occurs a maximum of 2 instructions after the access in the CPU flow. Note, if a COF occurs between access and breakpoint, the opcode address of the breakpoint can be elsewhere in the memory map.

Opcode fetches are not classed as data accesses. Thus data access matches are not possible on opcode fetches.

### 6.4.3.2 External Event

The DBGEEV input signal can force a state sequencer transition, independent of internal comparator matches. The DBGEEV is an input signal mapped directly to a device pin and configured by the EEVE field in DBGC1. The external events can change the state sequencer state, or force a trace buffer entry, or gate trace buffer entries.

If configured to change the state sequencer state, then the external match is mapped to DBGSCRx bits C3SC[1:0]. In this configuration, internal comparator channel3 is de-coupled from the state sequencer but can still be used for timestamps. The DBGEFR bit EEVF is set when an external event occurs.

### 6.4.3.3 Setting The TRIG Bit

Independent of comparator matches it is possible to initiate a tracing session and/or breakpoint by writing the TRIG bit in DBGC1 to a logic “1”. This forces the state sequencer into the Final State. If configured for End aligned tracing or for no tracing, the transition to Final State is followed immediately by a transition to State0. If configured for Begin- or Mid Aligned tracing, the state sequencer remains in Final State until tracing is complete, then it transitions to State0.

Breakpoints, if enabled, are issued on the transition to State0.

### 6.4.3.4 Profiling Trace Buffer Overflow Event

During code profiling a trace buffer overflow forces the state sequencer into the disarmed State0 and, if breakpoints are enabled, issues a breakpoint request to the CPU.

### 6.4.3.5 Event Priorities

If simultaneous events occur, the priority is resolved according to Table 6-45. Lower priority events are suppressed. It is thus possible to miss a lower priority event if it occurs simultaneously with an event of a higher priority. The event priorities dictate that in the case of simultaneous matches, the match on the higher comparator channel number (3,2,1,0) has priority.

If a write access to DBGC1 with the ARM bit position set occurs simultaneously to a hardware disarm from an internal event, then the ARM bit is cleared due to the hardware disarm.

**Table 6-45. Event Priorities**

| Priority | Source | Action |
|---|---|---|
| Highest | TB Overflow | Immediate force to state 0, generate breakpoint and terminate tracing |

**Table 6-45. Event Priorities**

| | | |
|---|---|---|
| | TRIG | Force immediately to final state |
| | DBGEEV | Force to next state as defined by state control registers (EEVE=2'b10) |
| | Match3 | Force to next state as defined by state control registers |
| | Match2 | Force to next state as defined by state control registers |
| | Match1 | Force to next state as defined by state control registers |
| Lowest | Match0 | Force to next state as defined by state control registers |

## 6.4.4 State Sequence Control

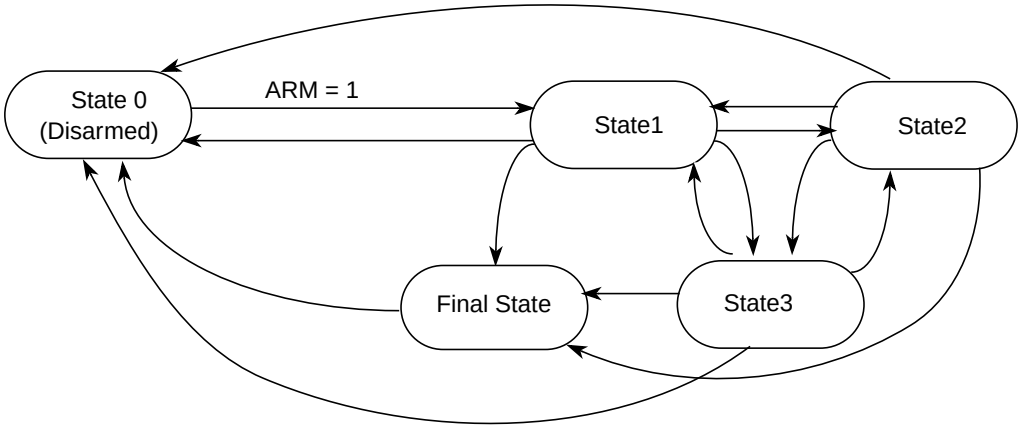



**Figure 6-26. State Sequencer Diagram**

The state sequencer allows a defined sequence of events to provide a breakpoint and/or a trigger point for tracing of data in the trace buffer. When the DBG module is armed by setting the ARM bit in the DBGC1 register, the state sequencer enters State1. Further transitions between the states are controlled by the state control registers and depend upon event occurrences (see Section 6.4.3). From Final State the only permitted transition is back to the disarmed State0. Transition between the states 1 to 3 is not restricted. Each transition updates the SSF[2:0] flags in DBGSR accordingly to indicate the current state. If breakpoints are enabled, then an event based transition to State0 generates the breakpoint request. A transition to State0 resulting from writing "0" to the ARM bit does not generate a breakpoint request.

### 6.4.4.1 Final State

On entering Final State a trigger may be issued to the trace buffer according to the trigger position control as defined by the TALIGN field (see Section 6.3.2.3").

If tracing is enabled and either Begin or Mid aligned triggering is selected, the state sequencer remains in Final State until completion of the trace. On completion of the trace the state sequencer returns to State0 and the debug module is disarmed; if breakpoints are enabled, a breakpoint request is generated.

If tracing is disabled or End aligned triggering is selected, then when the Final State is reached the state sequencer returns to State0 immediately and the debug module is disarmed. If breakpoints are enabled, a breakpoint request is generated on transitions to State0.

### 6.4.5 Trace Buffer Operation

The trace buffer is a 64 lines deep by 64-bits wide RAM array. If the TSOURCE bit is set the DBG module can store trace information in the RAM array in a circular buffer format. Data is stored in mode dependent formats, as described in the following sections. After each trace buffer entry, the counter register DBGCNT is incremented. Trace buffer rollover is possible when configured for End- or Mid-Aligned tracing, such that older entries are replaced by newer entries. Tracing of CPU activity is disabled when the BDC is active.

The RAM array can be accessed through the register DBGTB using 16-bit wide word accesses. After each read, the internal RAM pointer is incremented so that the next read will receive fresh information. Reading the trace buffer whilst the DBG is armed returns invalid data and the trace buffer pointer is not incremented.

In Detail mode the address range for CPU access tracing can be limited to a range specified by the TRANGE bits in DBGTCRH. This function uses comparators C and D to define an address range inside which accesses should be traced. Thus traced accesses can be restricted, for example, to particular register or RAM range accesses.

The external event pin can be configured to force trace buffer entries in Normal or Loop1 trace modes. All tracing modes support trace buffer gating. In Pure PC and Detail modes external events do not force trace buffer entries.

If the external event pin is configured to gate trace buffer entries then any trace mode is valid.

#### 6.4.5.1 Trace Trigger Alignment

Using the TALIGN bits (see Section 6.3.2.3”) it is possible to align the trigger with the end, the middle, or the beginning of a tracing session.

If End or Mid-Alignment is selected, tracing begins when the ARM bit in DBGC1 is set and State1 is entered. The transition to Final State if End-Alignment is selected, ends the tracing session. The transition to Final State if Mid-Alignment is selected signals that another 32 lines are traced before ending the tracing session. Tracing with Begin-Alignment starts at the trigger and ends when the trace buffer is full.

**Table 6-46. Tracing Alignment**

| TALIGN | Tracing Begin | Tracing End |
|---|---|---|
| 00 | On arming | At trigger |
| 01 | At trigger | When trace buffer is full |
| 10 | On arming | When 32 trace buffer lines have been filled after trigger |
| 11 | Reserved | |

##### 6.4.5.1.1 Storing with Begin-Alignment

Storing with Begin-Alignment, data is not stored in the trace buffer until the Final State is entered. Once the trigger condition is met the DBG module remains armed until 64 lines are stored in the trace buffer.

Using Begin-Alignment together with opcode address comparisons, if the instruction is about to be executed then the trace is started. If the trigger is at the address of a COF instruction, whilst tracing COF addresses, then that COF address is stored to the trace buffer. If breakpoints are enabled, the breakpoint is generated upon entry into State0 on completion of the tracing session; thus the breakpoint does not occur at the instruction boundary.

#### 6.4.5.1.2 Storing with Mid-Alignment

Storing with Mid-Alignment, data is stored in the trace buffer as soon as the DBG module is armed. When the trigger condition is met, another 32 lines are traced before ending the tracing session, irrespective of the number of lines stored before the trigger occurred, then the DBG module is disarmed and no more data is stored. Using Mid-Alignment with opcode address triggers, if the instruction is about to be executed then the trace is continued for another 32 lines. If breakpoints are enabled, the breakpoint is generated upon entry into State0 on completion of the tracing session; thus the breakpoint does not occur at the instruction boundary. When configured for Compressed Pure-PC tracing, the MAT info bit is set to indicate the last PC entry before a trigger event.

#### 6.4.5.1.3 Storing with End-Alignment

Storing with End-Alignment, data is stored in the trace buffer until the Final State is entered. Following this trigger, the DBG module immediately transitions to State0. If the trigger is at the address of a COF instruction the trigger event is not stored in the trace buffer.

### 6.4.5.2 Trace Modes

The DBG module can operate in four trace modes. The mode is selected using the TRCMOD bits in the DBGTCRH register. Normal, Loop1 and Detail modes can be configured to store a timestamp with each entry, by setting the STAMP bit. The modes are described in the following subsections.

In addition to the listed trace modes it is also possible to use code profiling to fill the trace buffer with a highly compressed COF format. This can be subsequently read out in the same fashion as the listed trace modes (see Section 6.4.6).

#### 6.4.5.2.1 Normal Mode

In Normal Mode, change of flow (COF) program counter (PC) addresses are stored.

CPU COF addresses are defined as follows:

- Source address of taken conditional branches (bit-conditional, and loop primitives)
- Destination address of indexed JMP and JSR instruction.s
- Destination address of RTI and RTS instructions.
- Vector address of interrupts

BRA, BSR, BGND as well as non-indexed JMP and JSR instructions are not classified as change of flow and are not stored in the trace buffer.

COF addresses stored include the full address bus of CPU and an information byte, which contains bits to indicate whether the stored address was a source, destination or vector address.

**NOTE**

When a CPU indexed jump instruction is executed, the destination address is stored to the trace buffer on instruction completion, indicating the COF has taken place. If an interrupt occurs simultaneously then the next instruction carried out is actually from the interrupt service routine. The instruction at the destination address of the original program flow gets executed after the interrupt service routine.

In the following example an IRQ interrupt occurs during execution of the indexed JMP at address MARK1. The NOP at the destination (SUB_1) is not executed until after the IRQ service routine but the destination address is entered into the trace buffer to indicate that the indexed JMP COF has taken place.

```
         LD      X,#SUB_1
MARK1:   JMP     (0,X)                  ; IRQ interrupt occurs during execution of this
MARK2:   NOP                            ;

SUB_1:   NOP                            ; JMP Destination address TRACE BUFFER ENTRY 1
                                        ; RTI Destination address TRACE BUFFER ENTRY 3
         NOP                            ;
ADDR1:   DBNE    D0,PART5               ; Source address TRACE BUFFER ENTRY 4

IRQ_ISR: LD      D1,#$F0                ; IRQ Vector $FFF2 = TRACE BUFFER ENTRY 2
         ST      D1,VAR_C1
         RTI                            ;
```

The execution flow taking into account the IRQ is as follows

```
         LD      X,#SUB_1
MARK1:   JMP     (0,X)                  ;
IRQ_ISR: LD      D1,#$F0                ;
         ST      D1,VAR_C1
         RTI                            ;
SUB_1:   NOP
         NOP                            ;
ADDR1:   DBNE    D0,PART5               ;
```

The Normal Mode trace buffer format is shown in the following tables. Whilst tracing in Normal or Loop1 modes each array line contains 2 data entries, thus in this case the DBGCNT[0] is incremented after each separate entry. Information byte bits indicate if an entry is a source, destination or vector address.

The external event input can force trace buffer entries independent of COF occurrences, in which case the EEVI bit is set and the PC value of the last instruction is stored to the trace buffer. If the external event coincides with a COF buffer entry a single entry is made with the EEVI bit set.

Normal mode profiling with timestamp is possible when tracing from a single source by setting the STAMP bit in DBGTCRL. This results in a different format (see Table 6-48).

**Table 6-47. Normal and Loop1 Mode Trace Buffer Format without Timestamp**

| Mode | 8-Byte Wide Trace Buffer Line | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Table 6-47. Normal and Loop1 Mode Trace Buffer Format without Timestamp**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| CPU | CINF1 | CPCH1 | CPCM1 | CPCL1 | CINF0 | CPCH0 | CPCM0 | CPCL0 |
| | CINF3 | CPCH3 | CPCM3 | CPCL3 | CINF2 | CPCH2 | CPCM2 | CPCL2 |

**Table 6-48. Normal and Loop1 Mode Trace Buffer Format with Timestamp**

| Mode | 8-Byte Wide Trace Buffer Line | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPU | Timestamp | Timestamp | Reserved | Reserved | CINF0 | CPCH0 | CPCM0 | CPCL0 |
| | Timestamp | Timestamp | Reserved | Reserved | CINF1 | CPCH1 | CPCM1 | CPCL1 |

CINF contains information relating to the CPU.

## CPU Information Byte CINF For Normal And Loop1 Modes

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CET | | 0 | 0 | CTI | EEVI | 0 | TOVF |

**Figure 6-27. CPU Information Byte CINF**

**Table 6-49. CINF Bit Descriptions**

| Field | Description |
|---|---|
| 7–6<br>CET | **CPU Entry Type Field** — Indicates the type of stored address of the trace buffer entry as described in Table 6-50 |
| 3<br>CTI | **Comparator Timestamp Indicator** — This bit indicates if the trace buffer entry corresponds to a comparator timestamp.<br>0 Trace buffer entry initiated by trace mode specification conditions or timestamp counter overflow<br>1 Trace buffer entry initiated by comparator D match |
| 2<br>EEVI | **External Event Indicator** — This bit indicates if the trace buffer entry corresponds to an external event.<br>0 Trace buffer entry not initiated by an external event<br>1 Trace buffer entry initiated by an external event |
| 0<br>TOVF | **Timestamp Overflow Indicator** — Indicates if the trace buffer entry corresponds to a timestamp overflow<br>0 Trace buffer entry not initiated by a timestamp overflow<br>1 Trace buffer entry initiated by a timestamp overflow |

**Table 6-50. CET Encoding**

| CET | Entry Type Description |
|---|---|
| 00 | Non COF opcode address (entry forced by an external event) |
| 01 | Vector destination address |
| 10 | Source address of COF opcode |
| 11 | Destination address of COF opcode |

#### 6.4.5.2.2 Loop1 Mode

Loop1 Mode, similarly to Normal Mode also stores only COF address information to the trace buffer, it however allows the filtering out of redundant information.

The intent of Loop1 Mode is to prevent the trace buffer from being filled entirely with duplicate information from a looping construct such as delays using the DBNE instruction. The DBG monitors trace buffer entries and prevents consecutive duplicate address entries resulting from repeated branches.

Loop1 Mode only inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses or vector addresses, since repeated entries of these could indicate a bug in application code that the DBG module is designed to help find.

The trace buffer format for Loop1 Mode is the same as that of Normal Mode.

#### 6.4.5.2.3 Detail Mode

When tracing CPU activity in Detail Mode, address and data of data and vector accesses are traced. The information byte indicates the size of access and the type of access (read or write).

ADRH, ADRM, ADRL denote address high, middle and low byte respectively. The numerical suffix indicates which tracing step. DBGCNT increments by 2 for each line completed.

If timestamps are enabled then each CPU entry can span 2 trace buffer lines, whereby the second line includes the timestamp. If a valid PC occurs in the same cycle as the timestamp, it is also stored to the trace buffer and the PC bit is set. The second line featuring the timestamp is only stored if no further data access occurs in the following cycle. This is shown in Table 6-52, where data accesses 2 and 3 occur in consecutive cycles, suppressing the entry2 timestamp. If 2 lines are used for an entry, then DBGCNT increments by 4. A timestamp line is indicated by bit1 in the TSINF byte. The timestamp counter is only reset each time a timestamp line entry is made. It is not reset when the data and address trace buffer line entry is made.

**Table 6-51. Detail Mode Trace Buffer Format without Timestamp**

| Mode | 8-Byte Wide Trace Buffer Line | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPU Detail | CDATA31 | CDATA21 | CDATA11 | CDATA01 | CINF1 | CADRH1 | CADRM1 | CADRL1 |
| | CDATA32 | CDATA22 | CDATA12 | CDATA02 | CINF2 | CADRH2 | CADRM2 | CADRL2 |

**Table 6-52. Detail Mode Trace Buffer Format with Timestamp**

| Mode | 8-Byte Wide Trace Buffer Line | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPU Detail | CDATA31 | CDATA21 | CDATA11 | CDATA01 | CINF1 | CADRH1 | CADRM1 | CADRL1 |
| | Timestamp | Timestamp | Reserved | Reserved | TSINF1 | CPCH1 | CPCM1 | CPCL1 |
| | CDATA32 | CDATA22 | CDATA12 | CDATA02 | CINF2 | CADRH2 | CADRM2 | CADRL2 |
| | CDATA33 | CDATA23 | CDATA13 | CDATA03 | CINF3 | CADRH3 | CADRM3 | CADRL3 |
| | Timestamp | Timestamp | Reserved | Reserved | TSINF3 | CPCH3 | CPCM3 | CPCL3 |

Detail Mode data entries store the bytes aligned to the address of the MSB accessed (Byte1 Table 6-53). Thus accesses split across 32-bit boundaries are wrapped around.

**Table 6-53. Detail Mode Data Byte Alignment**

| Access Address | Access Size | CDATA31 | CDATA21 | CDATA11 | CDATA01 |
|---|---|---|---|---|---|
| 00 | 32-bit | Byte1 | Byte2 | Byte3 | Byte4 |
| 01 | 32-bit | Byte4 | Byte1 | Byte2 | Byte3 |
| 10 | 32-bit | Byte3 | Byte4 | Byte1 | Byte2 |
| 11 | 32-bit | Byte2 | Byte3 | Byte4 | Byte1 |
| 00 | 24-bit | Byte1 | Byte2 | Byte3 | |
| 01 | 24-bit | | Byte1 | Byte2 | Byte3 |
| 10 | 24-bit | Byte3 | | Byte1 | Byte2 |
| 11 | 24-bit | Byte2 | Byte3 | | Byte1 |
| 00 | 16-bit | Byte1 | Byte2 | | |
| 01 | 16-bit | | Byte1 | Byte2 | |
| 10 | 16-bit | | | Byte1 | Byte2 |
| 11 | 16-bit | Byte2 | | | Byte1 |
| 00 | 8-bit | Byte1 | | | |
| 01 | 8-bit | | Byte1 | | |
| 10 | 8-bit | | | Byte1 | |
| 11 | 8-bit | | | | Byte1 |
| | | | Denotes byte that is not accessed. | | |

## Information Bytes

| BYTE | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| CINF | CSZ | | CRW | 0 | 0 | 0 | 0 | 0 |
| TSINF | 0 | 0 | 0 | 0 | CTI | PC | 1 | TOVF |

**Figure 6-28. Information Bytes CINF and XINF**

When tracing in Detail Mode, CINF provides information about the type of CPU access being made.

TSINF provides information about a timestamp. Bit1 indicates if the byte is a TSINF byte.

**Table 6-54. CINF Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>CSZ | **Access Type Indicator** — This field indicates the CPU access size.<br>00 8-bit Access<br>01 16-bit Access<br>10 24-bit Access<br>11 32-bit Access |
| 5<br>CRW | **Read/Write Indicator** — Indicates if the corresponding stored address corresponds to a read or write access.<br>0 Write Access<br>1 Read Access |

**Table 6-55. TSINF Field Descriptions**

| Field | Description |
|---|---|
| 3<br>CTI | **Comparator Timestamp Indicator** — This bit indicates if the trace buffer entry corresponds to a comparator timestamp.<br>0 Trace buffer entry initiated by trace mode specification conditions or timestamp counter overflow<br>1 Trace buffer entry initiated by comparator D match |
| 2<br>PC | **Program Counter Valid Indicator** — Indicates if the PC entry is valid on the timestamp line.<br>0 Trace buffer entry does not include PC value<br>1 Trace buffer entry includes PC value |
| 0<br>TOVF | **Timestamp Overflow Indicator** — Indicates if the trace buffer entry corresponds to a timestamp overflow<br>0 Trace buffer entry not initiated by a timestamp overflow<br>1 Trace buffer entry initiated by a timestamp overflow |

#### 6.4.5.2.4 Pure PC Mode

In Pure PC Mode, the PC addresses of all opcodes loaded into the execution stage, including illegal opcodes, are stored.

Tracing from a single source, compression is implemented to increase the effective trace depth. A compressed entry consists of the lowest PC byte only. A full entry consists of all PC bytes. If the PC remains in the same 256 byte range, then a compressed entry is made, otherwise a full entry is made. The full entry is always the last entry of a record.

Each trace buffer line consists of 7 payload bytes, PLB0-6, containing full or compressed CPU PC addresses and 1 information byte to indicate the type of entry (compressed or base address) for each payload byte.

Each trace buffer line is filled from right to left. The final entry on each line is always a base address, used as a reference for the previous entries on the same line. Whilst tracing, a base address is typically stored

in bytes[6:4], the other payload bytes may be compressed or complete addresses as indicated by the info byte bits.

**Table 6-56. Pure PC Mode Trace Buffer Format Single Source**

| Mode | 8-Byte Wide Trace Buffer Line | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPU | CXINF | BASE | BASE | BASE | PLB3 | PLB2 | PLB1 | PLB0 |

If the info bit for byte3 indicates a full CPU PC address, whereby bytes[5:3] are used, then the info bit mapped to byte[4] is redundant and the byte[6] is unused because a line overflow has occurred. Similarly a base address stored in bytes[4:2] causes line overflow, so bytes[6:5] are unused.

CXINF[6:4] indicate how many bytes in a line contain valid data, since tracing may terminate before a complete line has been filled.

### CXINF Information Byte Source Tracing

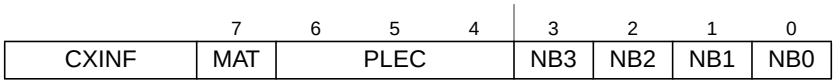



**Figure 6-29. Pure PC Mode CXINF**

**Table 6-57. CXINF Field Descriptions**

| Field | Description |
|---|---|
| MAT | **Mid Aligned Trigger**— This bit indicates a mid aligned trigger position. When a mid aligned trigger occurs, the next trace buffer entry is a base address and the counter is incremented to a new line, independent of the number of bytes used on the current line. The MAT bit is set on the current line, to indicate the position of the trigger. When configured for begin or end aligned trigger, this bit has no meaning.<br>NOTE: In the case when ARM and TRIG are simultaneously set together in the same cycle that a new PC value is registered, then this PC is stored to the same trace buffer line and MAT set.<br>0 Line filled without mid aligned trigger occurrence<br>1 Line last entry is the last PC entry before a mid aligned trigger |
| PLEC[2:0] | **Payload Entry Count**— This field indicates the number of valid bytes in the trace buffer line<br>Binary encoding is used to indicate up to 7 valid bytes. |
| NBx | **Payload Compression Indicator**— This field indicates if the corresponding payload byte is the lowest byte of a base PC entry<br>0 Corresponding payload byte is a not the lowest byte of a base PC entry<br>1 Corresponding payload byte is the lowest byte of a base PC entry |

Pure PC mode tracing does not support timestamps or external event entries.

## 6.4.5.3 Timestamp

When set, the STAMP bit in DBGTCRL configures the DBG to add a timestamp to trace buffer entries in Normal, Loop1 and Detail trace buffer modes. The timestamp is generated from a 16-bit counter and is stored to the trace buffer line each time a trace buffer entry is made.

The number of core clock cycles since the last entry equals the timestamp + 1. The core clock runs at twice the frequency of the bus clock. The timestamp of the first trace buffer entry is 0x0000. With timestamps enabled trace buffer entries are initiated in the following ways:

- according to the trace mode specification, for example COF PC addresses in Normal mode
- on a timestamp counter overflow
  If the timestamp counter reaches 0xFFFF then a trace buffer entry is made, with timestamp= 0xFFFF and the timestamp overflow bit TOVF is set.
- on a match of comparator D
  If STAMP and DSTAMP are set then comparator D is used for forcing trace buffer entries with timestamps. The state control register settings determine if comparator D is also used to trigger the state sequencer. Thus if the state control register configuration does not use comparator D, then it is used solely for the timestamp function. If comparator D initiates a timestamp then the CTI bit is set in the INFO byte. This can be used in Normal/Loop1 mode to indicate when a particular data access occurs relative to the PC flow. For example when the timing of an access may be unclear due to the use of indexes.

**NOTE**

If comparator D is configured to match a PC address then associated timestamps trigger a trace buffer entry during execution of the previous instruction. Thus the PC stored to the trace buffer is that of the previous instruction.The comparator must contain the PC address of the instruction's first opcode byte

Timestamps are disabled in Pure PC mode.

#### 6.4.5.4 Reading Data from Trace Buffer

The data stored in the trace buffer can be read using either the background debug controller (BDC) module or the CPU provided the DBG module is not armed and is configured for tracing by TSOURCE. When the ARM bit is set the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by an aligned word write to DBGTB when the module is disarmed. The trace buffer can only be read through the DBGTB register using aligned word reads. Reading the trace buffer while the DBG module is armed, or trace buffer locked returns 0xEE and no shifting of the RAM pointer occurs. Any byte or misaligned reads return 0xEE and do not cause the trace buffer pointer to increment to the next trace buffer address.

Reading the trace buffer is prevented by internal hardware whilst profiling is active because the RAM pointer is used to indicate the next row to be transmitted. Thus attempted reads of DBGTB do not return valid data when the PROFILE bit is set. To initialize the pointer and read profiling data, the PROFILE bit must be cleared and remain cleared.

The trace buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid 64-bit lines can be determined. DBGCNT does not decrement as data is read.

Whilst reading, an internal pointer is used to determine the next line to be read. After a tracing session, the pointer points to the oldest data entry, thus if no overflow has occurred, the pointer points to line0. The

pointer is initialized by each aligned write to DBGTB to point to the oldest data again. This enables an interrupted trace buffer read sequence to be easily restarted from the oldest data entry. After reading all trace buffer lines, the next read wraps around and returns the contents of line0.

The least significant word of each 64-bit wide array line is read out first. All bytes, including those containing invalid information are read out.

#### 6.4.5.5 Trace Buffer Reset State

The trace buffer contents are not initialized by a system reset. Thus should a system reset occur, the trace session information from immediately before the reset occurred can be read out. The DBGCNT bits are not cleared by a system reset. Thus should a reset occur, the number of valid lines in the trace buffer is indicated by DBGCNT. The internal pointer is cleared by a system reset. It can be initialized by an aligned word write to DBGTB following a reset during debugging, so that it points to the oldest valid data again. Debugging occurrences of system resets is best handled using mid or end trigger alignment since the reset may occur before the trace trigger, which in the begin trigger alignment case means no information would be stored in the trace buffer.

### 6.4.6 Code Profiling

#### 6.4.6.1 Code Profiling Overview

Code profiling supplies encoded COF information on the PDO pin and the reference clock on the PDOCLK pin. If the TSOURCE bit is set then code profiling is enabled by setting the PROFILE bit. The associated device pin is configured for code profiling by setting the PDOE bit. Once enabled, code profiling is activated by arming the DBG. During profiling, if PDOE is set, the PDO operates as an output pin at a half the internal bus frequency, driving both high and low.

Independent of PDOE status, profiling data is stored to the trace buffer and can be read out in the usual manner when the debug session ends and the PROFILE bit has been cleared.

The external debugger uses both edges of the clock output to strobe the data on PDO. The first PDOCLK edge is used to sample the first data bit on PDO.

**Figure 6-30. Profiling Output Interface**

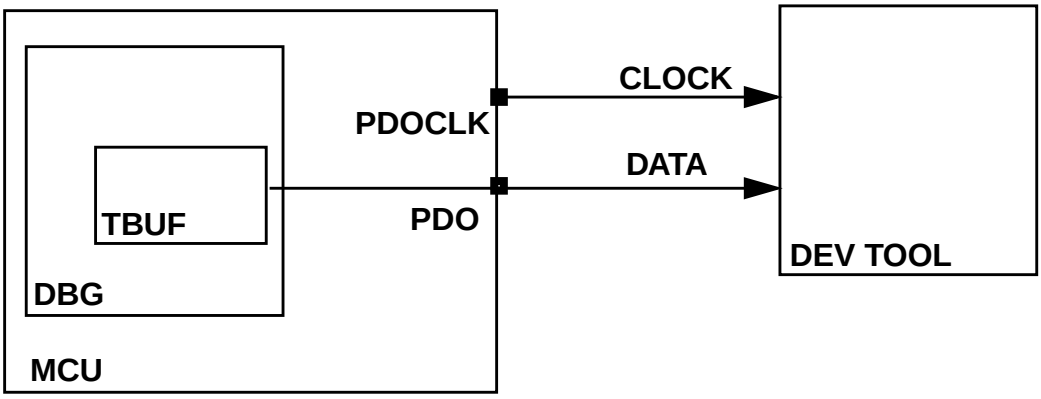
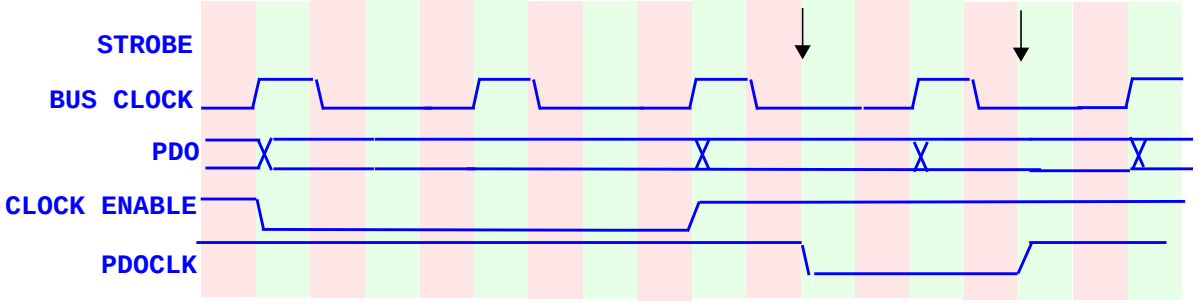


Figure 6-31 shows the profiling clock, PDOCLK, whose edges are offset from the bus clock, to ease setup and hold time requirements relative to PDO, which is synchronous to the bus clock.

**Figure 6-31. PDO Profiling Clock Control**




The trace buffer is used as a temporary storage medium to store COF information before it is transmitted. COF information can be transmitted whilst new information is written to the trace buffer. The trace buffer data is transmitted at PDO least significant bit first. After the first trace buffer entry is made, transmission begins in the first clock period in which no further data is written to the trace buffer.

If a trace buffer line transmission completes before the next trace buffer line is ready, then the clock output is held at a constant level until the line is ready for transfer.

### 6.4.6.2 Profiling Configuration, Alignment and Mode Dependencies

The PROFILE bit must be set and the DBG armed to enable profiling. Furthermore the PDOE bit must be set to configure the PDO and PDOCLK pins for profiling.

If TALIGN is configured for End-Aligned tracing then profiling begins as soon as the module is armed.

If TALIGN is configured for Begin-aligned tracing, then profiling begins when the state sequencer enters Final State and continues until a software disarm or trace buffer overflow occurs; thus profiling does not terminate after 64 line entries have been made.

Mid-Align tracing is not supported whilst profiling; if the TALIGN bits are configured for Mid-Align tracing when PROFILE is set, then the alignment defaults to end alignment.

Profiling entries continue until either a trace buffer overflow occurs or the DBG is disarmed by a state machine transition to State0. The profiling output transmission continues, even after disarming, until all trace buffer entries have been transmitted. The PTACT bit indicates if a profiling transmission is still active. The PTBOVF indicates if a trace buffer overflow has occurred.

The profiling timestamp feature is used only for the PTVB and PTW formats, thus differing from timestamps offered in other modes.

Profiling does not support trace buffer gating. The external pin gating feature is ignored during profiling.

When the DBG module is disarmed but profiling transmission is ongoing, register write accesses are suppressed.

When the DBG module is disarmed but profiling transmission is still ongoing, reading from the DBGTB returns the code 0xEE.

### 6.4.6.3 Code Profiling Internal Data Storage Format

When profiling starts, the first trace buffer entry is made to provide the start address. This uses a 4 byte format (PTS), including the INFO byte and a 3-byte PC start address. In order to avoid trace buffer overflow a fully compressed format is used for direct (conditional branch) COF information.

**Table 6-58. Profiling Trace buffer line format**

| Format | 8-Byte Wide Trace Buffer Line | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PTS | | | | | PC Start Address | | | INFO |
| PTIB | Indirect | Indirect | Indirect | Direct | Direct | Direct | Direct | INFO |
| PTHF | | | 0 | Direct | Direct | Direct | Direct | INFO |
| PTVB | Timestamp | Timestamp | Vector | Direct | Direct | Direct | Direct | INFO |
| PTW | Timestamp | Timestamp | 0 | Direct | Direct | Direct | Direct | INFO |

The INFO byte indicates the line format used. Up to 4 bytes of each line are dedicated to branch COFs. Further bytes are used for storing indirect COF information (indexed jumps and interrupt vectors). Indexed jumps force a full line entry with the PTIB format and require 3-bytes for the full 24-bit destination address. Interrupts force a full line entry with the PTVB format, whereby vectors are stored as a single byte and a 16-bit timestamp value is stored simultaneously to indicate the number of bus cycles relative to the previous COF. At each trace buffer entry the 16-bit timestamp counter is cleared. The device vectors use address[8:0] whereby address[1:0] are constant zero for vectors. Thus the value stored to the PTVB vector byte is equivalent to (Vector Address[8:1]).

After the PTS entry, the pointer increments and the DBG begins to fill the next line with direct COF information. This continues until the direct COF field is full or an indirect COF occurs, then the INFO byte and, if needed, indirect COF information are entered on that line and the pointer increments to the next line.

If a timestamp overflow occurs, indicating a 65536 bus clock cycles without COF, then an entry is made with the TSOVF bit set, INFO[6] (Table 6-59) and profiling continues.

If a trace buffer overflow occurs, a final entry is made with the TBOVF bit set, profiling is terminated and the DBG is disarmed. Trace buffer overflow occurs when the trace buffer contains 64 lines pending transmission.

Whenever the DBG is disarmed during profiling, a final entry is made with the TERM bit set to indicate the final entry.

When a final entry is made then by default the PTW line format is used, except if a COF occurs in the same cycle in which case the corresponding PTIB/PTVB/PTHF format is used. Since the development tool receives the INFO byte first, it can determine in advance the format of data it is about to receive. The

transmission of the INFO byte starts when a line is complete. Whole bytes are always transmitted. The grey shaded bytes of Table 6-58 are not transmitted.

**Figure 6-32. INFO byte encoding**

**Table 6-59. Profiling Format Encoding**

| INFO[3:0] | Line Format | Source | Description |
|---|---|---|---|
| 0000 | PTS | CPU | Initial CPU entry |
| 0001 | PTIB | CPU | Indexed jump with up to 31 direct COFs |
| 0010 | PTHF | CPU | 31 direct COFs without indirect COF |
| 0011 | PTVB | CPU | Vector with up to 31 direct COFs |
| 0111 | PTW | CPU | Error (Error codes in INFO[7:4]) |
| Others | Reserved | CPU | Reserved |
| **INFO[7:4]** | **Bit Name** | | **Description** |
| INFO[7] | Reserved | CPU | Reserved |
| INFO[6] | TSOVF | CPU | Timestamp Overflow |
| INFO[5] | TBOVF | CPU | Trace Buffer Overflow |
| INFO[4] | TERM | CPU | Profiling terminated by disarming |
| Vector[7:0] | Vector[7:0] | CPU | Device Interrupt Vector Address [8:1] |

### 6.4.6.4 Direct COF Compression

Each branch COF is stored to the trace buffer as a single bit (0=branch not taken, 1=branch taken) until an indirect COF (indexed jump, return, or interrupt) occurs. The branch COF entries are stored in the byte fields labelled “Direct” in Table 6-58. These entries start at byte1[0] and continue through to byte4[7], or until an indirect COF occurs, whichever occurs sooner. The entries use a format whereby the left most asserted bit is always the stop bit, which indicates that the bit to its right is the first direct COF and byte1[0] is the last COF that occurred before the indirect COF. This is shown in Table 6-60, whereby the Bytes 4 to 1 of the trace buffer are shown for 3 different cases. The stop bit field for each line is shaded.

In line0, the left most asserted bit is Byte4[7]. This indicates that all remaining 31 bits in the 4-byte field contain valid direct COF information, whereby each 1 represents branch taken and each 0 represents branch not taken. The stop bit of line1 indicates that all 30 bits to it’s right are valid, after the 30th direct COF entry, an indirect COF occurred, that is stored in bytes 7 to 5. In this case the bit to the left of the stop bit is redundant. Line2 indicates that an indirect COF occurred after 8 direct COF entries. The indirect COF address is stored in bytes 7 to 5. All bits to the left of the stop bit are redundant.

| Line | Byte4 | | | | | | | | Byte3 | | | | | | | | Byte2 | | | | | | | | Byte1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Line0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Line1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Line2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

**Table 6-60. Profiling Direct COF Format**

## 6.4.7 Breakpoints

Breakpoints can be generated by state sequencer transitions to State0. Transitions to State0 are forced by the following events

- Through comparator matches via Final State.
- Through software writing to the TRIG bit in the DBGC1 register via Final State.
- Through the external event input (DBGEEV) via Final State.
- Through a profiling trace buffer overflow event.

Breakpoints are not generated by software writes to DBGC1 that clear the ARM bit.

### 6.4.7.1 Breakpoints From Comparator Matches or External Events

Breakpoints can be generated when the state sequencer transitions to State0 following a comparator match or an external event.

If a tracing session is selected by TSOURCE, the transition to State0 occurs when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace. If End aligned tracing or no tracing session is selected, the transition to State0 and associated breakpoints are immediate.

### 6.4.7.2 Breakpoints Generated Via The TRIG Bit

When TRIG is written to “1”, the Final State is entered. If a tracing session is selected by TSOURCE, State0 is entered and breakpoints are requested only when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace. If no tracing session is selected, the state sequencer enters State0 immediately and breakpoints are requested. TRIG breakpoints are possible even if the DBG module is disarmed.

### 6.4.7.3 DBG Breakpoint Priorities

If a TRIG occurs after Begin or Mid aligned tracing has already been triggered by a comparator instigated transition to Final State, then TRIG no longer has an effect. When the associated tracing session is complete, the breakpoint occurs. Similarly if a TRIG is followed by a subsequent comparator match, it has no effect, since tracing has already started.

#### 6.4.7.3.1 DBG Breakpoint Priorities And BDC Interfacing

Breakpoint operation is dependent on the state of the S12ZBDC module. BDM cannot be entered from a breakpoint unless the BDC is enabled (ENBDC bit is set in the BDC). If BDM is already active, breakpoints are disabled. In addition, while executing a BDC STEP1 command, breakpoints are disabled.

When the DBG breakpoints are mapped to BDM (BDMBP set), then if a breakpoint request, either from a BDC BACKGROUND command or a DBG event, coincides with an SWI instruction in application code, (i.e. the DBG requests a breakpoint at the next instruction boundary and the next instruction is an SWI) then the CPU gives priority to the BDM request over the SWI request.

On returning from BDM, the SWI from user code gets executed. Breakpoint generation control is summarized in Table 6-61.

**Table 6-61. Breakpoint Mapping Summary**

| BRKCPU | BDMBP Bit (DBGC1[4]) | BDC Enabled | BDM Active | Breakpoint Mapping |
|---|---|---|---|---|
| 0 | X | X | X | No Breakpoint |
| 1 | 0 | X | 0 | Breakpoint to SWI |
| 1 | 0 | 1 | 1 | No Breakpoint |
| 1 | 1 | 0 | X | No Breakpoint |
| 1 | 1 | 1 | 0 | Breakpoint to BDM |
| 1 | 1 | 1 | 1 | No Breakpoint |

## 6.5 Application Information

### 6.5.1 Avoiding Unintended Breakpoint Re-triggering

Returning from an instruction address breakpoint using an RTI or BDC GO command without PC modification, returns to the instruction that generated the breakpoint. If an active breakpoint or trigger still exists at that address, this can re-trigger, disarming the DBG. If configured for BDM breakpoints, the user must apply the BDC STEP1 command to increment the PC past the current instruction.

If configured for SWI breakpoints, the DBG can be re configured in the SWI routine. If a comparator match occurs at an SWI vector address then a code SWI and DBG breakpoint SWI could occur simultaneously. In this case the SWI routine is executed twice before returning.

### 6.5.2 Debugging Through Reset

To debug through reset, the debugger can recognize a reset occurrence and pull the device BKGD pin low. This forces the device to leave reset in special single chip (SSC) mode, because the BKGD pin is used as the MODC signal in the reset phase. When the device leaves reset in SSC mode, CPU execution is halted and the device is in active BDM. Thus the debugger can configure the DBG for tracing and breakpoints before returning to application code execution. In this way it is possible to analyze the sequence of events emerging from reset. The recommended handling of the internal reset scenario is as follows:

- When a reset occurs the debugger pulls BKGD low until the reset ends, forcing SSC mode entry.
- Then the debugger reads the reset flags to determine the cause of reset.
- If required, the debugger can read the trace buffer to see what happened just before reset. Since the trace buffer and DBGCNT register are not affected by resets other than POR.
- The debugger configures and arms the DBG to start tracing on returning to application code.
- The debugger then sets the PC according to the reset flags.
- Then the debugger returns to user code with GO or STEP1.

### 6.5.3 Breakpoints from other S12Z sources

The DBG is neither affected by CPU BGND instructions, nor by BDC BACKGROUND commands.

### 6.5.4 Code Profiling

The code profiling data output pin PDO is mapped to a device pin that can also be used as GPIO in an application. If profiling is required and all pins are required in the application, it is recommended to use the device pin for a simple output function in the application, without feedback to the chip. In this way the application can still be profiled, since the pin has no effect on code flow.

The PDO provides a simple bit stream that must be strobed at both edges of the profiling clock when profiling. The external development tool activates profiling by setting the DBG ARM bit, with PROFILE and PDOE already set. Thereafter the first bit of the profiling bit stream is valid at the first rising edge of the profiling clock. No start bit is provided. The external development tool must detect this first rising edge after arming the DBG. To detect the end of profiling, the DBG ARM bit can be monitored using the BDC.

# Chapter 7
# S12 Clock, Reset and Power Management Unit (S12CPMU_UHV_V5)

## Revision History

| Rev. No. (Item No) | Date (Submitted By) | Sections Affected | Substantial Change(s) |
|---|---|---|---|
| V01.00 | 22.Dec. 10 | | Initial Version. |
| V02.00 | 08. Apr. 11 | Section 7.1.2.3/7-227<br>Section 7.1.2.4/7-228<br>Section 7.1.3/7-229<br>Section 7.3.1/7-233<br>Section 7.3.2.7/7-242<br>Section 7.3.2.19/7-261<br>Section 7.4.3/7-275<br>Section 7.4.4/7-276<br>Section 7.5.5/7-281<br>Section 7.7.2/7-285<br>Table 7-6<br>Table 7-15<br>Table 7-34<br>Figure 7-1<br>Figure 7-3<br>Figure 7-10 | Added bit CSAD (COP in Stop Mode ACLK Disable) in register CPMUCLKS. This bit allows halting the COP in Stop Mode (Full or Pseudo) when ACLK is the COP clock source. Description of Stop Modes, Block Diagram, CPMUCLKS register and COP Watchdog feature are updated. |
| V04.00 | 16 May 2011 | | • Removed output BDM clock (equals VCOCLK divided by 8)<br>• Extended PLL VCO range to 100MHz (CPMUSYNR register, VCOFRQ[1:0] bits) |
| V04.01 | 19 May 2011 | | • Changed Register Addresses (now with Offset)<br>• Block Diagram: added outputs OSCCLK and IRCCLK |
| V04.02 | 06 June 2011 | | • Voltage Regulator: created Pin description, changed CPMU Block Diagram accordingly<br>• Added CPMUVREGCTL register with Bits INTXON, EXTXON, EXTCON |
| V04.03 | 09 June 2011 | | • now only ONE reset vector for ALL reset sources<br>• added Reset Flags: COPRF, OMRF to Block Diagram and CPMURFLG register<br>• added Reset Flags Register CPMURFLG for all reset flags<br>• renamed CPMUFLG to CPMUIFLG (I= Interrupt flags)<br>• removed reset vectors for Oscillator Clock monitor reset and COP reset |
| V04.04 | 4 July 2011 | | • added PLL clock monitor reset |

| Rev. No. (Item No) | Date (Submitted By) | Sections Affected | Substantial Change(s) |
|---|---|---|---|
| V04.05 | 12 July 2011 | | • corrected frequencies in Figure 7-36. Startup of clock system after Reset to 12.5MHz and 25MHz<br>• changed all “ATD” references to “ADC”<br>• Corrected ACLK frequency “trimmed to 20KHz” |
| V04.06 | 26 July 2011 | | • CPMUOSC register: added reserved bit 5. |
| V04.07 | 19 Aug. 2011 | | • Corrected typos. Minor spelling corrections and clarifications |
| V04.08 | 18 Nov. 2011 | | • Corrected CPMURFLG address<br>• Added PMRF bit in memory map overview<br>• Removed illegal address reset<br>• VDD pin: added monitored by Power on Reset circuit |
| V04.09 | 25 Jan. 2012 | | • Minor format and syntax corrections<br>• Added statement to clarify RTI and COP in freeze mode |
| V05.00 | 14 Feb. 2012 | | • Removed EXTCON bit and VDDC supply<br>• Removed VDD and VDDF supply pins. Now VDD and VDDF are device internal |
| V05.01 | 16 March 2012 | | • minor syntax and typo corrections<br>• Added full swing pierce oscillator (OSCMOD bit in CPMUOSC2 register)<br>• Added oscillator clock monitor reset to be configurable with OMRE bit (CPMUOSC2 register)<br>• |
| V05.02 | 19 March 2012 | | • Application section: added C-code example for PLL and Oscillator configuration and startup |
| V05.03 | 28 March 2012 | | • CPMUPOSTDIV register and post divider: added how many bus cycles it takes until target frequency is reached. |
| V05.04 | 10 July 2012 | | • added functional safety tagging |
| V05.05 | 25 Sept. 2012 | | • In Table 7-2 “VCOCLK frequency selection” specified a maximum allowed value of 64MHz. Remaining VCOFRQ0/1 combinations specified as “Reserved”. |
| V05.06 | 20 Dec. 2012 | | • Format and font corrections<br>• Table 7-28. CPMUOSC Field Descriptions:removed Bit6 and Bit4-0 description as these bits no longer exist. |
| V05.07 | 25 Feb. 2013 | | • Format and font corrections<br>• added “_V5” to block name |
| V05.08 | 3 April 2013 | | • Format and font corrections |
| V05.09 | 27 June 2013 | | • PMRF register Bit: corrected description<br>• some wording improvements |
| V05.10 | 13 Aug.2013 | | • Signal description: added that availability of BCTL pin is defined in device specification<br>• some wording improvements and details |

| Rev. No. (Item No) | Date (Submitted By) | Sections Affected | Substantial Change(s) |
|---|---|---|---|
| V05.11 | 21 Aug.2013 | | • changed frequency upper limit of external Pierce Oscillator (XOSCLCP) from 16MHz to 20MHz<br>• $f_{PLLRST}$ changed to $f_{VCORST}$<br>• correct bit numbering for CSAD Bit<br>• corrected typo in heading of CPMUOSC**2** Field Description |

## 7.1 Introduction

This specification describes the function of the Clock, Reset and Power Management Unit (S12CPMU_UHV_V5).

- The Pierce oscillator (XOSCLCP) provides a robust, low-noise and low-power external clock source. It is designed for optimal start-up margin with typical crystal oscillators.
- The Voltage regulator (VREGAUTO) operates from the range 6V to 18V. It provides all the required chip internal voltages and voltage monitors.
- The Phase Locked Loop (PLL) provides a highly accurate frequency multiplier with internal filter.
- The Internal Reference Clock (IRC1M) provides a 1MHz internal clock.

### 7.1.1 Features

The Pierce Oscillator (XOSCLCP) contains circuitry to dynamically control current gain in the output amplitude. This ensures a signal with low harmonic distortion, low power and good noise immunity.

- Supports crystals or resonators from 4MHz to 20MHz.
- High noise immunity due to input hysteresis and spike filtering.
- Low RF emissions with peak-to-peak swing limited dynamically
- Transconductance (gm) sized for optimum start-up margin for typical crystals
- Dynamic gain control eliminates the need for external current limiting resistor
- Integrated resistor eliminates the need for external bias resistor
- Low power consumption: Operates from internal 1.8V (nominal) supply, Amplitude control limits power
- Optional oscillator clock monitor reset
- Optional full swing mode for higher immunity against noise injection on the cost of higher power consumption and increased emission

The Voltage Regulator (VREGAUTO) has the following features:

- Input voltage range from 6 to 18V (nominal operating range)
- Low-voltage detect (LVD) with low-voltage interrupt (LVI)
- Power-on reset (POR)
- Low-voltage reset (LVR)
- On Chip Temperature Sensor and Bandgap Voltage measurement via internal ADC channel.
- Voltage Regulator providing Full Performance Mode (FPM) and Reduced Performance Mode (RPM)
- External ballast device support to reduce internal power dissipation
- Capable of supplying both the MCU internally plus external components
- Over-temperature interrupt

The Phase Locked Loop (PLL) has the following features:

- Highly accurate and phase locked frequency multiplier
- Configurable internal filter for best stability and lock time
- Frequency modulation for defined jitter and reduced emission
- Automatic frequency lock detector
- Interrupt request on entry or exit from locked condition
- PLL clock monitor reset
- Reference clock either external (crystal) or internal square wave (1MHz IRC1M) based.
- PLL stability is sufficient for LIN communication in slave mode, even if using IRC1M as reference clock

The Internal Reference Clock (IRC1M) has the following features:

- Frequency trimming
  (A factory trim value for 1MHz is loaded from Flash Memory into the CPMUIRCTRIMH and CPMUIRCTRIML registers after reset, which can be overwritten by application if required)
- Temperature Coefficient (TC) trimming.
  (A factory trim value is loaded from Flash Memory into the IRCTRIM register to turn off TC trimming after reset. Application can trim the TC if required by overwriting the IRCTRIM register).

Other features of the S12CPMU_UHV_V5 include

- Oscillator clock monitor to detect loss of crystal
- Autonomous periodical interrupt (API)
- Bus Clock Generator
  - Clock switch to select either PLLCLK or external crystal/resonator based Bus Clock
  - PLLCLK divider to adjust system speed
- System Reset generation from the following possible sources:
  - Power-on reset (POR)
  - Low-voltage reset (LVR)
  - COP system watchdog, COP reset on time-out, windowed COP
  - Loss of oscillation (Oscillator clock monitor fail)
  - Loss of PLL clock (PLL clock monitor fail)
  - External pin $\overline{\text{RESET}}$

## 7.1.2 Modes of Operation

This subsection lists and briefly describes all operating modes supported by the S12CPMU_UHV_V5.

### 7.1.2.1 Run Mode

The voltage regulator is in Full Performance Mode (FPM).

**NOTE**

The voltage regulator is active, providing the nominal supply voltages with full current sourcing capability (see also Appendix for VREG electrical parameters). The features ACLK clock source, Low Voltage Interrupt (LVI), Low Voltage Reset (LVR) and Power-On Reset (POR) are available.

The Phase Locked Loop (PLL) is on.

The Internal Reference Clock (IRC1M) is on.

The API is available.

- **PLL Engaged Internal (PEI)**
  - This is the default mode after System Reset and Power-On Reset.
  - The Bus Clock is based on the PLLCLK.
  - After reset the PLL is configured for 50MHz VCOCLK operation.
    Post divider is 0x03, so PLLCLK is VCOCLK divided by 4, that is 12.5MHz and Bus Clock is 6.25MHz.
    The PLL can be re-configured for other bus frequencies.
  - The reference clock for the PLL (REFCLK) is based on internal reference clock IRC1M.
- **PLL Engaged External (PEE)**
  - The Bus Clock is based on the PLLCLK.
  - This mode can be entered from default mode PEI by performing the following steps:
    - Configure the PLL for desired bus frequency.
    - Program the reference divider (REFDIV[3:0] bits) to divide down oscillator frequency if necessary.
    - Enable the external oscillator (OSCE bit).
    - Wait for oscillator to start up (UPOSC=1) and PLL to lock (LOCK=1).
- **PLL Bypassed External (PBE)**
  - The Bus Clock is based on the Oscillator Clock (OSCCLK).
  - The PLLCLK is always on to qualify the external oscillator clock. Therefore it is necessary to make sure a valid PLL configuration is used for the selected oscillator frequency.
  - This mode can be entered from default mode PEI by performing the following steps:
    - Make sure the PLL configuration is valid for the selected oscillator frequency.

- Enable the external oscillator (OSCE bit).
- Wait for oscillator to start up (UPOSC=1).
- Select the Oscillator Clock (OSCCLK) as source of the Bus Clock (PLLSEL=0).

— The PLLCLK is on and used to qualify the external oscillator clock.

### 7.1.2.2 Wait Mode

For S12CPMU_UHV_V5 Wait Mode is the same as Run Mode.

### 7.1.2.3 Stop Mode

This mode is entered by executing the CPU STOP instruction.

The voltage regulator is in Reduced Performance Mode (RPM).

**NOTE**

The voltage regulator output voltage may degrade to a lower value than in Full Performance Mode (FPM), additionally the current sourcing capability is substantially reduced (see also Appendix for VREG electrical parameters). Only clock source ACLK is available and the Power On Reset (POR) circuitry is functional. The Low Voltage Interrupt (LVI) and Low Voltage Reset (LVR) are disabled.

The API is available.

The Phase Locked Loop (PLL) is off.

The Internal Reference Clock (IRC1M) is off.

Core Clock and Bus Clock are stopped.

Depending on the setting of the PSTP and the OSCE bit, Stop Mode can be differentiated between Full Stop Mode (PSTP = 0 or OSCE=0) and Pseudo Stop Mode (PSTP = 1 and OSCE=1). In addition, the behavior of the COP in each mode will change based on the clocking method selected by COPOSCSEL[1:0].

- **Full Stop Mode (PSTP = 0 or OSCE=0)**
  External oscillator (XOSCLCP) is disabled.
  — If COPOSCSEL1=0:
  The COP and RTI counters halt during Full Stop Mode.
  After wake-up from Full Stop Mode the Core Clock and Bus Clock are running on PLLCLK (PLLSEL=1). COP and RTI are running on IRCCLK (COPOSCSEL0=0, RTIOSCSEL=0).
  — If COPOSCSEL1=1:
  The clock for the COP is derived from ACLK (trimmable internal RC-Oscillator clock). During Full Stop Mode the ACLK for the COP can be stopped (COP static) or running (COP active) depending on the setting of bit CSAD. When bit CSAD is set the ACLK clock source for the COP is stopped during Full Stop Mode and COP continues to operate after exit from Full Stop

Mode. For this COP configuration (ACLK clock source, CSAD set) a latency time occurs when entering or exiting (Full, Pseudo) Stop Mode. When bit CSAD is clear the ACLK clock source is on for the COP during Full Stop Mode and COP is operating.
During Full Stop Mode the RTI counter halts.
After wake-up from Full Stop Mode the Core Clock and Bus Clock are running on PLLCLK (PLLSEL=1). The COP runs on ACLK and RTI is running on IRCCLK (COPOSCSEL0=0, RTIOSCSEL=0).

- **Pseudo Stop Mode (PSTP = 1 and OSCE=1)**
  External oscillator (XOSCLCP) continues to run.
  - If COPOSCSEL1=0:
    If the respective enable bits are set (PCE=1 and PRE=1) the COP and RTI will continue to run with a clock derived from the oscillator clock.
    The clock configuration bits PLLSEL, COPOSCSEL0, RTIOSCSEL are unchanged.
  - If COPOSCSEL1=1:
    If the respective enable bit for the RTI is set (PRE=1) the RTI will continue to run with a clock derived from the oscillator clock.
    The clock for the COP is derived from ACLK (trimmable internal RC-Oscillator clock). During Pseudo Stop Mode the ACLK for the COP can be stopped (COP static) or running (COP active) depending on the setting of bit CSAD. When bit CSAD is set the ACLK for the COP is stopped during Pseudo Stop Mode and COP continues to operate after exit from Pseudo Stop Mode.
    For this COP configuration (ACLK clock source, CSAD set) a latency time occurs when entering or exiting (Pseudo, Full) Stop Mode. When bit CSAD is clear the ACLK clock source is on for the COP during Pseudo Stop Mode and COP is operating.
    The clock configuration bits PLLSEL, COPOSCSEL0, RTIOSCSEL are unchanged.

**NOTE**

When starting up the external oscillator (either by programming OSCE bit to 1 or on exit from Full Stop Mode with OSCE bit already 1) the software must wait for a minimum time equivalent to the startup-time of the external oscillator $t_{UPOSC}$ before entering Pseudo Stop Mode.

### 7.1.2.4 Freeze Mode (BDM active)

For S12CPMU_UHV_V5 Freeze Mode is the same as Run Mode except for RTI and COP which can be frozen in Active BDM Mode with the RSBCK bit in the CPMUCOP register. After exiting BDM Mode RTI and COP will resume its operations starting from this frozen status.

Additionally the COP can be forced to the maximum time-out period in Active BDM Mode. For details please see also the RSBCK and CR[2:0] bit description field of Table 7-13 in Section 7.3.2.10, “S12CPMU_UHV_V5 COP Control Register (CPMUCOP)

## 7.1.3 S12CPMU_UHV_V5 Block Diagram

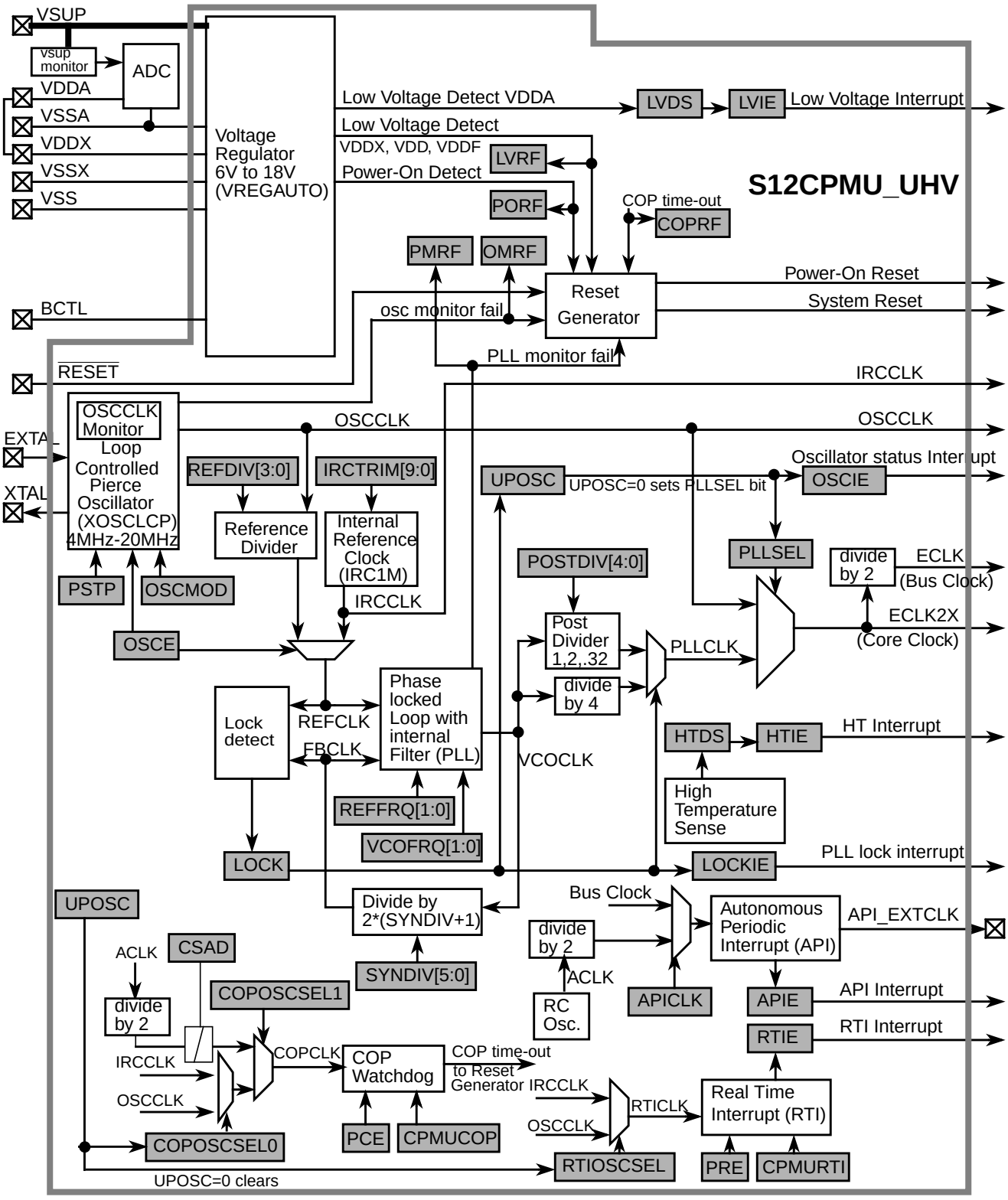



**Figure 7-1. Block diagram of S12CPMU_UHV_V5**

Figure 7-2 shows a block diagram of the XOSCLCP.

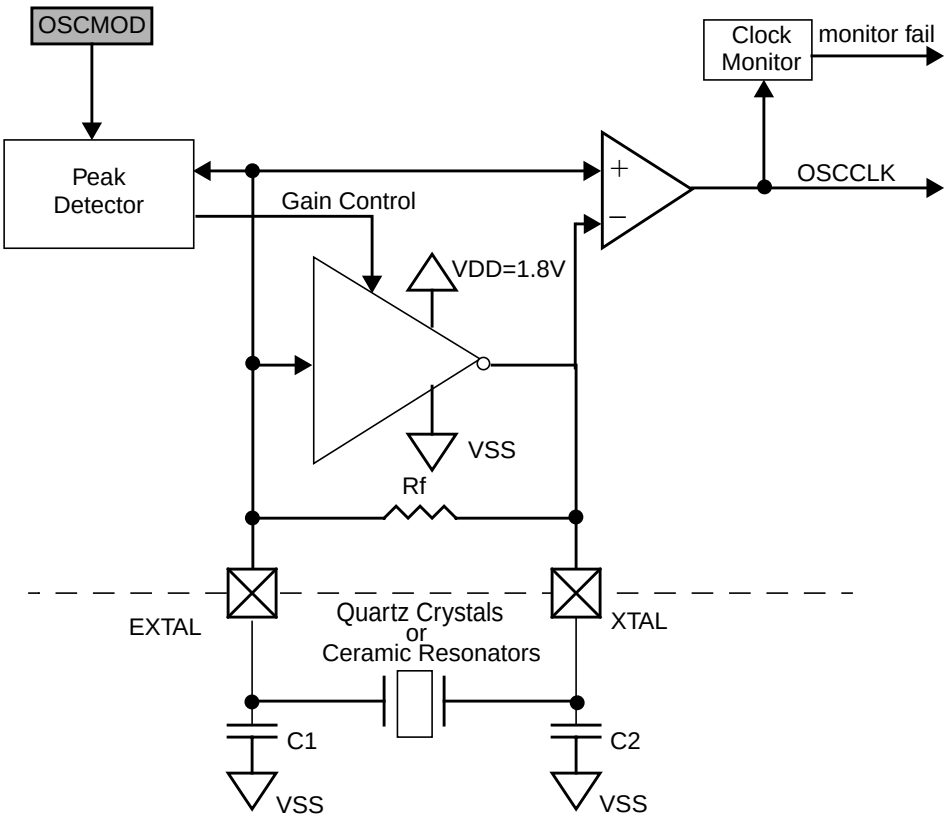



**Figure 7-2. XOSCLCP Block Diagram**

## 7.2 Signal Description

This section lists and describes the signals that connect off chip as well as internal supply nodes and special signals.

### 7.2.1 $\overline{\text{RESET}}$

Pin $\overline{\text{RESET}}$ is an active-low bidirectional pin. As an input it initializes the MCU asynchronously to a known start-up state. As an open-drain output it indicates that an MCU-internal reset has been triggered.

### 7.2.2 EXTAL and XTAL

These pins provide the interface for a crystal to control the internal clock generator circuitry. EXTAL is the input to the crystal oscillator amplifier. XTAL is the output of the crystal oscillator amplifier. If XOSCLCP is enabled, the MCU internal OSCCLK_LCP is derived from the EXTAL input frequency. If OSCE=0, the EXTAL pin is pulled down by an internal resistor of approximately 200 kΩ and the XTAL pin is pulled down by an internal resistor of approximately 700 kΩ.

**NOTE**

Freescale recommends an evaluation of the application board and chosen resonator or crystal by the resonator or crystal supplier.
The loop controlled circuit (XOSCLCP) is not suited for overtone resonators and crystals.

### 7.2.3 VSUP — Regulator Power Input Pin

Pin VSUP is the power input of VREGAUTO. All currents sourced into the regulator loads flow through this pin.

A suitable reverse battery protection network can be used to connect VSUP to the car battery supply network.

### 7.2.4 VDDA, VSSA — Regulator Reference Supply Pins

Pins VDDA and VSSA are used to supply the analog parts of the regulator. Internal precision reference circuits are supplied from these signals.

An off-chip decoupling capacitor (220 nF(X7R ceramic)) between VDDA and VSSA is required and can improve the quality of this supply.

VDDA has to be connected externally to VDDX.

### 7.2.5 VDDX, VSSX — Pad Supply Pins

VDDX is the supply domain for the digital Pads.

An off-chip decoupling capacitor (10µF plus 220 nF(X7R ceramic)) between VDDX and VSSX is required.

This supply domain is monitored by the Low Voltage Reset circuit.

VDDX has to be connected externally to VDDA.

### 7.2.6 BCTL — Base Control Pin for external PNP

BCTL is the ballast connection for the on chip voltage regulator. It provides the base current of an external

BJT (PNP) of the VDDX and VDDA supplies. An additional 5.6KΩ resistor between emitter and base of the BJT is required. See the device specification if this pin is available on this device.

### 7.2.7 VSS — Core Logic Ground Pin

VSS is the core logic supply return pin. It must be grounded.

### 7.2.8 VDD — Internal Regulator Output Supply (Core Logic)

Node VDD is a device internal supply output of the voltage regulator that provides the power supply for the internal core logic.

This supply domain is monitored by the Low Voltage Reset circuit and The Power On Reset circuit.

### 7.2.9 VDDF — Internal Regulator Output Supply (NVM Logic)

Node VDDF is a device internal supply output of the voltage regulator that provides the power supply for the NVM logic.

This supply domain is monitored by the Low Voltage Reset circuit.

### 7.2.10 API_EXTCLK — API external clock output pin

This pin provides the signal selected via APIES and is enabled with APIEA bit. See the device specification if this clock output is available on this device and to which pin it might be connected.

### 7.2.11 TEMPSENSE — Internal Temperature Sensor Output Voltage

Depending on the VSEL setting either the voltage level generated by the temperature sensor or the VREG bandgap voltage is driven to a special channel input of the ADC Converter. See device level specification for connectivity of ADC special channels.

## 7.3 Memory Map and Registers

This section provides a detailed description of all registers accessible in the S12CPMU_UHV_V5.

### 7.3.1 Module Memory Map

The S12CPMU_UHV_V5 registers are shown in Figure 7-3.

| Address Offset | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | CPMU RESERVED00 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0001 | CPMU RESERVED01 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0002 | CPMU RESERVED02 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0003 | CPMURFLG | R | 0 | PORF | LVRF | 0 | COPRF | 0 | OMRF | PMRF |
| | | W | | | | | | | | |
| 0x0004 | CPMU SYNR | R | VCOFRQ[1:0] | | SYNDIV[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x0005 | CPMU REFDIV | R | REFFRQ[1:0] | | 0 | 0 | REFDIV[3:0] | | | |
| | | W | | | | | | | | |
| 0x0006 | CPMU POSTDIV | R | 0 | 0 | 0 | POSTDIV[4:0] | | | | |
| | | W | | | | | | | | |
| 0x0007 | CPMUIFLG | R | RTIF | 0 | 0 | LOCKIF | LOCK | 0 | OSCIF | UPOSC |
| | | W | | | | | | | | |
| 0x0008 | CPMUINT | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | OSCIE | 0 |
| | | W | | | | | | | | |
| 0x0009 | CPMUCLKS | R | PLLSEL | PSTP | CSAD | COP OSCSEL1 | PRE | PCE | RTI OSCSEL | COP OSCSEL0 |
| | | W | | | | | | | | |
| 0x000A | CPMUPLL | R | 0 | 0 | FM1 | FM0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000B | CPMURTI | R | RTDEC | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 |
| | | W | | | | | | | | |
| 0x000C | CPMUCOP | R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 |
| | | W | | | WRTMASK | | | | | |
| 0x000D | RESERVED CPMUTEST0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000E | RESERVED CPMUTEST1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| | | | = Unimplemented or Reserved | | | | | | | |

**Figure 7-3. CPMU Register Summary**

| Address Offset | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000F | CPMU ARMCOP | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x0010 | CPMU HTCTL | R | 0 | 0 | VSEL | 0 | HTE | HTDS | HTIE | HTIF |
| | | W | | | | | | | | |
| 0x0011 | CPMU LVCTL | R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| | | W | | | | | | | | |
| 0x0012 | CPMU APICTL | R | APICLK | 0 | 0 | APIES | APIEA | APIFE | APIE | APIF |
| | | W | | | | | | | | |
| 0x0013 | CPMUACLKTR | R | ACLKTR5 | ACLKTR4 | ACLKTR3 | ACLKTR2 | ACLKTR1 | ACLKTR0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0014 | CPMUAPIRH | R | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| | | W | | | | | | | | |
| 0x0015 | CPMUAPIRL | R | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| | | W | | | | | | | | |
| 0x0016 | RESERVED CPMUTEST3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0017 | CPMUHTTR | R | HTOE | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| | | W | | | | | | | | |
| 0x0018 | CPMU IRCTRIMH | R | TCTRIM[4:0] | | | | | 0 | IRCTRIM[9:8] | |
| | | W | | | | | | | | |
| 0x0019 | CPMU IRCTRIML | R | IRCTRIM[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x001A | CPMUOSC | R | OSCE | 0 | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001B | CPMUPROT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PROT |
| | | W | | | | | | | | |
| 0x001C | RESERVED CPMUTEST2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001D | CPMU VREGCTL | R | 0 | 0 | 0 | 0 | 0 | 0 | EXTXON | INTXON |
| | | W | | | | | | | | |
| 0x001E | CPMUOSC2 | R | 0 | 0 | 0 | 0 | 0 | 0 | OMRE | OSCMOD |
| | | W | | | | | | | | |
| 0x001F | CPMU RESERVED1F | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 7-3. CPMU Register Summary**

## 7.3.2 Register Descriptions

This section describes all the S12CPMU_UHV_V5 registers and their individual bits.

Address order is as listed in Figure 7-3

### 7.3.2.1 S12CPMU_UHV_V5 Reset Flags Register (CPMURFLG)

This register provides S12CPMU_UHV_V5 reset flags.

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PORF | LVRF | 0 | COPRF | 0 | OMRF | PMRF |
| W | | | | | | | | |
| Reset | 0 | Note 1 | Note 2 | 0 | Note 3 | 0 | Note 4 | Note 5 |

1. PORF is set to 1 when a power on reset occurs. Unaffected by System Reset.
2. LVRF is set to 1 when a low voltage reset occurs. Unaffected by System Reset. Set by power on reset.
3. COPRF is set to 1 when COP reset occurs. Unaffected by System Reset. Cleared by power on reset.
4. OMRF is set to 1 when an oscillator clock monitor reset occurs. Unaffected by System Reset. Cleared by power on reset.
5. PMRF is set to 1 when a PLL clock monitor reset occurs. Unaffected by System Reset. Cleared by power on reset.

= Unimplemented or Reserved

**Figure 7-4. S12CPMU_UHV_V5 Flags Register (CPMURFLG)**

Read: Anytime

Write: Refer to each bit for individual write conditions

**Table 7-1. CPMURFLG Field Descriptions**

| Field | Description |
|---|---|
| 6<br>PORF | **Power on Reset Flag** — PORF is set to 1 when a power on reset occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Power on reset has not occurred.<br>1 Power on reset has occurred. |
| 5<br>LVRF | **Low Voltage Reset Flag** — LVRF is set to 1 when a low voltage reset occurs on the VDD, VDDF or VDDX domain. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Low voltage reset has not occurred.<br>1 Low voltage reset has occurred. |
| 3<br>COPRF | **COP Reset Flag** — COPRF is set to 1 when a COP (Computer Operating Properly) reset occurs. Refer to 7.5.5, “Computer Operating Properly Watchdog (COP) Reset and 7.3.2.10, “S12CPMU_UHV_V5 COP Control Register (CPMUCOP) for details.This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 COP reset has not occurred.<br>1 COP reset has occurred. |

**Table 7-1. CPMURFLG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>OMRF | **Oscillator Clock Monitor Reset Flag** — OMRF is set to 1 when a loss of oscillator (crystal) clock occurs. Refer to7.5.3, “Oscillator Clock Monitor Reset for details.This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Loss of oscillator clock reset has not occurred.<br>1 Loss of oscillator clock reset has occurred. |
| 0<br>PMRF | **PLL Clock Monitor Reset Flag** — PMRF is set to 1 when a loss of PLL clock occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect.<br>0 Loss of PLL clock reset has not occurred.<br>1 Loss of PLL clock reset has occurred. |

### 7.3.2.2 S12CPMU_UHV_V5 Synthesizer Register (CPMUSYNR)

The CPMUSYNR register controls the multiplication factor of the PLL and selects the VCO frequency range.

Module Base + 0x0004

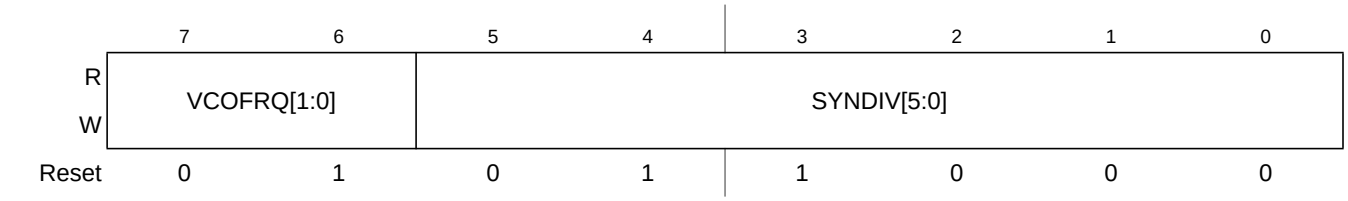



**Figure 7-5. S12CPMU_UHV_V5 Synthesizer Register (CPMUSYNR)**

Read: Anytime

Write: If PROT=0 (CPMUPROT register) and PLLSEL=1 (CPMUCLKS register), then write anytime. Else write has no effect.

**NOTE**

Writing to this register clears the LOCK and UPOSC status bits.

If PLL has locked (LOCK=1) $f_{VCO} = 2 \times f_{REF} \times (SYNDIV + 1)$

**NOTE**

$f_{VCO}$ must be within the specified VCO frequency lock range. Bus frequency $f_{bus}$ must not exceed the specified maximum.

The VCOFRQ[1:0] bits are used to configure the VCO gain for optimal stability and lock time. For correct PLL operation the VCOFRQ[1:0] bits have to be selected according to the actual target VCOCLK

frequency as shown in Table 7-2. Setting the VCOFRQ[1:0] bits incorrectly can result in a non functional PLL (no locking and/or insufficient stability).

**Table 7-2. VCO Clock Frequency Selection**

| VCOCLK Frequency Ranges | VCOFRQ[1:0] |
|---|---|
| 32MHz <= $f_{VCO}$ <= 48MHz | 00 |
| 48MHz < $f_{VCO}$ <= 64MHz | 01 |
| Reserved | 10 |
| Reserved | 11 |

### 7.3.2.3 S12CPMU_UHV_V5 Reference Divider Register (CPMUREFDIV)

The CPMUREFDIV register provides a finer granularity for the PLL multiplier steps when using the external oscillator as reference.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | REFFRQ[1:0] | | 0 | 0 | REFDIV[3:0] | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Figure 7-6. S12CPMU_UHV_V5 Reference Divider Register (CPMUREFDIV)**

Read: Anytime

Write: If PROT=0 (CPMUPROT register) and PLLSEL=1 (CPMUCLKS register), then write anytime. Else write has no effect.

**NOTE**

Write to this register clears the LOCK and UPOSC status bits.

If XOSCLCP is enabled (OSCE=1) $f_{REF} = \frac{f_{OSC}}{(REFDIV + 1)}$

If XOSCLCP is disabled (OSCE=0) $f_{REF} = f_{IRC1M}$

The REFFRQ[1:0] bits are used to configure the internal PLL filter for optimal stability and lock time. For correct PLL operation the REFFRQ[1:0] bits have to be selected according to the actual REFCLK frequency as shown in Table 7-3.

If IRC1M is selected as REFCLK (OSCE=0) the PLL filter is fixed configured for the 1MHz <= $f_{REF}$ <= 2MHz range. The bits can still be written but will have no effect on the PLL filter configuration.

For OSCE=1, setting the REFFRQ[1:0] bits incorrectly can result in a non functional PLL (no locking and/or insufficient stability).

**Table 7-3. Reference Clock Frequency Selection if OSC_LCP is enabled**

| REFCLK Frequency Ranges (OSCE=1) | REFFRQ[1:0] |
|---|---|
| 1MHz <= $f_{REF}$ <= 2MHz | 00 |
| 2MHz < $f_{REF}$ <= 6MHz | 01 |
| 6MHz < $f_{REF}$ <= 12MHz | 10 |
| $f_{REF}$ >12MHz | 11 |

### 7.3.2.4 S12CPMU_UHV_V5 Post Divider Register (CPMUPOSTDIV)

The POSTDIV register controls the frequency ratio between the VCOCLK and the PLLCLK.

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | POSTDIV[4:0] | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

= Unimplemented or Reserved

**Figure 7-7. S12CPMU_UHV_V5 Post Divider Register (CPMUPOSTDIV)**

Read: Anytime

Write: If PLLSEL=1 write anytime, else write has no effect

If PLL is locked (LOCK=1) $$f_{PLL} = \frac{f_{VCO}}{(POSTDIV + 1)}$$

If PLL is not locked (LOCK=0) $$f_{PLL} = \frac{f_{VCO}}{4}$$

If PLL is selected (PLLSEL=1) $$f_{bus} = \frac{f_{PLL}}{2}$$

When changing the POSTDIV[4:0] value or PLL transitions to locked stated (lock=1), it takes up to 32 Bus Clock cycles until $f_{PLL}$ is at the desired target frequency. This is because the post divider gradually changes (increases or decreases) $f_{PLL}$ in order to avoid sudden load changes for the on-chip voltage regulator.

### 7.3.2.5 S12CPMU_UHV_V5 Interrupt Flags Register (CPMUIFLG)

This register provides S12CPMU_UHV_V5 status bits and interrupt flags.

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIF | 0 | 0 | LOCKIF | LOCK | 0 | OSCIF | UPOSC |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-8. S12CPMU_UHV_V5 Flags Register (CPMUIFLG)**

Read: Anytime

Write: Refer to each bit for individual write conditions

**Table 7-4. CPMUIFLG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIF | **Real Time Interrupt Flag** — RTIF is set to 1 at the end of the RTI period. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (RTIE=1), RTIF causes an interrupt request.<br>0 RTI time-out has not yet occurred.<br>1 RTI time-out has occurred. |
| 4<br>LOCKIF | **PLL Lock Interrupt Flag** — LOCKIF is set to 1 when LOCK status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LOCKIE=1), LOCKIF causes an interrupt request.<br>0 No change in LOCK bit.<br>1 LOCK bit has changed. |
| 3<br>LOCK | **Lock Status Bit** — LOCK reflects the current state of PLL lock condition. Writes have no effect. While PLL is unlocked (LOCK=0) $f_{PLL}$ is $f_{VCO}$ / 4 to protect the system from high core clock frequencies during the PLL stabilization time $t_{lock}$.<br>0 VCOCLK is not within the desired tolerance of the target frequency.<br>$f_{PLL} = f_{VCO}/4$.<br>1 VCOCLK is within the desired tolerance of the target frequency.<br>$f_{PLL} = f_{VCO}/(POSTDIV+1)$. |
| 1<br>OSCIF | **Oscillator Interrupt Flag** — OSCIF is set to 1 when UPOSC status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (OSCIE=1), OSCIF causes an interrupt request.<br>0 No change in UPOSC bit.<br>1 UPOSC bit has changed. |
| 0<br>UPOSC | **Oscillator Status Bit** — UPOSC reflects the status of the oscillator. Writes have no effect. Entering Full Stop Mode UPOSC is cleared.<br>0 The oscillator is off or oscillation is not qualified by the PLL.<br>1 The oscillator is qualified by the PLL. |

### 7.3.2.6 S12CPMU_UHV_V5 Interrupt Enable Register (CPMUINT)

This register enables S12CPMU_UHV_V5 interrupt requests.

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | OSCIE | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-9. S12CPMU_UHV_V5 Interrupt Enable Register (CPMUINT)**

Read: Anytime

Write: Anytime

**Table 7-5. CPMUINT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIE | **Real Time Interrupt Enable Bit**<br>0 Interrupt requests from RTI are disabled.<br>1 Interrupt will be requested whenever RTIF is set. |
| 4<br>LOCKIE | **PLL Lock Interrupt Enable Bit**<br>0 PLL LOCK interrupt requests are disabled.<br>1 Interrupt will be requested whenever LOCKIF is set. |
| 1<br>OSCIE | **Oscillator Corrupt Interrupt Enable Bit**<br>0 Oscillator Corrupt interrupt requests are disabled.<br>1 Interrupt will be requested whenever OSCIF is set. |

### 7.3.2.7 S12CPMU_UHV_V5 Clock Select Register (CPMUCLKS)

This register controls S12CPMU_UHV_V5 clock selection.

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PLLSEL | PSTP | CSAD | COP OSCSEL1 | PRE | PCE | RTI OSCSEL | COP OSCSEL0 |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-10. S12CPMU_UHV_V5 Clock Select Register (CPMUCLKS)**

Read: Anytime

Write:

- Only possible if PROT=0 (CPMUPROT register) in all MCU Modes (Normal and Special Mode).
- All bits in Special Mode (if PROT=0).
- PLLSEL, PSTP, PRE, PCE, RTIOSCSEL: In Normal Mode (if PROT=0).
- CSAD: In Normal Mode (if PROT=0) until CPMUCOP write once has taken place.
- COPOSCSEL0: In Normal Mode (if PROT=0) until CPMUCOP write once has taken place. If COPOSCSEL0 was cleared by UPOSC=0 (entering Full Stop Mode with COPOSCSEL0=1 or insufficient OSCCLK quality), then COPOSCSEL0 can be set once again.
- COPOSCSEL1: In Normal Mode (if PROT=0) until CPMUCOP write once has taken place. COPOSCSEL1 will not be cleared by UPOSC=0 (entering Full Stop Mode with COPOSCSEL1=1 or insufficient OSCCLK quality if OSCCLK is used as clock source for other clock domains: for instance core clock etc.).

**NOTE**

After writing CPMUCLKS register, it is strongly recommended to read back CPMUCLKS register to make sure that write of PLLSEL, RTIOSCSEL and COPOSCSEL was successful. This is because under certain circumstances writes have no effect or bits are automatically changed (see CPMUCLKS register and bit descriptions).

**NOTE**

When using the oscillator clock as system clock (write PLLSEL = 0) it is highly recommended to enable the oscillator clock monitor reset feature (write OMRE = 1 in CPMUOSC2 register). If the oscillator monitor reset feature is disabled (OMRE = 0) and the oscillator clock is used as system clock, the system will stall in case of loss of oscillation.

**Table 7-6. CPMUCLKS Descriptions**

| Field | Description |
|---|---|
| 7<br>PLLSEL | **PLL Select Bit**<br>This bit selects the PLLCLK as source of the System Clocks (Core Clock and Bus Clock).<br>PLLSEL can only be set to 0, if UPOSC=1.<br>UPOSC= 0 sets the PLLSEL bit.<br>Entering Full Stop Mode sets the PLLSEL bit.<br>0 System clocks are derived from OSCCLK if oscillator is up (UPOSC=1, $f_{bus} = f_{osc} / 2$).<br>1 System clocks are derived from PLLCLK, $f_{bus} = f_{PLL} / 2$. |
| 6<br>PSTP | **Pseudo Stop Bit**<br>This bit controls the functionality of the oscillator during Stop Mode.<br>0 Oscillator is disabled in Stop Mode (Full Stop Mode).<br>1 Oscillator continues to run in Stop Mode (Pseudo Stop Mode), option to run RTI and COP.<br>**Note:** Pseudo Stop Mode allows for faster STOP recovery and reduces the mechanical stress and aging of the resonator in case of frequent STOP conditions at the expense of a slightly increased power consumption.<br>**Note:** When starting up the external oscillator (either by programming OSCE bit to 1 or on exit from Full Stop Mode with OSCE bit already 1) the software must wait for a minimum time equivalent to the startup-time of the external oscillator $t_{UPOSC}$ before entering Pseudo Stop Mode. |
| 5<br>CSAD | **COP in Stop Mode ACLK Disable** — This bit disables the ACLK for the COP in Stop Mode. Hence the COP is static while in Stop Mode and continues to operate after exit from Stop Mode.<br>Due to clock domain crossing synchronization there is a latency time to enter and exit Stop Mode if COP clock source is ACLK and this clock is stopped in Stop Mode. This maximum latency time is 4 ACLK cycles which must be added to the Stop Mode recovery time tSTP_REC from exit of current Stop Mode to entry of next Stop Mode. This latency time occurs no matter which Stop Mode (Full, Pseudo) is currently exited or entered next. After exit from Stop Mode (Pseudo, Full) for 2 ACLK cycles no Stop Mode request (STOP instruction) should be generated to make sure the COP counter increments at each Stop Mode exit.<br>This bit does not influence the ACLK for the API.<br>0 COP running in Stop Mode (ACLK for COP enabled in Stop Mode).<br>1 COP stopped in Stop Mode (ACLK for COP disabled in Stop Mode) |
| 4<br>COP<br>OSCSEL1 | **COP Clock Select 1** — COPOSCSEL0 and COPOSCSEL1 combined determine the clock source to the COP (see also Table 7-7).<br>If COPOSCSEL1 = 1, COPOSCSEL0 has no effect regarding clock select and changing the COPOSCSEL0 bit does not re-start the COP time-out period.<br>COPOSCSEL1 selects the clock source to the COP to be either ACLK (derived from trimmable internal RC-Oscillator) or clock selected via COPOSCSEL0 (IRCCLK or OSCCLK).<br>Changing the COPOSCSEL1 bit re-starts the COP time-out period.<br>COPOSCSEL1 can be set independent from value of UPOSC.<br>UPOSC= 0 does not clear the COPOSCSEL1 bit.<br>0 COP clock source defined by COPOSCSEL0<br>1 COP clock source is ACLK derived from a trimmable internal RC-Oscillator |
| 3<br>PRE | **RTI Enable During Pseudo Stop Bit** — PRE enables the RTI during Pseudo Stop Mode.<br>0 RTI stops running during Pseudo Stop Mode.<br>1 RTI continues running during Pseudo Stop Mode if RTIOSCSEL=1.<br>**Note:** If PRE=0 or RTIOSCSEL=0 then the RTI will go static while Stop Mode is active. The RTI counter will not be reset. |
| 2<br>PCE | **COP Enable During Pseudo Stop Bit** — PCE enables the COP during Pseudo Stop Mode.<br>0 COP stops running during Pseudo Stop Mode<br>1 COP continues running during Pseudo Stop Mode if COPOSCSEL=1<br>**Note:** If PCE=0 or COPOSCSEL=0 then the COP will go static while Stop Mode is active. The COP counter will not be reset. |

**Table 7-6. CPMUCLKS Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>RTIOSCSEL | **RTI Clock Select**— RTIOSCSEL selects the clock source to the RTI. Either IRCCLK or OSCCLK. Changing the RTIOSCSEL bit re-starts the RTI time-out period.<br>RTIOSCSEL can only be set to 1, if UPOSC=1.<br>UPOSC= 0 clears the RTIOSCSEL bit.<br>0 RTI clock source is IRCCLK.<br>1 RTI clock source is OSCCLK. |
| 0<br>COP<br>OSCSEL0 | **COP Clock Select 0** — COPOSCSEL0 and COPOSCSEL1 combined determine the clock source to the COP (see also Table 7-7)<br>If COPOSCSEL1 = 1, COPOSCSEL0 has no effect regarding clock select and changing the COPOSCSEL0 bit does not re-start the COP time-out period.<br>When COPOSCSEL1=0,COPOSCSEL0 selects the clock source to the COP to be either IRCCLK or OSCCLK. Changing the COPOSCSEL0 bit re-starts the COP time-out period.<br>COPOSCSEL0 can only be set to 1, if UPOSC=1.<br>UPOSC= 0 clears the COPOSCSEL0 bit.<br>0 COP clock source is IRCCLK.<br>1 COP clock source is OSCCLK |

**Table 7-7. COPOSCSEL1, COPOSCSEL0 clock source select description**

| COPOSCSEL1 | COPOSCSEL0 | COP clock source |
|---|---|---|
| 0 | 0 | IRCCLK |
| 0 | 1 | OSCCLK |
| 1 | x | ACLK |

### 7.3.2.8 S12CPMU_UHV_V5 PLL Control Register (CPMUPLL)

This register controls the PLL functionality.

Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | FM1 | FM0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-11. S12CPMU_UHV_V5 PLL Control Register (CPMUPLL)**

Read: Anytime

Write: Anytime if PROT=0 (CPMUPROT register) and PLLSEL=1 (CPMUCLKS register). Else write has no effect.

**NOTE**

Write to this register clears the LOCK and UPOSC status bits.

**NOTE**

Care should be taken to ensure that the bus frequency does not exceed the specified maximum when frequency modulation is enabled.

**Table 7-8. CPMUPLL Field Descriptions**

| Field | Description |
|---|---|
| 5, 4<br>FM1, FM0 | PLL **Frequency Modulation Enable Bits** — FM1 and FM0 enable frequency modulation on the VCOCLK. This is to reduce noise emission. The modulation frequency is $f_{ref}$ divided by 16. See Table 7-9 for coding. |

**Table 7-9. FM Amplitude selection**

| FM1 | FM0 | FM Amplitude / $f_{VCO}$ Variation |
|---|---|---|
| 0 | 0 | FM off |
| 0 | 1 | ±1% |
| 1 | 0 | ±2% |
| 1 | 1 | ±4% |

### 7.3.2.9 S12CPMU_UHV_V5 RTI Control Register (CPMURTI)

This register selects the time-out period for the Real Time Interrupt.

The clock source for the RTI is either IRCCLK or OSCCLK depending on the setting of the RTIOSCSEL bit. In Stop Mode with PSTP=1 (Pseudo Stop Mode) and RTIOSCSEL=1 the RTI continues to run, else the RTI counter halts in Stop Mode.

Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RTDEC | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-12. S12CPMU_UHV_V5 RTI Control Register (CPMURTI)**

Read: Anytime

Write: Anytime

**NOTE**

A write to this register starts the RTI time-out period. A change of the RTIOSCSEL bit (writing a different value or loosing UPOSC status) re-starts the RTI time-out period.

**Table 7-10. CPMURTI Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTDEC | **Decimal or Binary Divider Select Bit** — RTDEC selects decimal or binary based prescaler values.<br>0 Binary based divider value. See Table 7-11<br>1 Decimal based divider value. See Table 7-12 |
| 6–4<br>RTR[6:4] | **Real Time Interrupt Prescale Rate Select Bits** — These bits select the prescale rate for the RTI.See Table 7-11 and Table 7-12. |
| 3–0<br>RTR[3:0] | **Real Time Interrupt Modulus Counter Select Bits** — These bits select the modulus counter target value to provide additional granularity.Table 7-11 and Table 7-12 show all possible divide values selectable by the CPMURTI register. |

**Table 7-11. RTI Frequency Divide Rates for RTDEC = 0**

| RTR[3:0] | RTR[6:4] = | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **000 (OFF)** | **001 ($2^{10}$)** | **010 ($2^{11}$)** | **011 ($2^{12}$)** | **100 ($2^{13}$)** | **101 ($2^{14}$)** | **110 ($2^{15}$)** | **111 ($2^{16}$)** |
| **0000 (÷1)** | OFF[1] | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{13}$ | $2^{14}$ | $2^{15}$ | $2^{16}$ |
| **0001 (÷2)** | OFF | $2x2^{10}$ | $2x2^{11}$ | $2x2^{12}$ | $2x2^{13}$ | $2x2^{14}$ | $2x2^{15}$ | $2x2^{16}$ |
| **0010 (÷3)** | OFF | $3x2^{10}$ | $3x2^{11}$ | $3x2^{12}$ | $3x2^{13}$ | $3x2^{14}$ | $3x2^{15}$ | $3x2^{16}$ |
| **0011 (÷4)** | OFF | $4x2^{10}$ | $4x2^{11}$ | $4x2^{12}$ | $4x2^{13}$ | $4x2^{14}$ | $4x2^{15}$ | $4x2^{16}$ |
| **0100 (÷5)** | OFF | $5x2^{10}$ | $5x2^{11}$ | $5x2^{12}$ | $5x2^{13}$ | $5x2^{14}$ | $5x2^{15}$ | $5x2^{16}$ |
| **0101 (÷6)** | OFF | $6x2^{10}$ | $6x2^{11}$ | $6x2^{12}$ | $6x2^{13}$ | $6x2^{14}$ | $6x2^{15}$ | $6x2^{16}$ |
| **0110 (÷7)** | OFF | $7x2^{10}$ | $7x2^{11}$ | $7x2^{12}$ | $7x2^{13}$ | $7x2^{14}$ | $7x2^{15}$ | $7x2^{16}$ |
| **0111 (÷8)** | OFF | $8x2^{10}$ | $8x2^{11}$ | $8x2^{12}$ | $8x2^{13}$ | $8x2^{14}$ | $8x2^{15}$ | $8x2^{16}$ |
| **1000 (÷9)** | OFF | $9x2^{10}$ | $9x2^{11}$ | $9x2^{12}$ | $9x2^{13}$ | $9x2^{14}$ | $9x2^{15}$ | $9x2^{16}$ |
| **1001 (÷10)** | OFF | $10x2^{10}$ | $10x2^{11}$ | $10x2^{12}$ | $10x2^{13}$ | $10x2^{14}$ | $10x2^{15}$ | $10x2^{16}$ |
| **1010 (÷11)** | OFF | $11x2^{10}$ | $11x2^{11}$ | $11x2^{12}$ | $11x2^{13}$ | $11x2^{14}$ | $11x2^{15}$ | $11x2^{16}$ |
| **1011 (÷12)** | OFF | $12x2^{10}$ | $12x2^{11}$ | $12x2^{12}$ | $12x2^{13}$ | $12x2^{14}$ | $12x2^{15}$ | $12x2^{16}$ |
| **1100 (÷13)** | OFF | $13x2^{10}$ | $13x2^{11}$ | $13x2^{12}$ | $13x2^{13}$ | $13x2^{14}$ | $13x2^{15}$ | $13x2^{16}$ |
| **1101 (÷14)** | OFF | $14x2^{10}$ | $14x2^{11}$ | $14x2^{12}$ | $14x2^{13}$ | $14x2^{14}$ | $14x2^{15}$ | $14x2^{16}$ |
| **1110 (÷15)** | OFF | $15x2^{10}$ | $15x2^{11}$ | $15x2^{12}$ | $15x2^{13}$ | $15x2^{14}$ | $15x2^{15}$ | $15x2^{16}$ |
| **1111 (÷16)** | OFF | $16x2^{10}$ | $16x2^{11}$ | $16x2^{12}$ | $16x2^{13}$ | $16x2^{14}$ | $16x2^{15}$ | $16x2^{16}$ |

[1] Denotes the default value out of reset.This value should be used to disable the RTI to ensure future backwards compatibility.

**Table 7-12. RTI Frequency Divide Rates for RTDEC=1**

| RTR[3:0] | RTR[6:4] = | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **000 ($1x10^3$)** | **001 ($2x10^3$)** | **010 ($5x10^3$)** | **011 ($10x10^3$)** | **100 ($20x10^3$)** | **101 ($50x10^3$)** | **110 ($100x10^3$)** | **111 ($200x10^3$)** |
| **0000 (÷1)** | $1x10^3$ | $2x10^3$ | $5x10^3$ | $10x10^3$ | $20x10^3$ | $50x10^3$ | $100x10^3$ | $200x10^3$ |
| **0001 (÷2)** | $2x10^3$ | $4x10^3$ | $10x10^3$ | $20x10^3$ | $40x10^3$ | $100x10^3$ | $200x10^3$ | $400x10^3$ |
| **0010 (÷3)** | $3x10^3$ | $6x10^3$ | $15x10^3$ | $30x10^3$ | $60x10^3$ | $150x10^3$ | $300x10^3$ | $600x10^3$ |
| **0011 (÷4)** | $4x10^3$ | $8x10^3$ | $20x10^3$ | $40x10^3$ | $80x10^3$ | $200x10^3$ | $400x10^3$ | $800x10^3$ |
| **0100 (÷5)** | $5x10^3$ | $10x10^3$ | $25x10^3$ | $50x10^3$ | $100x10^3$ | $250x10^3$ | $500x10^3$ | $1x10^6$ |
| **0101 (÷6)** | $6x10^3$ | $12x10^3$ | $30x10^3$ | $60x10^3$ | $120x10^3$ | $300x10^3$ | $600x10^3$ | $1.2x10^6$ |
| **0110 (÷7)** | $7x10^3$ | $14x10^3$ | $35x10^3$ | $70x10^3$ | $140x10^3$ | $350x10^3$ | $700x10^3$ | $1.4x10^6$ |
| **0111 (÷8)** | $8x10^3$ | $16x10^3$ | $40x10^3$ | $80x10^3$ | $160x10^3$ | $400x10^3$ | $800x10^3$ | $1.6x10^6$ |
| **1000 (÷9)** | $9x10^3$ | $18x10^3$ | $45x10^3$ | $90x10^3$ | $180x10^3$ | $450x10^3$ | $900x10^3$ | $1.8x10^6$ |
| **1001 (÷10)** | $10\ x10^3$ | $20x10^3$ | $50x10^3$ | $100x10^3$ | $200x10^3$ | $500x10^3$ | $1x10^6$ | $2x10^6$ |
| **1010 (÷11)** | $11\ x10^3$ | $22x10^3$ | $55x10^3$ | $110x10^3$ | $220x10^3$ | $550x10^3$ | $1.1x10^6$ | $2.2x10^6$ |
| **1011 (÷12)** | $12x10^3$ | $24x10^3$ | $60x10^3$ | $120x10^3$ | $240x10^3$ | $600x10^3$ | $1.2x10^6$ | $2.4x10^6$ |
| **1100 (÷13)** | $13x10^3$ | $26x10^3$ | $65x10^3$ | $130x10^3$ | $260x10^3$ | $650x10^3$ | $1.3x10^6$ | $2.6x10^6$ |
| **1101 (÷14)** | $14x10^3$ | $28x10^3$ | $70x10^3$ | $140x10^3$ | $280x10^3$ | $700x10^3$ | $1.4x10^6$ | $2.8x10^6$ |
| **1110 (÷15)** | $15x10^3$ | $30x10^3$ | $75x10^3$ | $150x10^3$ | $300x10^3$ | $750x10^3$ | $1.5x10^6$ | $3x10^6$ |
| **1111 (÷16)** | $16x10^3$ | $32x10^3$ | $80x10^3$ | $160x10^3$ | $320x10^3$ | $800x10^3$ | $1.6x10^6$ | $3.2x10^6$ |

### 7.3.2.10 S12CPMU_UHV_V5 COP Control Register (CPMUCOP)

This register controls the COP (Computer Operating Properly) watchdog.

The clock source for the COP is either ACLK, IRCCLK or OSCCLK depending on the setting of the COPOSCSEL0 and COPOSCSEL1 bit (see also Table 7-7).
In Stop Mode with PSTP=1 (Pseudo Stop Mode), COPOSCSEL0=1 and COPOSCEL1=0 and PCE=1 the COP continues to run, else the COP counter halts in Stop Mode with COPOSCSEL1 =0.
In Full Stop Mode and Pseudo Stop Mode with COPOSCSEL1=1 the COP continues to run.

Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 |
| W | | | WRTMASK | | | | | |
| Reset | F | 0 | 0 | 0 | 0 | F | F | F |

After de-assert of System Reset the values are automatically loaded from the Flash memory. See Device specification for details.

= Unimplemented or Reserved

**Figure 7-13. S12CPMU_UHV_V5 COP Control Register (CPMUCOP)**

Read: Anytime

Write:

1. RSBCK: Anytime in Special Mode; write to “1” but not to “0” in Normal Mode
2. WCOP, CR2, CR1, CR0:
   - Anytime in Special Mode, when WRTMASK is 0, otherwise it has no effect
   - Write once in Normal Mode, when WRTMASK is 0, otherwise it has no effect.
     - Writing CR[2:0] to “000” has no effect, but counts for the “write once” condition.
     - Writing WCOP to “0” has no effect, but counts for the “write once” condition.

When a non-zero value is loaded from Flash to CR[2:0] the COP time-out period is started.

A change of the COPOSCSEL0 or COPOSCSEL1 bit (writing a different value) or loosing UPOSC status while COPOSCSEL1 is clear and COPOSCSEL0 is set, re-starts the COP time-out period.

In Normal Mode the COP time-out period is restarted if either of these conditions is true:

1. Writing a non-zero value to CR[2:0] (anytime in special mode, once in normal mode) with WRTMASK = 0.
2. Writing WCOP bit (anytime in Special Mode, once in Normal Mode) with WRTMASK = 0.
3. Changing RSBCK bit from “0” to “1”.

In Special Mode, any write access to CPMUCOP register restarts the COP time-out period.

**Table 7-13. CPMUCOP Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WCOP | **Window COP Mode Bit** — When set, a write to the CPMUARMCOP register must occur in the last 25% of the selected period. A write during the first 75% of the selected period generates a COP reset. As long as all writes occur during this window, $55 can be written as often as desired. Once $AA is written after the $55, the time-out logic restarts and the user must wait until the next window before writing to CPMUARMCOP. Table 7-14 shows the duration of this window for the seven available COP rates.<br>0 Normal COP operation<br>1 Window COP operation |
| 6<br>RSBCK | **COP and RTI Stop in Active BDM Mode Bit**<br>0 Allows the COP and RTI to keep running in Active BDM mode.<br>1 Stops the COP and RTI counters whenever the part is in Active BDM mode. |
| 5<br>WRTMASK | **Write Mask for WCOP and CR[2:0] Bit** — This write-only bit serves as a mask for the WCOP and CR[2:0] bits while writing the CPMUCOP register. It is intended for BDM writing the RSBCK without changing the content of WCOP and CR[2:0].<br>0 Write of WCOP and CR[2:0] has an effect with this write of CPMUCOP<br>1 Write of WCOP and CR[2:0] has no effect with this write of CPMUCOP.<br>(Does not count for “write once”.) |
| 2–0<br>CR[2:0] | **COP Watchdog Timer Rate Select** — These bits select the COP time-out rate (see Table 7-14 and Table 7-15). Writing a nonzero value to CR[2:0] enables the COP counter and starts the time-out period. A COP counter time-out causes a System Reset. This can be avoided by periodically (before time-out) initializing the COP counter via the CPMUARMCOP register.<br>While all of the following four conditions are true the CR[2:0], WCOP bits are ignored and the COP operates at highest time-out period ($2^{24}$ cycles) in normal COP mode (Window COP mode disabled):<br>1) COP is enabled (CR[2:0] is not 000)<br>2) BDM mode active<br>3) RSBCK = 0<br>4) Operation in Special Mode |

**Table 7-14. COP Watchdog Rates if COPOSCSEL1=0. (default out of reset)**

| CR2 | CR1 | CR0 | COPCLK Cycles to time-out (COPCLK is either IRCCLK or OSCCLK depending on the COPOSCSEL0 bit) |
|---|---|---|---|
| 0 | 0 | 0 | COP disabled |
| 0 | 0 | 1 | $2^{14}$ |
| 0 | 1 | 0 | $2^{16}$ |
| 0 | 1 | 1 | $2^{18}$ |
| 1 | 0 | 0 | $2^{20}$ |
| 1 | 0 | 1 | $2^{22}$ |
| 1 | 1 | 0 | $2^{23}$ |
| 1 | 1 | 1 | $2^{24}$ |

**Table 7-15. COP Watchdog Rates if COPOSCSEL1=1.**

| CR2 | CR1 | CR0 | COPCLK Cycles to time-out (COPCLK is ACLK divided by 2) |
|---|---|---|---|
| 0 | 0 | 0 | COP disabled |
| 0 | 0 | 1 | $2^{7}$ |
| 0 | 1 | 0 | $2^{9}$ |
| 0 | 1 | 1 | $2^{11}$ |
| 1 | 0 | 0 | $2^{13}$ |
| 1 | 0 | 1 | $2^{15}$ |
| 1 | 1 | 0 | $2^{16}$ |
| 1 | 1 | 1 | $2^{17}$ |

### 7.3.2.11 Reserved Register CPMUTEST0

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in Special Mode can alter the S12CPMU_UHV_V5's functionality.

Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-14. Reserved Register (CPMUTEST0)**

Read: Anytime

Write: Only in Special Mode

### 7.3.2.12 Reserved Register CPMUTEST1

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in Special Mode can alter the S12CPMU_UHV_V5's functionality.

Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-15. Reserved Register (CPMUTEST1)**

Read: Anytime

Write: Only in Special Mode

### 7.3.2.13 S12CPMU_UHV_V5 COP Timer Arm/Reset Register (CPMUARMCOP)

This register is used to restart the COP time-out period.

Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | ARMCOP-Bit 7 | ARMCOP-Bit 6 | ARMCOP-Bit 5 | ARMCOP-Bit 4 | ARMCOP-Bit 3 | ARMCOP-Bit 2 | ARMCOP-Bit 1 | ARMCOP-Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-16. S12CPMU_UHV_V5 CPMUARMCOP Register**

Read: Always reads $00

Write: Anytime

When the COP is disabled (CR[2:0] = “000”) writing to this register has no effect.

When the COP is enabled by setting CR[2:0] nonzero, the following applies:

Writing any value other than $55 or $AA causes a COP reset. To restart the COP time-out period write $55 followed by a write of $AA. These writes do not need to occur back-to-back, but the sequence ($55, $AA) must be completed prior to COP end of time-out period to avoid a COP reset. Sequences of $55 writes are allowed. When the WCOP bit is set, $55 and $AA writes must be done in the last 25% of the selected time-out period; writing any value in the first 75% of the selected period will cause a COP reset.

### 7.3.2.14 High Temperature Control Register (CPMUHTCTL)

The CPMUHTCTL register configures the temperature sense features.

Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | VSEL | 0 | HTE | HTDS | HTIE | HTIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-17. High Temperature Control Register (CPMUHTCTL)**

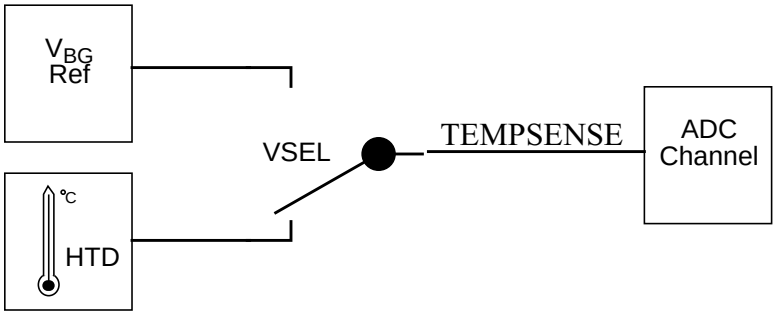
Read: Anytime

Write: VSEL, HTE, HTIE and HTIF are write anytime, HTDS is read only

**Table 7-16. CPMUHTCTL Field Descriptions**

| Field | Description |
|---|---|
| 5<br>VSEL | **Voltage Access Select Bit** — If set, the bandgap reference voltage $V_{BG}$ can be accessed internally (i.e. multiplexed to an internal Analog to Digital Converter channel). If not set, the die temperature proportional voltage $V_{HT}$ of the temperature sensor can be accessed internally. See device level specification for connectivity. For any of these access the HTE bit must be set.<br>0 An internal temperature proportional voltage $V_{HT}$ can be accessed internally.<br>1 Bandgap reference voltage $V_{BG}$ can be accessed internally. |
| 3<br>HTE | **High Temperature Sensor/Bandgap Voltage Enable Bit** — This bit enables the high temperature sensor and bandgap voltage amplifier.<br>0 The temperature sensor and bandgap voltage amplifier is disabled.<br>1 The temperature sensor and bandgap voltage amplifier is enabled. |
| 2<br>HTDS | **High Temperature Detect Status Bit** — This read-only status bit reflects the temperature status. Writes have no effect.<br>0 Junction Temperature is below level $T_{HTID}$ or RPM.<br>1 Junction Temperature is above level $T_{HTIA}$ and FPM. |
| 1<br>HTIE | **High Temperature Interrupt Enable Bit**<br>0 Interrupt request is disabled.<br>1 Interrupt will be requested whenever HTIF is set. |
| 0<br>HTIF | **High Temperature Interrupt Flag** — HTIF is set to 1 when HTDS status bit changes. This flag can only be cleared by writing a 1.<br>Writing a 0 has no effect. If enabled (HTIE=1), HTIF causes an interrupt request.<br>0 No change in HTDS bit.<br>1 HTDS bit has changed. |

**Figure 7-18. Voltage Access Select**



### 7.3.2.15 Low Voltage Control Register (CPMULVCTL)

The CPMULVCTL register allows the configuration of the low-voltage detect features.

Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | U | 0 | U |

The Reset state of LVDS and LVIF depends on the external supplied VDDA level

= Unimplemented or Reserved

**Figure 7-19. Low Voltage Control Register (CPMULVCTL)**

Read: Anytime

Write: LVIE and LVIF are write anytime, LVDS is read only

**Table 7-17. CPMULVCTL Field Descriptions**

| Field | Description |
|---|---|
| 2<br>LVDS | **Low-Voltage Detect Status Bit** — This read-only status bit reflects the voltage level on VDDA. Writes have no effect.<br>0 Input voltage VDDA is above level $V_{LVID}$ or RPM.<br>1 Input voltage VDDA is below level $V_{LVIA}$ and FPM. |
| 1<br>LVIE | **Low-Voltage Interrupt Enable Bit**<br>0 Interrupt request is disabled.<br>1 Interrupt will be requested whenever LVIF is set. |
| 0<br>LVIF | **Low-Voltage Interrupt Flag** — LVIF is set to 1 when LVDS status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LVIE = 1), LVIF causes an interrupt request.<br>0 No change in LVDS bit.<br>1 LVDS bit has changed. |

### 7.3.2.16 Autonomous Periodical Interrupt Control Register (CPMUAPICTL)

The CPMUAPICTL register allows the configuration of the autonomous periodical interrupt features.

Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | APICLK | 0 | 0 | APIES | APIEA | APIFE | APIE | APIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-20. Autonomous Periodical Interrupt Control Register (CPMUAPICTL)**

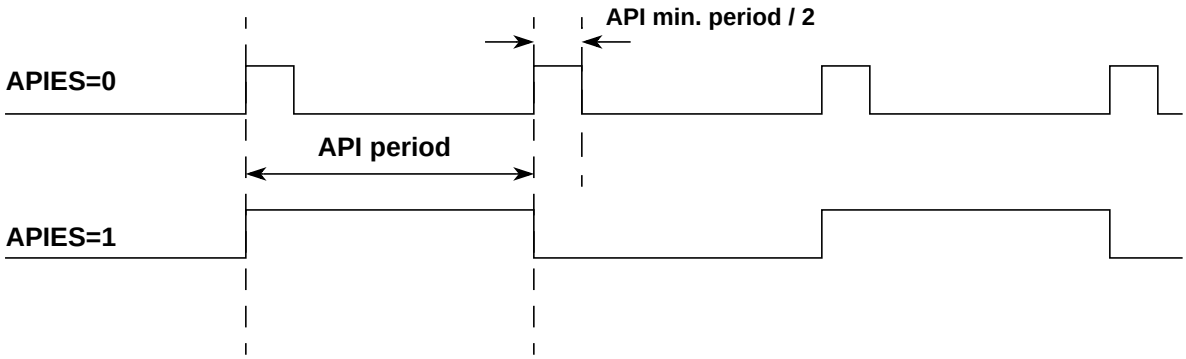
Read: Anytime

Write: Anytime

**Table 7-18. CPMUAPICTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>APICLK | **Autonomous Periodical Interrupt Clock Select Bit** — Selects the clock source for the API. Writable only if APIFE = 0. APICLK cannot be changed if APIFE is set by the same write operation.<br>0 Autonomous Clock (ACLK) used as source.<br>1 Bus Clock used as source. |
| 4<br>APIES | **Autonomous Periodical Interrupt External Select Bit** — Selects the waveform at the external pin API_EXTCLK as shown in Figure 7-21. See device level specification for connectivity of API_EXTCLK pin.<br>0 If APIEA and APIFE are set, at the external pin API_EXTCLK periodic high pulses are visible at the end of every selected period with the size of half of the minimum period (APIR=0x0000 in Table 7-22).<br>1 If APIEA and APIFE are set, at the external pin API_EXTCLK a clock is visible with 2 times the selected API Period. |
| 3<br>APIEA | **Autonomous Periodical Interrupt External Access Enable Bit** — If set, the waveform selected by bit APIES can be accessed externally. See device level specification for connectivity.<br>0 Waveform selected by APIES can not be accessed externally.<br>1 Waveform selected by APIES can be accessed externally, if APIFE is set. |
| 2<br>APIFE | **Autonomous Periodical Interrupt Feature Enable Bit** — Enables the API feature and starts the API timer when set.<br>0 Autonomous periodical interrupt is disabled.<br>1 Autonomous periodical interrupt is enabled and timer starts running. |
| 1<br>APIE | **Autonomous Periodical Interrupt Enable Bit**<br>0 API interrupt request is disabled.<br>1 API interrupt will be requested whenever APIF is set. |
| 0<br>APIF | **Autonomous Periodical Interrupt Flag** — APIF is set to 1 when the in the API configured time has elapsed. This flag can only be cleared by writing a 1.Writing a 0 has no effect. If enabled (APIE = 1), APIF causes an interrupt request.<br>0 API time-out has not yet occurred.<br>1 API time-out has occurred. |

**Figure 7-21. Waveform selected on API_EXTCLK pin (APIEA=1, APIFE=1)**



### 7.3.2.17 Autonomous Clock Trimming Register (CPMUACLKTR)

The CPMUACLKTR register configures the trimming of the Autonomous Clock (ACLK - trimmable internal RC-Oscillator) which can be selected as clock source for some CPMU features.

Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ACLKTR5 | ACLKTR4 | ACLKTR3 | ACLKTR2 | ACLKTR1 | ACLKTR0 | 0 | 0 |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | 0 | 0 |

After de-assert of System Reset a value is automatically loaded from the Flash memory.

**Figure 7-22. Autonomous Clock Trimming Register (CPMUACLKTR)**

Read: Anytime

Write: Anytime

**Table 7-19. CPMUACLKTR Field Descriptions**

| Field | Description |
|---|---|
| 7–2<br>ACLKTR[5:0] | **Autonomous Clock Period Trimming Bits** — See Table 7-20 for trimming effects. The ACLKTR[5:0] value represents a signed number influencing the ACLK period time. |

**Table 7-20. Trimming Effect of ACLKTR[5:0]**

| Bit | Trimming Effect |
|---|---|
| ACLKTR[5] | Increases period |
| ACLKTR[4] | Decreases period less than ACLKTR[5] increased it |
| ACLKTR[3] | Decreases period less than ACLKTR[4] |
| ACLKTR[2] | Decreases period less than ACLKTR[3] |
| ACLKTR[1] | Decreases period less than ACLKTR[2] |
| ACLKTR[0] | Decreases period less than ACLKTR[1] |

### 7.3.2.18 Autonomous Periodical Interrupt Rate High and Low Register (CPMUAPIRH / CPMUAPIRL)

The CPMUAPIRH and CPMUAPIRL registers allow the configuration of the autonomous periodical interrupt rate.

Module Base + 0x0014

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-23. Autonomous Periodical Interrupt Rate High Register (CPMUAPIRH)**

Module Base + 0x0015

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-24. Autonomous Periodical Interrupt Rate Low Register (CPMUAPIRL)**

Read: Anytime

Write: Anytime if APIFE=0, Else writes have no effect.

**Table 7-21. CPMUAPIRH / CPMUAPIRL Field Descriptions**

| Field | Description |
|---|---|
| 15-0<br>APIR[15:0] | **Autonomous Periodical Interrupt Rate Bits** — These bits define the time-out period of the API. See Table 7-22 for details of the effect of the autonomous periodical interrupt rate bits. |

The period can be calculated as follows depending on logical value of the APICLK bit:

APICLK=0: Period = 2*(APIR[15:0] + 1) * (ACLK Clock Period * 2)
APICLK=1: Period = 2*(APIR[15:0] + 1) * Bus Clock Period

**NOTE**

For APICLK bit clear the first time-out period of the API will show a latency time between two to three $f_{ACLK}$ cycles due to synchronous clock gate release when the API feature gets enabled (APIFE bit set).

**Table 7-22. Selectable Autonomous Periodical Interrupt Periods**

| APICLK | APIR[15:0] | Selected Period |
|---|---|---|
| 0 | 0000 | 0.2 ms[1] |
| 0 | 0001 | 0.4 ms[1] |
| 0 | 0002 | 0.6 ms[1] |
| 0 | 0003 | 0.8 ms[1] |
| 0 | 0004 | 1.0 ms[1] |
| 0 | 0005 | 1.2 ms[1] |
| 0 | ..... | ..... |
| 0 | FFFD | 13106.8 ms[1] |
| 0 | FFFE | 13107.0 ms[1] |
| 0 | FFFF | 13107.2 ms[1] |
| 1 | 0000 | 2 * Bus Clock period |
| 1 | 0001 | 4 * Bus Clock period |
| 1 | 0002 | 6 * Bus Clock period |
| 1 | 0003 | 8 * Bus Clock period |
| 1 | 0004 | 10 * Bus Clock period |
| 1 | 0005 | 12 * Bus Clock period |
| 1 | ..... | ..... |
| 1 | FFFD | 131068 * Bus Clock period |
| 1 | FFFE | 131070 * Bus Clock period |
| 1 | FFFF | 131072 * Bus Clock period |

1 When $f_{ACLK}$ is trimmed to 20KHz.

### 7.3.2.19 Reserved Register CPMUTEST3

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in Special Mode can alter the S12CPMU_UHV_V5's functionality.

Module Base + 0x0016

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-25. Reserved Register (CPMUTEST3)**

Read: Anytime

Write: Only in Special Mode

### 7.3.2.20 High Temperature Trimming Register (CPMUHTTR)

The CPMUHTTR register configures the trimming of the S12CPMU_UHV_V5 temperature sense.

Module Base + 0x0017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | HTOE | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | F | F | F | F |

After de-assert of System Reset a trim value is automatically loaded from the Flash memory. See Device specification for details.

= Unimplemented or Reserved

**Figure 7-26. High Temperature Trimming Register (CPMUHTTR)**

Read: Anytime

Write: Anytime

**Table 7-24. CPMUHTTR Field Descriptions**

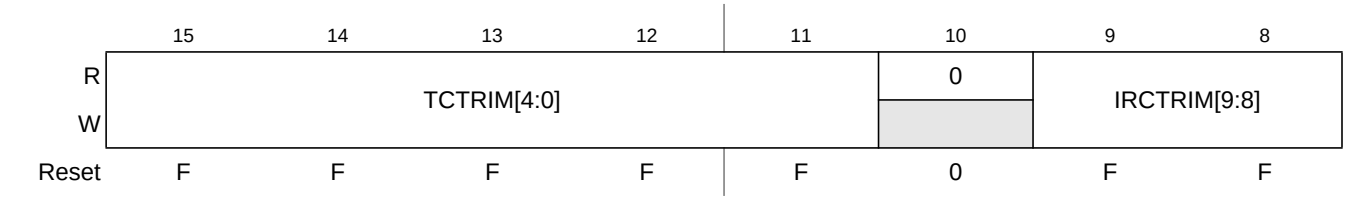
| Field | Description |
|---|---|
| 7<br>HTOE | **High Temperature Offset Enable Bit** — If set the temperature sense offset is enabled.<br>0 The temperature sense offset is disabled. HTTR[3:0] bits don't care.<br>1 The temperature sense offset is enabled. HTTR[3:0] select the temperature offset. |
| 3–0<br>HTTR[3:0] | **High Temperature Trimming Bits** — See Table 7-25 for trimming effects. |

**Table 7-25. Trimming Effect of HTTR**

| Bit | Trimming Effect |
|---|---|
| HTTR[3] | Increases $V_{HT}$ twice of HTTR[2] |
| HTTR[2] | Increases $V_{HT}$ twice of HTTR[1] |
| HTTR[1] | Increases $V_{HT}$ twice of HTTR[0] |
| HTTR[0] | Increases $V_{HT}$ (to compensate Temperature Offset) |

### 7.3.2.21 S12CPMU_UHV_V5 IRC1M Trim Registers (CPMUIRCTRIMH / CPMUIRCTRIML)

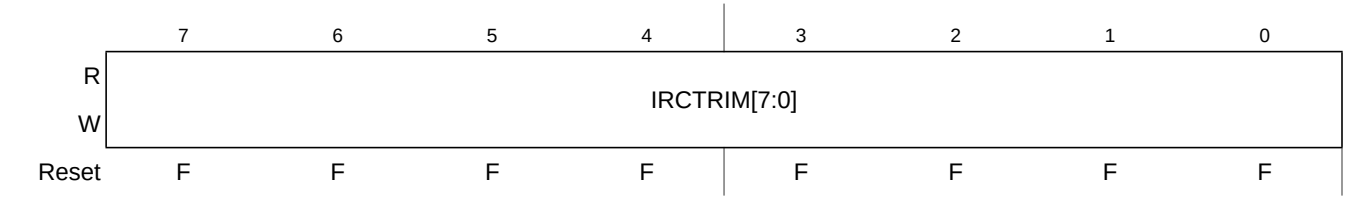
Module Base + 0x0018




After de-assert of System Reset a factory programmed trim value is automatically loaded from the Flash memory to provide trimmed Internal Reference Frequency $f_{IRC1M_TRIM}$.

**Figure 7-27. S12CPMU_UHV_V5 IRC1M Trim High Register (CPMUIRCTRIMH)**

Module Base + 0x0019




After de-assert of System Reset a factory programmed trim value is automatically loaded from the Flash memory to provide trimmed Internal Reference Frequency $f_{IRC1M_TRIM}$.

**Figure 7-28. S12CPMU_UHV_V5 IRC1M Trim Low Register (CPMUIRCTRIML)**

Read: Anytime

Write: Anytime if PROT=0 (CPMUPROT register). Else write has no effect

**NOTE**

Writes to these registers while PLLSEL=1 clears the LOCK and UPOSC status bits.

**Table 7-26. CPMUIRCTRIMH/L Field Descriptions**

| Field | Description |
|---|---|
| 15-11<br>TCTRIM[4:0] | **IRC1M temperature coefficient Trim Bits**<br>Trim bits for the Temperature Coefficient (TC) of the IRC1M frequency.<br>Table 7-27 shows the influence of the bits TCTRIM[4:0] on the relationship between frequency and temperature.<br>Figure 7-30 shows an approximate TC variation, relative to the nominal TC of the IRC1M (i.e. for TCTRIM[4:0]=0x00000 or 0x10000). |
| 9-0<br>IRCTRIM[9:0] | **IRC1M Frequency Trim Bits** — Trim bits for Internal Reference Clock<br>After System Reset the factory programmed trim value is automatically loaded into these registers, resulting in a Internal Reference Frequency $f_{IRC1M_TRIM}$.See device electrical characteristics for value of $f_{IRC1M_TRIM}$.<br>The frequency trimming consists of two different trimming methods:<br>A rough trimming controlled by bits IRCTRIM[9:6] can be done with frequency leaps of about 6% in average.<br>A fine trimming controlled by bits IRCTRIM[5:0] can be done with frequency leaps of about 0.3% (this trimming determines the precision of the frequency setting of 0.15%, i.e. 0.3% is the distance between two trimming values).<br>Figure 7-29 shows the relationship between the trim bits and the resulting IRC1M frequency. |

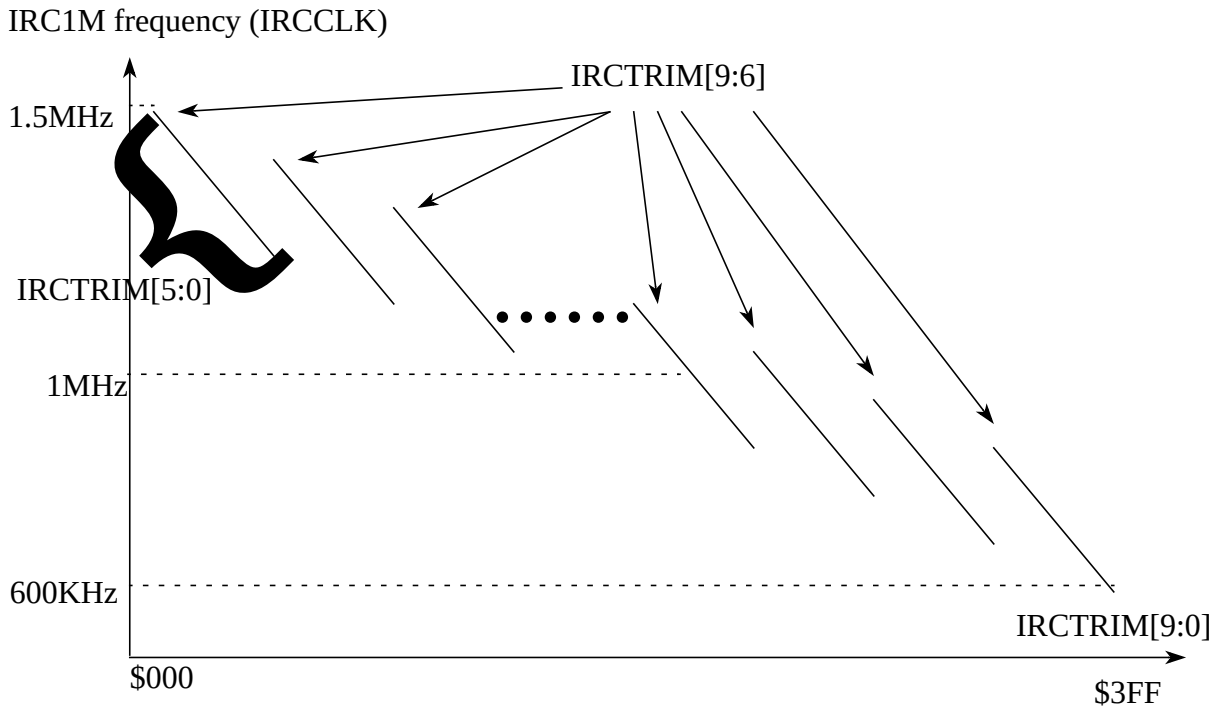



**Figure 7-29. IRC1M Frequency Trimming Diagram**

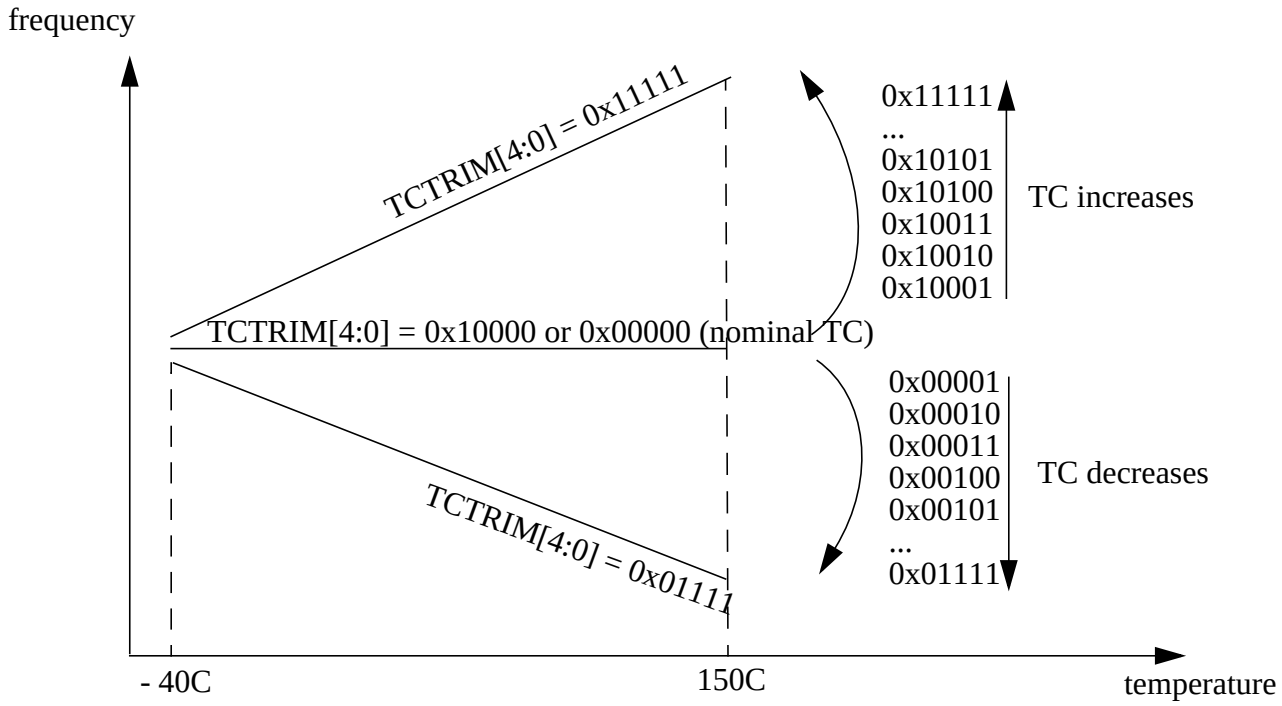



**Figure 7-30. Influence of TCTRIM[4:0] on the Temperature Coefficient**

### NOTE

The frequency is not necessarily linear with the temperature (in most cases it will not be). The above diagram is meant only to give the direction (positive or negative) of the variation of the TC, relative to the nominal TC.

Setting TCTRIM[4:0] at 0x00000 or 0x10000 does not mean that the temperature coefficient will be zero. These two combinations basically switch off the TC compensation module, which results in the nominal TC of the IRC1M.

**Table 7-27. TC trimming of the frequency of the IRC1M at ambient temperature**

| TCTRIM[4:0] | IRC1M Indicative relative TC variation | IRC1M indicative frequency drift for relative TC variation |
|---|---|---|
| 00000 | 0 (nominal TC of the IRC) | 0% |
| 00001 | -0.27% | -0.5% |
| 00010 | -0.54% | -0.9% |
| 00011 | -0.81% | -1.3% |
| 00100 | -1.08% | -1.7% |
| 00101 | -1.35% | -2.0% |
| 00110 | -1.63% | -2.2% |
| 00111 | -1.9% | -2.5% |
| 01000 | -2.20% | -3.0% |
| 01001 | -2.47% | -3.4% |
| 01010 | -2.77% | -3.9% |
| 01011 | -3.04 | -4.3% |
| 01100 | -3.33% | -4.7% |
| 01101 | -3.6% | -5.1% |
| 01110 | -3.91% | -5.6% |
| 01111 | -4.18% | -5.9% |
| 10000 | 0 (nominal TC of the IRC) | 0% |
| 10001 | +0.27% | +0.5% |
| 10010 | +0.54% | +0.9% |
| 10011 | +0.81% | +1.3% |
| 10100 | +1.07% | +1.7% |
| 10101 | +1.34% | +2.0% |
| 10110 | +1.59% | +2.2% |
| 10111 | +1.86% | +2.5% |
| 11000 | +2.11% | +3.0% |
| 11001 | +2.38% | +3.4% |
| 11010 | +2.62% | +3.9% |
| 11011 | +2.89% | +4.3% |
| 11100 | +3.12% | +4.7% |
| 11101 | +3.39% | +5.1% |
| 11110 | +3.62% | +5.6% |
| 11111 | +3.89% | +5.9% |

**NOTE**

Since the IRC1M frequency is not a linear function of the temperature, but more like a parabola, the above relative variation is only an indication and should be considered with care.

Be aware that the output frequency varies with the TC trimming. A frequency trimming correction is therefore necessary. The values provided in Table 7-27 are typical values at ambient temperature which can vary from device to device.

### 7.3.2.22 S12CPMU_UHV_V5 Oscillator Register (CPMUOSC)

This registers configures the external oscillator (XOSCLCP).

Module Base + 0x001A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | OSCE | 0 | Reserved | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-31. S12CPMU_UHV_V5 Oscillator Register (CPMUOSC)**

Read: Anytime

Write: Anytime if PROT=0 (CPMUPROT register) and PLLSEL=1 (CPMUCLKS register). Else write has no effect.

**NOTE**.

Write to this register clears the LOCK and UPOSC status bits.

**Table 7-28. CPMUOSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>OSCE | **Oscillator Enable Bit** — This bit enables the external oscillator (XOSCLCP). The UPOSC status bit in the CPMIUFLG register indicates when the oscillation is stable and when OSCCLK can be selected as source of the Bus Clock or source of the COP or RTI.**If the oscillator clock monitor reset is enabled (OMRE = 1 in CPMUOSC2 register), then a loss of oscillation will lead to an oscillator clock monitor reset.**<br>0 External oscillator is disabled.<br>REFCLK for PLL is IRCCLK.<br>1 External oscillator is enabled.<br>Oscillator clock monitor is enabled.<br>External oscillator is qualified by PLLCLK.<br>REFCLK for PLL is the external oscillator clock divided by REFDIV.<br>**If OSCE bit has been set (write "1") the EXTAL and XTAL pins are exclusively reserved for the oscillator and they can not be used anymore as general purpose I/O until the next system reset.**<br><br>**Note:** When starting up the external oscillator (either by programming OSCE bit to 1 or on exit from Full Stop Mode with OSCE bit already 1) the software must wait for a minimum time equivalent to the startup-time of the external oscillator $t_{UPOSC}$ before entering Pseudo Stop Mode. |
| 5<br>Reserved | Do not alter this bit from its reset value. It is for Manufacturer use only and can change the Oscillator behavior. |

### 7.3.2.23 S12CPMU_UHV_V5 Protection Register (CPMUPROT)

This register protects the clock configuration registers from accidental overwrite:

CPMUSYNR, CPMUREFDIV, CPMUCLKS, CPMUPLL, CPMUIRCTRIMH/L, CPMUOSC and CPMUOSC2

Module Base + 0x001B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PROT |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-32. S12CPMU_UHV_V5 Protection Register (CPMUPROT)**

Read: Anytime

Write: Anytime

| Field | Description |
|---|---|
| PROT | **Clock Configuration Registers Protection Bit** — This bit protects the clock configuration registers from accidental overwrite (see list of protected registers above): Writing 0x26 to the CPMUPROT register clears the PROT bit, other write accesses set the PROT bit.<br>0 Protection of clock configuration registers is disabled.<br>1 Protection of clock configuration registers is enabled. (see list of protected registers above). |

### 7.3.2.24 Reserved Register CPMUTEST2

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in Special Mode can alter the S12CPMU_UHV_V5's functionality.

Module Base + 0x001C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-33. Reserved Register CPMUTEST2**

Read: Anytime

Write: Only in Special Mode

### 7.3.2.25 Voltage Regulator Control Register (CPMUVREGCTL)

The CPMUVREGCTL allows to enable or disable certain parts of the voltage regulator.This register must be configured after system startup.

Module Base + 0x001D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | EXTXON | INTXON |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

= Unimplemented or Reserved

**Figure 7-34. Voltage Regulator Control Register (CPMUVREGCTL)**

Read: Anytime

Write: Once in normal modes, anytime in special modes

**Table 7-29. Effects of writing the EXTXON and INTXON bits**

| value of EXTXON to be written | value of INTXON to be written | Write Access |
|---|---|---|
| 0 | 0 | blocked, no effect |
| 0 | 1 | legal access |
| 1 | 0 | legal access |
| 1 | 1 | blocked, no effect |

**Table 7-30. CPMUVREGCTL Field Descriptions**

| Field | Description |
|---|---|
| 1<br>EXTXON | **External voltage regulator Enable Bit for VDDX domain** — Should be set to 1 if external BJT is present on the PCB, cleared otherwise.<br>0 VDDX control loop does not use external BJT<br>1 VDDX control loop uses external BJT |
| 0<br>INTXON | **Internal voltage regulator Enable Bit for VDDX domain**— Should be set to 1 if no external BJT is present on the PCB, cleared otherwise.<br>0 VDDX control loop does not use internal power transistor<br>1 VDDX control loop uses internal power transistor |

### 7.3.2.26 S12CPMU_UHV_V5 Oscillator Register 2 (CPMUOSC2)

This registers configures the external oscillator (XOSCLCP).

Module Base + 0x001E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | OMRE | OSCMOD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 7-35. S12CPMU_UHV_V5 Oscillator Register 2 (CPMUOSC2)**

Read: Anytime

Write: Anytime if PROT=0 (CPMUPROT register) and PLLSEL=1 (CPMUCLKS register). Else write has no effect.

**Table 7-31. CPMUOSC2 Field Descriptions**

| Field | Description |
|---|---|
| 0<br>OSCMOD | This bit selects the mode of the external oscillator (XOSCLCP)<br>If OSCE bit in CPMUOSC register is 1, then the OSCMOD bit can not be changed (writes will have no effect).<br>0 External oscillator configured for loop controlled mode (reduced amplitude on EXTAL and XTAL))<br>1 External oscillator configured for full swing mode (full swing amplitude on EXTAL and XTAL) |
| 1<br>OMRE | This bit enables the oscillator clock monitor reset. If OSCE bit in CPMUOSC register is 1, then the OMRE bit can not be changed (writes will have no effect).<br>0 Oscillator clock monitor reset is disabled<br>1 Oscillator clock monitor reset is enabled |

# 7.4 Functional Description

## 7.4.1 Phase Locked Loop with Internal Filter (PLL)

The PLL is used to generate a high speed PLLCLK based on a low frequency REFCLK.

The REFCLK is by default the IRCCLK which is trimmed to $f_{IRC1M_TRIM}$=1MHz.

If using the oscillator (OSCE=1) REFCLK will be based on OSCCLK. For increased flexibility, OSCCLK can be divided in a range of 1 to 16 to generate the reference frequency REFCLK using the REFDIV[3:0] bits. Based on the SYNDIV[5:0] bits the PLL generates the VCOCLK by multiplying the reference clock by a 2, 4, 6,... 126, 128. Based on the POSTDIV[4:0] bits the VCOCLK can be divided in a range of 1,2, 3, 4, 5, 6,... to 32 to generate the PLLCLK.

If oscillator is enabled (OSCE=1) $$f_{REF} = \frac{f_{OSC}}{(REFDIV + 1)}$$

If oscillator is disabled (OSCE=0) $$f_{REF} = f_{IRC1M}$$

$$f_{VCO} = 2 \times f_{REF} \times (SYNDIV + 1)$$

If PLL is locked (LOCK=1) $$f_{PLL} = \frac{f_{VCO}}{(POSTDIV + 1)}$$

If PLL is not locked (LOCK=0) $$f_{PLL} = \frac{f_{VCO}}{4}$$

If PLL is selected (PLLSEL=1) $$f_{bus} = \frac{f_{PLL}}{2}$$

**NOTE**

Although it is possible to set the dividers to command a very high clock frequency, do not exceed the specified bus frequency limit for the MCU.

Several examples of PLL divider settings are shown in Table 7-32. The following rules help to achieve optimum stability and shortest lock time:

- Use lowest possible $f_{VCO}$ / $f_{REF}$ ratio (SYNDIV value).
- Use highest possible REFCLK frequency $f_{REF}$.

**Table 7-32. Examples of PLL Divider Settings**

| $f_{osc}$ | REFDIV[3:0] | $f_{REF}$ | REFFRQ[1:0] | SYNDIV[5:0] | $f_{VCO}$ | VCOFRQ[1:0] | POSTDIV[4:0] | $f_{PLL}$ | $f_{bus}$ |
|---|---|---|---|---|---|---|---|---|---|
| off | $00 | 1MHz | 00 | $18 | 50MHz | 01 | $03 | 12.5MHz | 6.25MHz |
| off | $00 | 1MHz | 00 | $18 | 50MHz | 01 | $00 | 50MHz | 25MHz |
| 4MHz | $00 | 4MHz | 01 | $05 | 48MHz | 00 | $00 | 48MHz | 24MHz |

The phase detector inside the PLL compares the feedback clock (FBCLK = VCOCLK/(SYNDIV+1)) with the reference clock (REFCLK = (IRC1M or OSCCLK)/(REFDIV+1)). Correction pulses are generated based on the phase difference between the two signals. The loop filter alters the DC voltage on the internal filter capacitor, based on the width and direction of the correction pulse which leads to a higher or lower VCO frequency.

The user must select the range of the REFCLK frequency (REFFRQ[1:0] bits) and the range of the VCOCLK frequency (VCOFRQ[1:0] bits) to ensure that the correct PLL loop bandwidth is set.

The lock detector compares the frequencies of the FBCLK and the REFCLK. Therefore the speed of the lock detector is directly proportional to the reference clock frequency. The circuit determines the lock condition based on this comparison. So e.g. a failure in the reference clock will cause the PLL not to lock.

If PLL LOCK interrupt requests are enabled, the software can wait for an interrupt request and for instance check the LOCK bit. If interrupt requests are disabled, software can poll the LOCK bit continuously (during PLL start-up) or at periodic intervals. In either case, only when the LOCK bit is set, the VCOCLK will have stabilized to the programmed frequency.

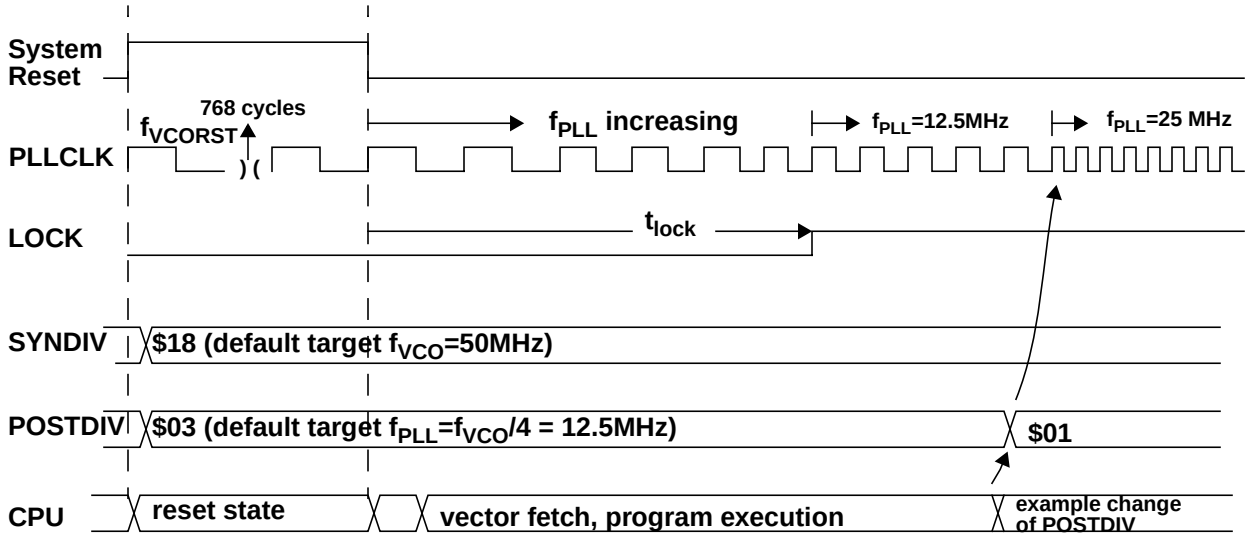
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within the tolerance, $\Delta_{Lock}$, and is cleared when the VCO frequency is out of the tolerance, $\Delta_{unl}$.
- Interrupt requests can occur if enabled (LOCKIE = 1) when the lock condition changes, toggling the LOCK bit.

In case of loss of reference clock (e.g. IRCCLK) the PLL will not lock or if already locked, then it will unlock. The frequency of the VCOCLK will be very low and will depend on the value of the VCOFRQ[1:0] bits.

## 7.4.2 Startup from Reset

An example for startup of the clock system from Reset is given in Figure 7-36.

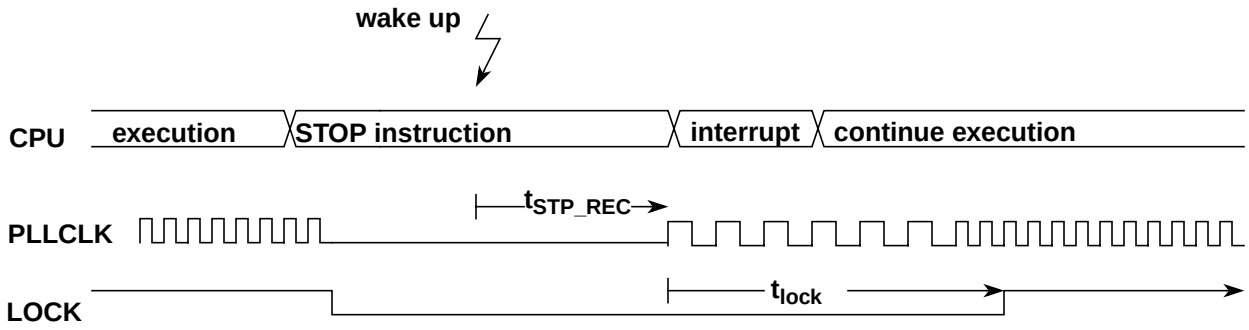
**Figure 7-36. Startup of clock system after Reset**




## 7.4.3 Stop Mode using PLLCLK as source of the Bus Clock

An example of what happens going into Stop Mode and exiting Stop Mode after an interrupt is shown in Figure 7-37. Disable PLL Lock interrupt (LOCKIE=0) before going into Stop Mode.

**Figure 7-37. Stop Mode using PLLCLK as source of the Bus Clock**




Depending on the COP configuration there might be an additional significant latency time until COP is active again after exit from Stop Mode due to clock domain crossing synchronization. This latency time of 2 ACLK cycles occurs if COP clock source is ACLK and the CSAD bit is set and must be added to the device Stop Mode recovery time $t_{STP_REC}$. After exit from Stop Mode (Pseudo, Full) for this latency time

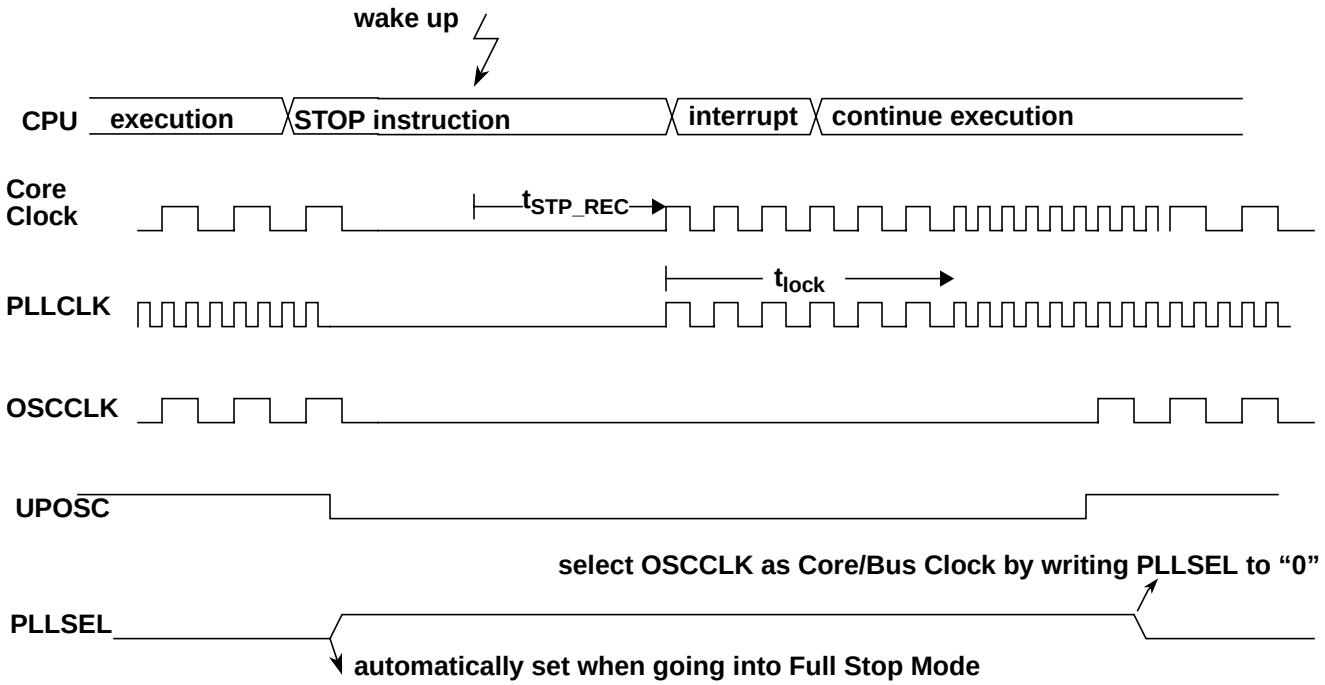
of 2 ACLK cycles no Stop Mode request (STOP instruction) should be generated to make sure the COP counter can increment at each Stop Mode exit.

### 7.4.4 Full Stop Mode using Oscillator Clock as source of the Bus Clock

An example of what happens going into Full Stop Mode and exiting Full Stop Mode after an interrupt is shown in Figure 7-38.

Disable PLL Lock interrupt (LOCKIE=0) and oscillator status change interrupt (OSCIE=0) before going into Full Stop Mode.

**Figure 7-38. Full Stop Mode using Oscillator Clock as source of the Bus Clock**



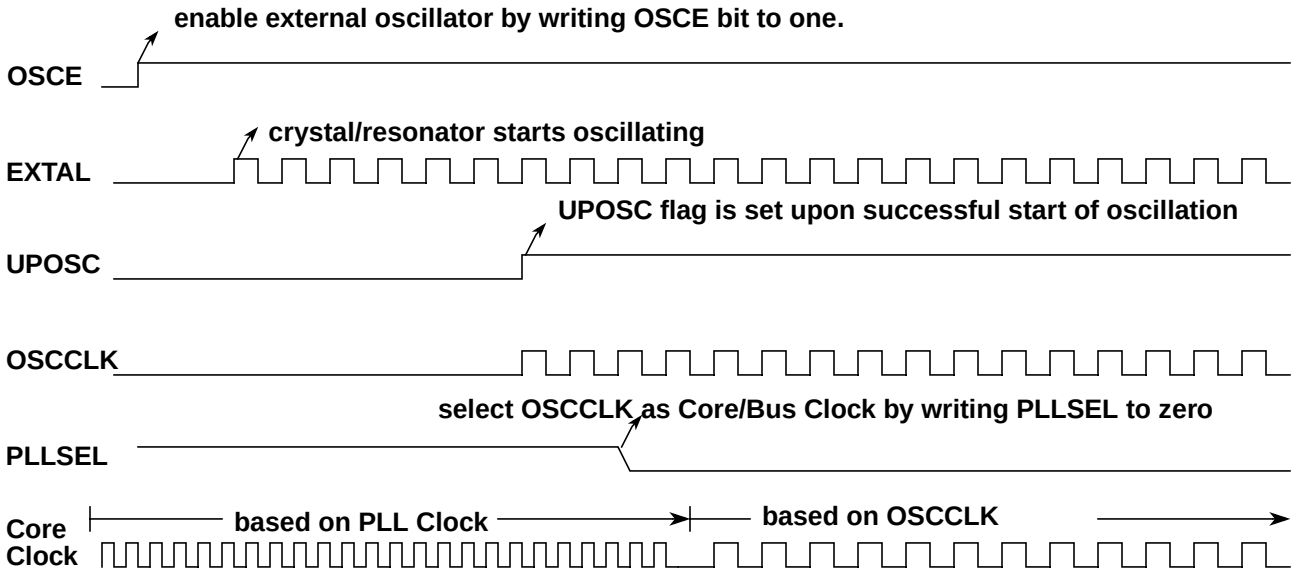

Depending on the COP configuration there might be a significant latency time until COP is active again after exit from Stop Mode due to clock domain crossing synchronization. This latency time of 2 ACLK cycles occurs if COP clock source is ACLK and the CSAD bit is set and must be added to the device Stop Mode recovery time $t_{STP_REC}$. After exit from Stop Mode (Pseudo, Full) for this latency time of 2 ACLK cycles no Stop Mode request (STOP instruction) should be generated to make sure the COP counter can increment at each Stop Mode exit.

## 7.4.5 External Oscillator

### 7.4.5.1 Enabling the External Oscillator

An example of how to use the oscillator as source of the Bus Clock is shown in Figure 7-39.

**Figure 7-39. Enabling the external oscillator**



## 7.4.6 System Clock Configurations

### 7.4.6.1 PLL Engaged Internal Mode (PEI)

This mode is the default mode after System Reset or Power-On Reset.

The Bus Clock is based on the PLLCLK, the reference clock for the PLL is internally generated (IRC1M). The PLL is configured to 50 MHz VCOCLK with POSTDIV set to 0x03. If locked (LOCK=1) this results in a PLLCLK of 12.5 MHz and a Bus Clock of 6.25 MHz. The PLL can be re-configured to other bus frequencies.

The clock sources for COP and RTI can be based on the internal reference clock generator (IRC1M) or the RC-Oscillator (ACLK).

### 7.4.6.2 PLL Engaged External Mode (PEE)

In this mode, the Bus Clock is based on the PLLCLK as well (like PEI). The reference clock for the PLL is based on the external oscillator.

The clock sources for COP and RTI can be based on the internal reference clock generator or on the external oscillator clock or the RC-Oscillator (ACLK).

This mode can be entered from default mode PEI by performing the following steps:

1. Configure the PLL for desired bus frequency.
2. Enable the external Oscillator (OSCE bit).
3. Wait for oscillator to start-up and the PLL being locked (LOCK = 1) and (UPOSC =1).
4. Clear all flags in the CPMUIFLG register to be able to detect any future status bit change.
5. Optionally status interrupts can be enabled (CPMUINT register).

Loosing PLL lock status (LOCK=0) means loosing the oscillator status information as well (UPOSC=0).

The impact of loosing the oscillator status (UPOSC=0) in PEE mode is as follows:

- The PLLCLK is derived from the VCO clock (with its actual frequency) divided by four until the PLL locks again.

Application software needs to be prepared to deal with the impact of loosing the oscillator status at any time.

#### 7.4.6.3 PLL Bypassed External Mode (PBE)

In this mode, the Bus Clock is based on the external oscillator clock. The reference clock for the PLL is based on the external oscillator.

The clock sources for COP and RTI can be based on the internal reference clock generator or on the external oscillator clock or the RC-Oscillator (ACLK).

This mode can be entered from default mode PEI by performing the following steps:

1. Make sure the PLL configuration is valid.
2. Enable the external Oscillator (OSCE bit)
3. Wait for the oscillator to start-up and the PLL being locked (LOCK = 1) and (UPOSC =1)
4. Clear all flags in the CPMUIFLG register to be able to detect any status bit change.
5. Optionally status interrupts can be enabled (CPMUINT register).
6. Select the Oscillator clock as source of the Bus clock (PLLSEL=0)

Loosing PLL lock status (LOCK=0) means loosing the oscillator status information as well (UPOSC=0).

The impact of loosing the oscillator status (UPOSC=0) in PBE mode is as follows:

- PLLSEL is set automatically and the Bus clock is switched back to the PLL clock.
- The PLLCLK is derived from the VCO clock (with its actual frequency) divided by four until the PLL locks again.

Application software needs to be prepared to deal with the impact of loosing the oscillator status at any time.

## 7.5 Resets

### 7.5.1 General

All reset sources are listed in Table 7-33. There is only one reset vector for all these reset sources. Refer to MCU specification for reset vector address.

**Table 7-33. Reset Summary**

| Reset Source | Local Enable |
|---|---|
| Power-On Reset (POR) | None |
| Low Voltage Reset (LVR) | None |
| External pin $\overline{\text{RESET}}$ | None |
| PLL Clock Monitor Reset | None |
| Oscillator Clock Monitor Reset | OSCE Bit in CPMUOSC register and OMRE Bit in CPMUOSC2 register |
| COP Reset | CR[2:0] in CPMUCOP register |

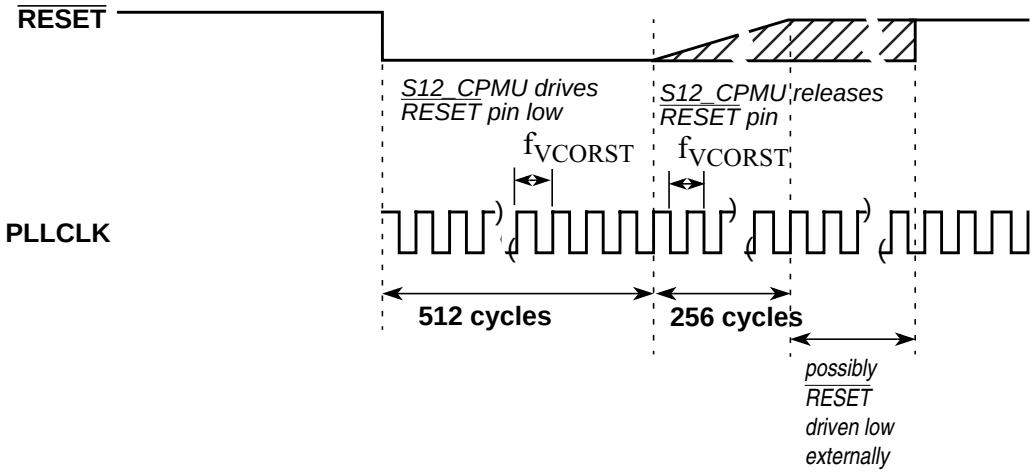
## 7.5.2 Description of Reset Operation

Upon detection of any reset of Table 7-33, an internal circuit drives the $\overline{\text{RESET}}$ pin low for 512 PLLCLK cycles. After 512 PLLCLK cycles the $\overline{\text{RESET}}$ pin is released. The internal reset of the MCU remains asserted while the reset generator completes the 768 PLLCLK cycles long reset sequence.In case the $\overline{\text{RESET}}$ pin is externally driven low for more than these 768 PLLCLK cycles (External Reset), the internal reset remains asserted longer.

**NOTE**

While System Reset is asserted the PLLCLK runs with the frequency $f_{VCORST}$.

**Figure 7-40. RESET Timing**




## 7.5.3 Oscillator Clock Monitor Reset

If the external oscillator is enabled (OSCE=1)and the oscillator clock monitor reset is enabled (OMRE=1), then in case of loss of oscillation or the oscillator frequency drops below the failure assert frequency $f_{CMFA}$ (see device electrical characteristics for values), the S12CPMU_UHV_V5 generates an Oscillator Clock Monitor Reset. In Full Stop Mode the external oscillator and the oscillator clock monitor are disabled.

## 7.5.4 PLL Clock Monitor Reset

In case of loss of PLL clock oscillation or the PLL clock frequency is below the failure assert frequency $f_{PMFA}$ (see device electrical characteristics for values), the S12CPMU_UHV_V5 generates a PLL Clock Monitor Reset. In Full Stop Mode the PLL and the PLL clock monitor are disabled.

### 7.5.5 Computer Operating Properly Watchdog (COP) Reset

The COP (free running watchdog timer) enables the user to check that a program is running and sequencing properly. When the COP is being used, software is responsible for keeping the COP from timing out. If the COP times out it is an indication that the software is no longer being executed in the intended sequence; thus COP reset is generated.

The clock source for the COP is either ACLK, IRCCLK or OSCCLK depending on the setting of the COPOSCSEL0 and COPOSCSEL1 bit.
Due to clock domain crossing synchronization there is a latency time to enter and exit Stop Mode if the COP clock source is ACLK and this clock is stopped in Stop Mode. This maximum total latency time is 4 ACLK cycles (2 ACLK cycles for Stop Mode entry and exit each) which must be added to the Stop Mode recovery time $t_{STP_REC}$ from exit of current Stop Mode to entry of next Stop Mode. This latency time occurs no matter which Stop Mode (Full, Pseudo) is currently exited or entered next.
After exit from Stop Mode (Pseudo, Full) for this latency time of 2 ACLK cycles no Stop Mode request (STOP instruction) should be generated to make sure the COP counter can increment at each Stop Mode exit.
Table 7-34 gives an overview of the COP condition (run, static) in Stop Mode depending on legal configuration and status bit settings:

**Table 7-34. COP condition (run, static) in Stop Mode**

| COPOSCSEL1 | CSAD | PSTP | PCE | COPOSCSEL0 | OSCE | UPOSC | COP counter behavior in Stop Mode (clock source) |
|---|---|---|---|---|---|---|---|
| 1 | 0 | x | x | x | x | x | Run (ACLK) |
| 1 | 1 | x | x | x | x | x | Static (ACLK) |
| 0 | x | 1 | 1 | 1 | 1 | 1 | Run (OSCCLK) |
| 0 | x | 1 | 1 | 0 | 0 | x | Static (IRCCLK) |
| 0 | x | 1 | 1 | 0 | 1 | x | Static (IRCCLK) |
| 0 | x | 1 | 0 | 0 | x | x | Static (IRCCLK) |
| 0 | x | 1 | 0 | 1 | 1 | 1 | Static (OSCCLK) |
| 0 | x | 0 | 1 | 1 | 1 | 1 | Static (OSCCLK) |
| 0 | x | 0 | 1 | 0 | 1 | x | Static (IRCCLK) |
| 0 | x | 0 | 1 | 0 | 0 | 0 | Static (IRCCLK) |
| 0 | x | 0 | 0 | 1 | 1 | 1 | Satic (OSCCLK) |
| 0 | x | 0 | 0 | 0 | 1 | 1 | Static (IRCCLK) |
| 0 | x | 0 | 0 | 0 | 1 | 0 | Static (IRCCLK) |
| 0 | x | 0 | 0 | 0 | 0 | 0 | Static (IRCCLK) |

Three control bits in the CPMUCOP register allow selection of seven COP time-out periods.

When COP is enabled, the program must write $55 and $AA (in this order) to the CPMUARMCOP register during the selected time-out period. Once this is done, the COP time-out period is restarted. If the program fails to do this and the COP times out, a COP reset is generated. Also, if any value other than $55 or $AA is written, a COP reset is generated.

Windowed COP operation is enabled by setting WCOP in the CPMUCOP register. In this mode, writes to the CPMUARMCOP register to clear the COP timer must occur in the last 25% of the selected time-out period. A premature write will immediately reset the part.

In MCU Normal Mode the COP time-out period (CR[2:0]) and COP window (WCOP) setting can be automatically pre-loaded at reset release from NVM memory (if values are defined in the NVM by the application). By default the COP is off and no window COP feature is enabled after reset release via NVM memory. The COP control register CPMUCOP can be written once in an application in MCU Normal Mode to update the COP time-out period (CR[2:0]) and COP window (WCOP) setting loaded from NVM memory at reset release. Any value for the new COP time-out period and COP window setting is allowed except COP off value if the COP was enabled during pre-load via NVM memory.
The COP clock source select bits can not be pre-loaded via NVM memory at reset release. The IRC clock is the default COP clock source out of reset.
The COP clock source select bits (COPOSCSEL0/1) and ACLK clock control bit in Stop Mode (CSAD) can be modified until the CPMUCOP register write once has taken place. Therefore these control bits should be modified before the final COP time-out period and window COP setting is written.
The CPMUCOP register access to modify the COP time-out period and window COP setting in MCU Normal Mode after reset release must be done with the WRTMASK bit cleared otherwise the update is ignored and this access does not count as the write once.

### 7.5.6 Power-On Reset (POR)

The on-chip POR circuitry detects when the internal supply VDD drops below an appropriate voltage level. The POR is deasserted, if the internal supply VDD exceeds an appropriate voltage level (voltage levels not specified, because the internal supply can not be monitored externally).The POR circuitry is always active. It acts as LVR in Stop Mode.

### 7.5.7 Low-Voltage Reset (LVR)

The on-chip LVR circuitry detects when one of the supply voltages VDD, VDDX and VDDF drops below an appropriate voltage level. If LVR is deasserted the MCU is fully operational at the specified maximum speed. The LVR assert and deassert levels for the supply voltage VDDX are $V_{LVRXA}$ and $V_{LVRXD}$ and are specified in the device Reference Manual.The LVR circuitry is active in Run- and Wait Mode.

## 7.6 Interrupts

The interrupt vectors requested by the S12CPMU_UHV_V5 are listed in Table 7-35. Refer to MCU specification for related vector addresses and priorities.

**Table 7-35. S12CPMU_UHV_V5 Interrupt Vectors**

| Interrupt Source | CCR Mask | Local Enable |
|---|---|---|
| RTI time-out interrupt | I bit | CPMUINT (RTIE) |
| PLL lock interrupt | I bit | CPMUINT (LOCKIE) |
| Oscillator status interrupt | I bit | CPMUINT (OSCIE) |
| Low voltage interrupt | I bit | CPMULVCTL (LVIE) |
| High temperature interrupt | I bit | CPMUHTCTL (HTIE) |
| Autonomous Periodical Interrupt | I bit | CPMUAPICTL (APIE) |

### 7.6.1 Description of Interrupt Operation

#### 7.6.1.1 Real Time Interrupt (RTI)

The clock source for the RTI is either IRCCLK or OSCCLK depending on the setting of the RTIOSCSEL bit. In Stop Mode with PSTP=1 (Pseudo Stop Mode), RTIOSCSEL=1 and PRE=1 the RTI continues to run, else the RTI counter halts in Stop Mode.

The RTI can be used to generate hardware interrupts at a fixed periodic rate. If enabled (by setting RTIE=1), this interrupt will occur at the rate selected by the CPMURTI register. At the end of the RTI time-out period the RTIF flag is set to one and a new RTI time-out period starts immediately.

A write to the CPMURTI register restarts the RTI time-out period.

#### 7.6.1.2 PLL Lock Interrupt

The S12CPMU_UHV_V5 generates a PLL Lock interrupt when the lock condition (LOCK status bit) of the PLL changes, either from a locked state to an unlocked state or vice versa. Lock interrupts are locally disabled by setting the LOCKIE bit to zero. The PLL Lock interrupt flag (LOCKIF) is set to1 when the lock condition has changed, and is cleared to 0 by writing a 1 to the LOCKIF bit.

#### 7.6.1.3 Oscillator Status Interrupt

When the OSCE bit is 0, then UPOSC stays 0. When OSCE=1 the UPOSC bit is set after the LOCK bit is set.

Upon detection of a status change (UPOSC) the OSCIF flag is set. Going into Full Stop Mode or disabling the oscillator can also cause a status change of UPOSC.

Any change in PLL configuration or any other event which causes the PLL lock status to be cleared leads to a loss of the oscillator status information as well (UPOSC=0).

Oscillator status change interrupts are locally enabled with the OSCIE bit.

**NOTE**

Loosing the oscillator status (UPOSC=0) affects the clock configuration of the system[1]. This needs to be dealt with in application software.

#### 7.6.1.4 Low-Voltage Interrupt (LVI)

In FPM the input voltage VDDA is monitored. Whenever VDDA drops below level $V_{LVIA,}$ the status bit LVDS is set to 1. When VDDA rises above level $V_{LVID}$ the status bit LVDS is cleared to 0. An interrupt, indicated by flag LVIF = 1, is triggered by any change of the status bit LVDS if interrupt enable bit LVIE = 1.

#### 7.6.1.5 HTI - High Temperature Interrupt

In FPM the junction temperature $T_J$ is monitored. Whenever $T_J$ exceeds level $T_{HTIA}$ the status bit HTDS is set to 1. Vice versa, HTDS is reset to 0 when $T_J$ get below level $T_{HTID}$. An interrupt, indicated by flag HTIF = 1, is triggered by any change of the status bit HTDS, if interrupt enable bit HTIE = 1.

#### 7.6.1.6 Autonomous Periodical Interrupt (API)

The API sub-block can generate periodical interrupts independent of the clock source of the MCU. To enable the timer, the bit APIFE needs to be set.

The API timer is either clocked by the Autonomous Clock (ACLK - trimmable internal RC oscillator) or the Bus Clock. Timer operation will freeze when MCU clock source is selected and Bus Clock is turned off. The clock source can be selected with bit APICLK. APICLK can only be written when APIFE is not set.

The APIR[15:0] bits determine the interrupt period. APIR[15:0] can only be written when APIFE is cleared. As soon as APIFE is set, the timer starts running for the period selected by APIR[15:0] bits. When the configured time has elapsed, the flag APIF is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1. The timer is re-started automatically again after it has set APIF.

The procedure to change APICLK or APIR[15:0] is first to clear APIFE, then write to APICLK or APIR[15:0], and afterwards set APIFE.

The API Trimming bits ACLKTR[5:0] must be set so the minimum period equals 0.2 ms if stable frequency is desired.

See Table 7-20 for the trimming effect of ACLKTR[5:0].

1. For details please refer to “7.4.6 System Clock Configurations”

**NOTE**

The first period after enabling the counter by APIFE might be reduced by API start up delay $t_{sdel}$.

It is possible to generate with the API a waveform at the external pin API_EXTCLK by setting APIFE and enabling the external access with setting APIEA.

## 7.7 Initialization/Application Information

### 7.7.1 General Initialization Information

Usually applications run in MCU Normal Mode.

It is recommended to write the CPMUCOP register in any case from the application program initialization routine after reset no matter if the COP is used in the application or not, even if a configuration is loaded via the flash memory after reset. By doing a “controlled” write access in MCU Normal Mode (with the right value for the application) the write once for the COP configuration bits (WCOP,CR[2:0]) takes place which protects these bits from further accidental change. In case of a program sequencing issue (code runaway) the COP configuration can not be accidentally modified anymore.

### 7.7.2 Application information for COP and API usage

In many applications the COP is used to check that the program is running and sequencing properly. Often the COP is kept running during Stop Mode and periodic wake-up events are needed to service the COP on time and maybe to check the system status.

For such an application it is recommended to use the ACLK as clock source for both COP and API. This guarantees lowest possible IDD current during Stop Mode. Additionally it eases software implementation using the same clock source for both, COP and API.

The Interrupt Service Routine (ISR) of the Autonomous Periodic Interrupt API should contain the write instruction to the CPMUARMCOP register. The value (byte) written is derived from the “main routine” (alternating sequence of $55 and $AA) of the application software.

Using this method, then in the case of a runtime or program sequencing issue the application “main routine” is not executed properly anymore and the alternating values are not provided properly. Hence the COP is written at the correct time (due to independent API interrupt request) but the wrong value is written (alternating sequence of $55 and $AA is no longer maintained) which causes a COP reset.

If the COP is stopped during any Stop Mode it is recommended to service the COP shortly before Stop Mode is entered.

### 7.7.3 Application Information for PLL and Oscillator Startup

The following C-code example shows a recommended way of setting up the system clock system using the PLL and Oscillator:

```
/* Procedure proposed by to setup PLL and Oscillator */
/* example for OSC = 4 MHz and Bus Clock = 25MHz, That is VCOCLK = 50MHz */

/* Initialize */
/* PLL Clock = 50 MHz, divide by one */
CPMUPOSTDIV = 0x00;

/* Generally: Whenever changing PLL reference clock (REFCLK) frequency to a higher value */
/* it is recommended to write CPMUSYNR = 0x00 in order to stay within specified */
/* maximum frequency of the MCU */
CPMUSYNR = 0x00;

/* configure PLL reference clock (REFCLK) for usage with Oscillator */
/* OSC=4MHz divide by 4 (3+1) = 1MHz, REFCLK range 1MHz to 2 MHz (REFFRQ[1:0] = 00) */
CPMUREFDV = 0x03;

/* enable external Oscillator, switch PLL reference clock (REFCLK) to OSC */
CPMUOSC = 0x80;

/* multiply REFCLK = 1MHz by 2*(24+1)*1MHz = 50MHz */
/* VCO range 48 to 80 MHz (VCOFRQ[1:0] = 01) */
CPMUSYNR = 0x58;

/* clear all flags, especially LOCKIF and OSCIF */
CPMUIFLG = 0xFF;

/* put your code to loop and wait for the LOCKIF and OSCIF or */
/* poll CPMUIFLG register until both UPOSC and LOCK status are “1” */
/* that is CPMIFLG == 0x1B */

/*..............continue to your main code execution here..............*/

/* in case later in your code you want to disable the Oscillator and use the */
/* 1MHz IRCCLK as PLL reference clock */

/* Generally: Whenever changing PLL reference clock (REFCLK) frequency to a higher value */
/* it is recommended to write CPMUSYNR = 0x00 in order to stay within specified */
/* maximum frequency of the MCU */
CPMUSYNR = 0x00;

/* disable OSC and switch PLL reference clock to IRC */
CPMUOSC = 0x00;

/* multiply REFCLK = 1MHz by 2*(24+1)*1MHz = 50MHz */
/* VCO range 48 to 80 MHz (VCOFRQ[1:0] = 01) */
CPMUSYNR = 0x58;

/* clear all flags, especially LOCKIF and OSCIF */
CPMUIFLG = 0xFF;

/* put your code to loop and wait for the LOCKIF or */
/* poll CPMUIFLG register until both LOCK status is “1” */
/* that is CPMIFLG == 0x18 */

/*..............continue to your main code execution here..............*/
```

# Chapter 8
# Timer Module (TIM16B8CV3) Block Description

**Table 8-1.**

| | | | |
|---|---|---|---|
| V03.00 | Jan. 28, 2009 | | Initial version |
| V03.01 | Aug. 26, 2009 | 8.1.2/8-288<br>Figure 1-4./1-8<br>8.3.2.15/8-304<br>8.3.2.2/8-294,<br>8.3.2.3/8-294,<br>8.3.2.4/8-295,<br>8.4.3/8-310 | - Correct typo: TSCR ->TSCR1;<br>- Correct typo: ECTxxx->TIMxxx<br>- Correct reference: Figure 8-25 -> Figure 8-30<br>- Add description, “a counter overflow when TTOV[7] is set”, to be the condition of channel 7 override event.<br>- Phrase the description of OC7M to make it more explicit |
| V03.02 | Apri,12,2010 | 8.3.2.8/8-298<br>8.3.2.11/8-301<br>8.4.3/8-310 | -Add Table 8-10<br>-update TCRE bit description<br>-add Figure 8-31 |
| V03.03 | Jan,14,2013 | | -single source generate different channel guide |

## 8.1 Introduction

The basic scalable timer consists of a 16-bit, software-programmable counter driven by a flexible programmable prescaler.

This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

This timer could contain up to 8 input capture/output compare channels with one pulse accumulator available only on channel 7. The input capture function is used to detect a selected transition edge and record the time. The output compare function is used for generating output signals or for timer software delays. The 16-bit pulse accumulator is used to operate as a simple event counter or a gated time accumulator. The pulse accumulator shares timer channel 7 when the channel is available and when in event mode.

A full access for the counter registers or the input capture/output compare registers should take place in one clock cycle. Accessing high byte and low byte separately for all of these registers may not yield the same result as accessing them in one word.

### 8.1.1 Features

The TIM16B8CV3 includes these distinctive features:

- Up to 8 channels available. (refer to device specification for exact number)
- All channels have same input capture/output compare functionality.

- Clock prescaling.
- 16-bit counter.
- 16-bit pulse accumulator on channel 7 .

## 8.1.2 Modes of Operation

Stop: Timer is off because clocks are stopped.

Freeze: Timer counter keeps on running, unless TSFRZ in TSCR1 is set to 1.

Wait: Counters keeps on running, unless TSWAI in TSCR1 is set to 1.

Normal: Timer counter keep on running, unless TEN in TSCR1 is cleared to 0.

## 8.1.3 Block Diagrams

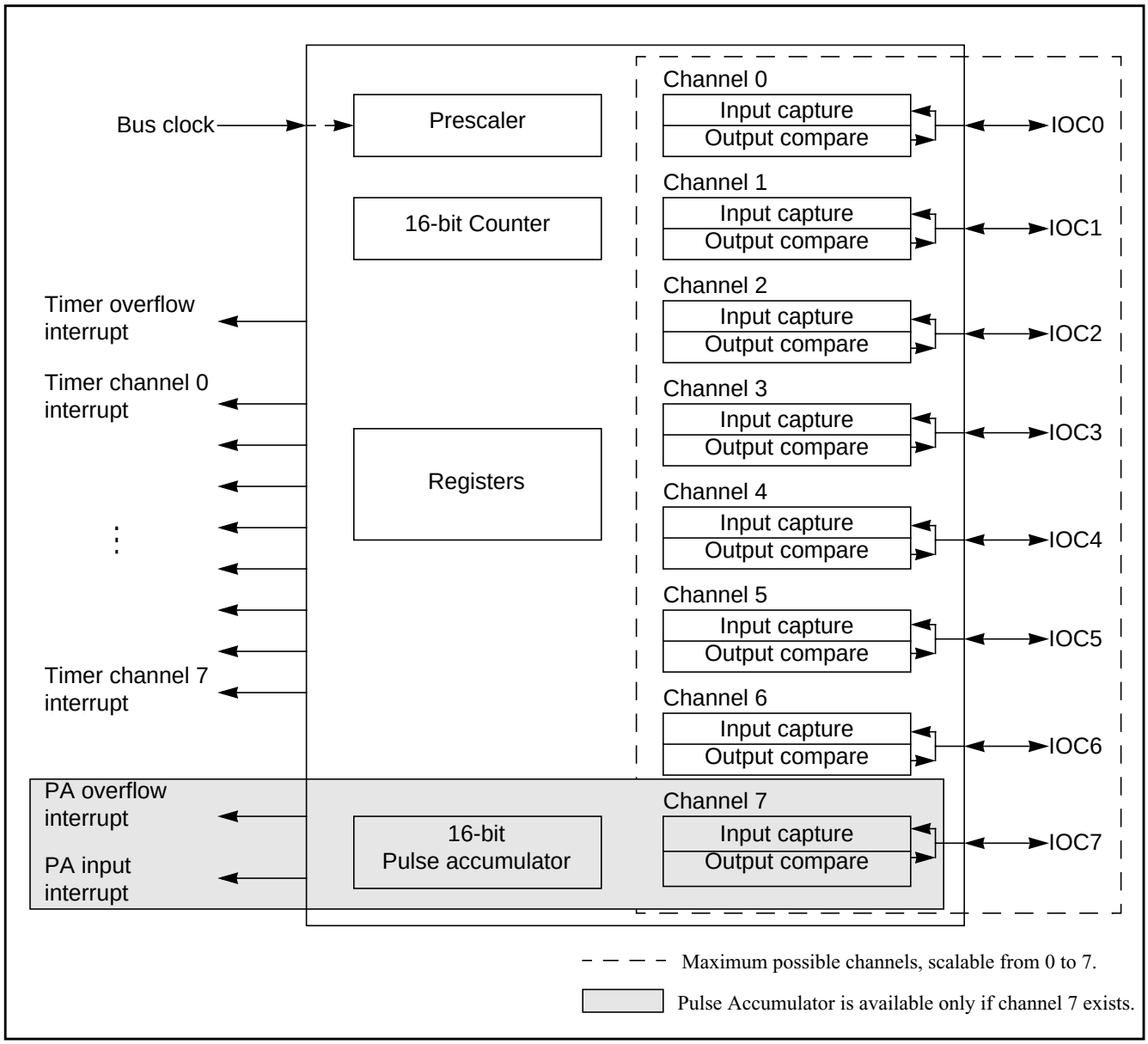



**Figure 8-1. TIM16B8CV3 Block Diagram**

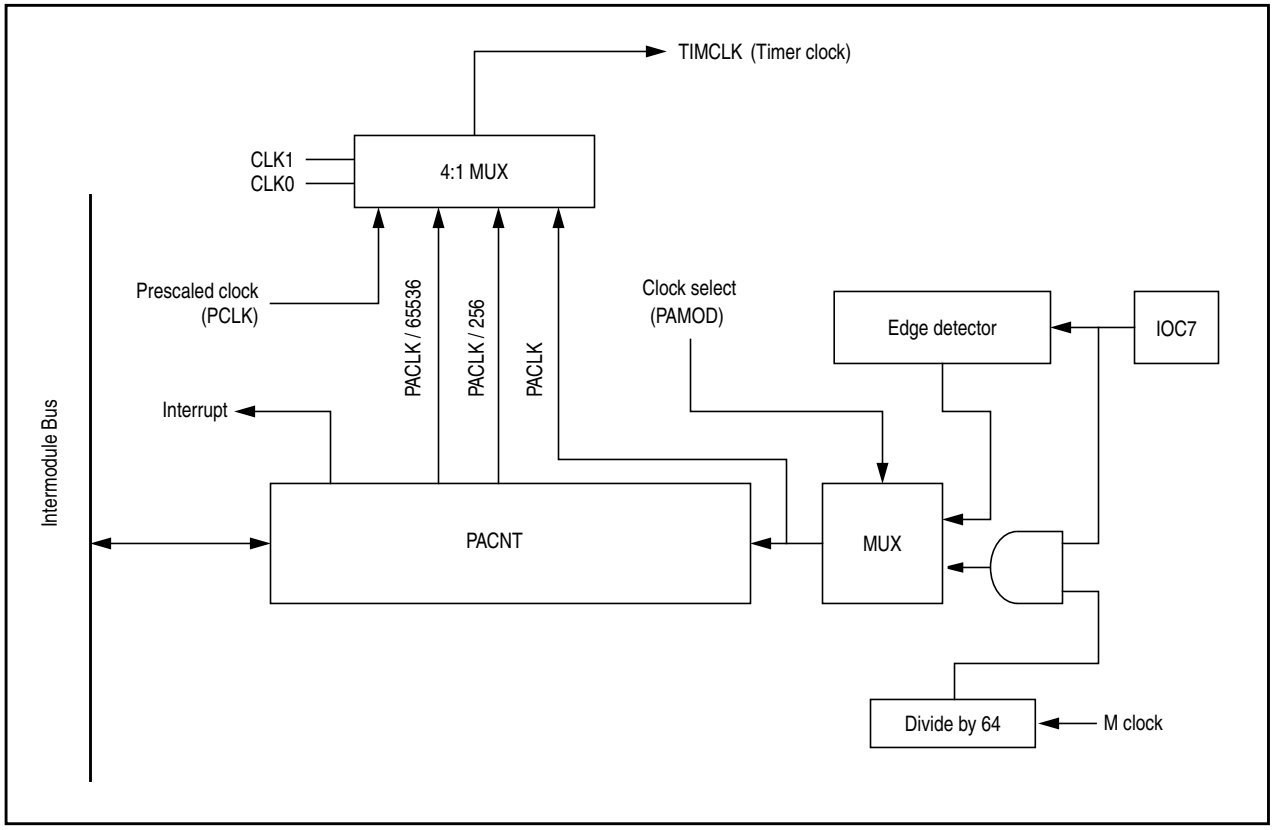



**Figure 8-2. 16-Bit Pulse Accumulator Block Diagram**

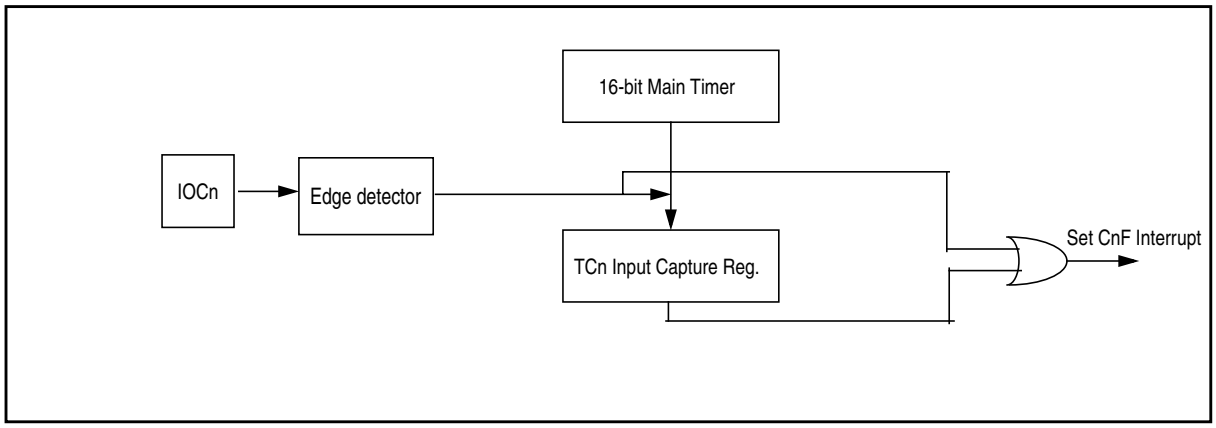



**Figure 8-3. Interrupt Flag Setting**

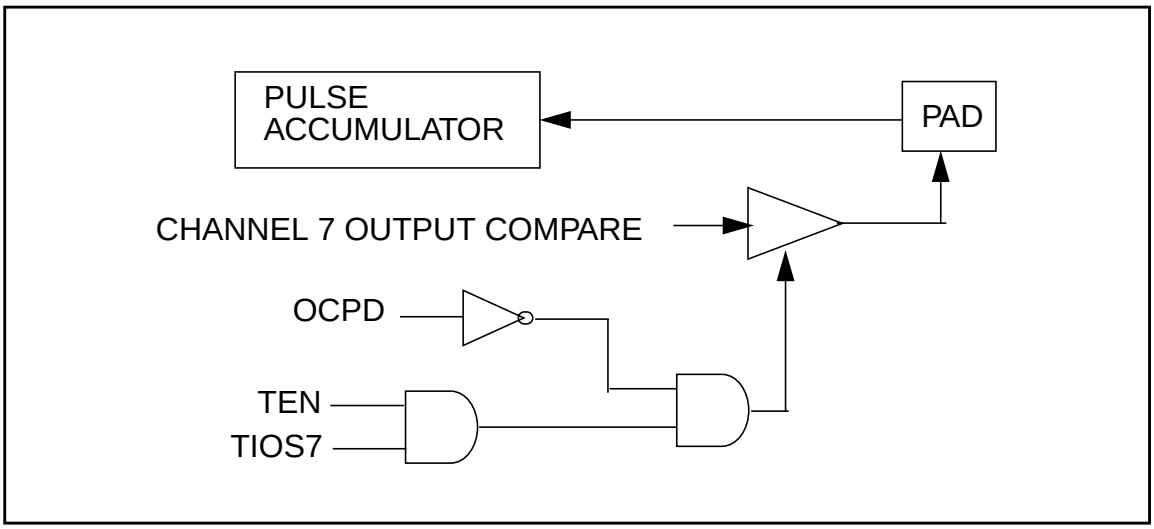



**Figure 8-4. Channel 7 Output Compare/Pulse Accumulator Logic**

## 8.2 External Signal Description

The TIM16B8CV3 module has a selected number of external pins. Refer to device specification for exact number.

### 8.2.1 IOC7 — Input Capture and Output Compare Channel 7

This pin serves as input capture or output compare for channel 7 . This can also be configured as pulse accumulator input.

### 8.2.2 IOC6 - IOC0 — Input Capture and Output Compare Channel 6-0

Those pins serve as input capture or output compare for TIM16B8CV3 channel .

**NOTE**

For the description of interrupts see Section 8.6, “Interrupts”.

## 8.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

### 8.3.1 Module Memory Map

The memory map for the TIM16B8CV3 module is given below in Figure 8-5. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the TIM16B8CV3 module and the address offset for each register.

## 8.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Only bits related to implemented channels are valid.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 TIOS | R | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | W | | | | | | | | |
| 0x0001 CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x0002 OC7M | R | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| | W | | | | | | | | |
| 0x0003 OC7D | R | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| | W | | | | | | | | |
| 0x0004 TCNTH | R | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| | W | | | | | | | | |
| 0x0005 TCNTL | R | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| | W | | | | | | | | |
| 0x0006 TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 TTOV | R | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | W | | | | | | | | |
| 0x0008 TCTL1 | R | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| | W | | | | | | | | |
| 0x0009 TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | W | | | | | | | | |
| 0x000A TCTL3 | R | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| | W | | | | | | | | |
| 0x000B TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | W | | | | | | | | |
| 0x000C TIE | R | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| | W | | | | | | | | |
| 0x000D TSCR2 | R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| | W | | | | | | | | |
| 0x000E TFLG1 | R | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| | W | | | | | | | | |
| 0x000F TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0010–0x001F TCxH–TCxL[1] | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0020 PACTL | R | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| | W | | | | | | | | |

**Figure 8-5. TIM16B8CV3 Register Summary (Sheet 1 of 2)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0021 PAFLG | R | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| | W | | | | | | | | |
| 0x0022 PACNTH | R | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |
| | W | | | | | | | | |
| 0x0023 PACNTL | R | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| | W | | | | | | | | |
| 0x0024–0x002B Reserved | R | | | | | | | | |
| | W | | | | | | | | |
| 0x002C OCPD | R | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| | W | | | | | | | | |
| 0x002D Reserved | R | | | | | | | | |
| 0x002E PTPSR | R | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| | W | | | | | | | | |
| 0x002F Reserved | R | | | | | | | | |
| | W | | | | | | | | |

**Figure 8-5. TIM16B8CV3 Register Summary (Sheet 2 of 2)**

1 The register is available only if corresponding channel exists.

### 8.3.2.1 Timer Input Capture/Output Compare Select (TIOS)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-6. Timer Input Capture/Output Compare Select (TIOS)**

Read: Anytime

Write: Anytime

**Table 8-2. TIOS Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero.

| Field | Description |
|---|---|
| 7:0<br>IOS[7:0] | **Input Capture or Output Compare Channel Configuration**<br>0 The corresponding implemented channel acts as an input capture.<br>1 The corresponding implemented channel acts as an output compare. |

### 8.3.2.2 Timer Compare Force Register (CFORC)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-7. Timer Compare Force Register (CFORC)**

Read: Anytime but will always return 0x0000 (1 state is transient)

Write: Anytime

**Table 8-3. CFORC Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero.

| Field | Description |
|---|---|
| 7:0<br>FOC[7:0] | **Note: Force Output Compare Action for Channel 7:0** — A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “x” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set. A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides any channel 6:0 compares. If forced output compare on any channel occurs at the same time as the successful output compare then forced output compare action will take precedence and interrupt flag won’t get set. |

### 8.3.2.3 Output Compare 7 Mask Register (OC7M)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-8. Output Compare 7 Mask Register (OC7M)**

Read: Anytime

Write: Anytime

**Table 8-4. OC7M Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OC7M[7:0] | **Output Compare 7 Mask** — A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides any channel 6:0 compares. For each OC7M bit that is set, the output compare action reflects the corresponding OC7D bit.<br>0 The corresponding OC7Dx bit in the output compare 7 data register will not be transferred to the timer port on a channel 7 event, even if the corresponding pin is setup for output compare.<br>1 The corresponding OC7Dx bit in the output compare 7 data register will be transferred to the timer port on a channel 7 event.<br>**Note:** The corresponding channel must also be setup for output compare (IOSx = 1 and OCPDx = 0) for data to be transferred from the output compare 7 data register to the timer port. |

## 8.3.2.4 Output Compare 7 Data Register (OC7D)

[1] .

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-9. Output Compare 7 Data Register (OC7D)**

Read: Anytime

Write: Anytime

**Table 8-5. OC7D Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>OC7D[7:0] | **Output Compare 7 Data** — A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, can cause bits in the output compare 7 data register to transfer to the timer port data register depending on the output compare 7 mask register. |

## 8.3.2.5 Timer Count Register (TCNT)

Module Base + 0x0004

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 9 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-10. Timer Count Register High (TCNTH)**

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-11. Timer Count Register Low (TCNTL)**

The 16-bit main timer is an up counter.

A full access for the counter register should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

Read: Anytime

Write: Has no meaning or effect in the normal mode; only writable in special modes .

The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

### 8.3.2.6 Timer System Control Register 1 (TSCR1)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-12. Timer System Control Register 1 (TSCR1)**

Read: Anytime

Write: Anytime

**Table 8-6. TSCR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TEN | **Timer Enable**<br>0 Disables the main timer, including the counter. Can be used for reducing power consumption.<br>1 Allows the timer to function normally.<br>If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator because the ÷64 is generated by the timer prescaler. |
| 6<br>TSWAI | **Timer Module Stops While in Wait**<br>0 Allows the timer module to continue running during wait.<br>1 Disables the timer module when the MCU is in the wait mode. Timer interrupts cannot be used to get the MCU out of wait.<br>TSWAI also affects pulse accumulator. |

**Table 8-6. TSCR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>TSFRZ | **Timer Stops While in Freeze Mode**<br>0 Allows the timer counter to continue running while in freeze mode.<br>1 Disables the timer counter whenever the MCU is in freeze mode. This is useful for emulation.<br>TSFRZ does not stop the pulse accumulator. |
| 4<br>TFFCA | **Timer Fast Flag Clear All**<br>0 Allows the timer flag clearing to function normally.<br>1 For TFLG1(0x000E), a read from an input capture or a write to the output compare channel (0x0010–0x001F) causes the corresponding channel flag, CnF, to be cleared. For TFLG2 (0x000F), any access to the TCNT register (0x0004, 0x0005) clears the TOF flag. Any access to the PACNT registers (0x0022, 0x0023) clears the PAOVF and PAIF flags in the PAFLG register (0x0021) if channel 7 exists. This has the advantage of eliminating software overhead in a separate clear sequence. Extra care is required to avoid accidental flag clearing due to unintended accesses. |
| 3<br>PRNT | **Precision Timer**<br>0 Enables legacy timer. PR0, PR1, and PR2 bits of the TSCR2 register are used for timer counter prescaler selection.<br>1 Enables precision timer. All bits of the PTPSR register are used for Precision Timer Prescaler Selection, and all bits.<br>This bit is writable only once out of reset. |

## 8.3.2.7 Timer Toggle On Overflow Register 1 (TTOV)

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-13. Timer Toggle On Overflow Register 1 (TTOV)**

Read: Anytime

Write: Anytime

**Table 8-7. TTOV Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero.

| Field | Description |
|---|---|
| 7:0<br>TOV[7:0] | **Toggle On Overflow Bits** — TOVx toggles output compare pin on overflow. This feature only takes effect when in output compare mode. When set, it takes precedence over forced output compare but not channel 7 override events.<br>0 Toggle output compare pin on overflow feature disabled.<br>1 Toggle output compare pin on overflow feature enabled. |

### 8.3.2.8 Timer Control Register 1/Timer Control Register 2 (TCTL1/TCTL2)

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-14. Timer Control Register 1 (TCTL1)**

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-15. Timer Control Register 2 (TCTL2)**

Read: Anytime

Write: Anytime

**Table 8-8. TCTL1/TCTL2 Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero

| Field | Description |
|---|---|
| 7:0<br>OMx | **Output Mode** — These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx.<br>**Note:** To enable output action by OMx bits on timer port, the corresponding bit in OC7M should be cleared. For an output line to be driven by an OCx the OCPDx must be cleared. |
| 7:0<br>OLx | **Output Level** — These eightpairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx.<br>**Note:** To enable output action by OLx bits on timer port, the corresponding bit in OC7M should be cleared. For an output line to be driven by an OCx the OCPDx must be cleared. |

**Table 8-9. Compare Result Output Action**

| OMx | OLx | Action |
|---|---|---|
| 0 | 0 | No output compare<br>action on the timer output signal |
| 0 | 1 | Toggle OCx output line |
| 1 | 0 | Clear OCx output line to zero |
| 1 | 1 | Set OCx output line to one |

Note: To enable output action using the OM7 and OL7 bits on the timer port,the corresponding bit OC7M7 in the OC7M register must also be cleared. The settings for these bits can be seen inTable 8-10.

**Table 8-10. The OC7 and OCx event priority**

| OC7M7=0 | | | | OC7M7=1 | | | |
|---|---|---|---|---|---|---|---|
| OC7Mx=1 | | OC7Mx=0 | | OC7Mx=1 | | OC7Mx=0 | |
| TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx | TC7=TCx | TC7>TCx |
| IOCx=OC7Dx IOC7=OM7/OL7 | IOCx=OC7Dx +OMx/OLx IOC7=OM7/OL7 | IOCx=OMx/OLx IOC7=OM7/OL7 | | IOCx=OC7Dx IOC7=OC7D7 | IOCx=OC7Dx +OMx/OLx IOC7=OC7D7 | IOCx=OMx/OLx IOC7=OC7D7 | |

Note: in Table 8-10, the IOS7 and IOSx should be set to 1

IOSx is the register TIOS bit x,

OC7Mx is the register OC7M bit x,

TCx is timer Input Capture/Output Compare register,

IOCx is channel x,

OMx/OLx is the register TCTL1/TCTL2,

OC7Dx is the register OC7D bit x.

IOCx = OC7Dx+ OMx/OLx, means that both OC7 event and OCx event will change channel x value.

### 8.3.2.9 Timer Control Register 3/Timer Control Register 4 (TCTL3 and TCTL4)

Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-16. Timer Control Register 3 (TCTL3)**

Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-17. Timer Control Register 4 (TCTL4)**

Read: Anytime

Write: Anytime.

**Table 8-11. TCTL3/TCTL4 Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero.

| Field | Description |
|---|---|
| 7:0<br>EDGnB<br>EDGnA | **Input Capture Edge Control** — These eight pairs of control bits configure the input capture edge detector circuits. |

**Table 8-12. Edge Detector Circuit Configuration**

| EDGnB | EDGnA | Configuration |
|---|---|---|
| 0 | 0 | Capture disabled |
| 0 | 1 | Capture on rising edges only |
| 1 | 0 | Capture on falling edges only |
| 1 | 1 | Capture on any edge (rising or falling) |

## 8.3.2.10 Timer Interrupt Enable Register (TIE)

Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-18. Timer Interrupt Enable Register (TIE)**

Read: Anytime

Write: Anytime.

**Table 8-13. TIE Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero

| Field | Description |
|---|---|
| 7:0<br>C7I:C0I | **Input Capture/Output Compare "x" Interrupt Enable —** The bits in TIE correspond bit-for-bit with the bits in the TFLG1 status register. If cleared, the corresponding flag is disabled from causing a hardware interrupt. If set, the corresponding flag is enabled to cause a interrupt. |

### 8.3.2.11 Timer System Control Register 2 (TSCR2)

Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-19. Timer System Control Register 2 (TSCR2)**

Read: Anytime

Write: Anytime.

**Table 8-14. TSCR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOI | **Timer Overflow Interrupt Enable**<br>0 Interrupt inhibited.<br>1 Hardware interrupt requested when TOF flag set. |
| 3<br>TCRE | **Timer Counter Reset Enable** — This bit allows the timer counter to be reset by a successful output compare 7 event. This mode of operation is similar to an up-counting modulus counter.<br>0 Counter reset inhibited and counter free runs.<br>1 Counter reset by a successful output compare 7.<br>**Note:** If TC7 = 0x0000 and TCRE = 1, TCNT will stay at 0x0000 continuously. If TC7 = 0xFFFF and TCRE = 1, TOF will never be set when TCNT is reset from 0xFFFF to 0x0000.<br>**Note:** TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler counter width" + "1 Bus Clock", for a more detail explanation please refer to Section 8.4.3, “Output Compare<br>**Note:** This bit and feature is available only when channel 7 exists. If channel 7 doesn’t exist, this bit is reserved. Writing to reserved bit has no effect. Read from reserved bit return a zero. |
| 2:0<br>PR[2:0] | **Timer Prescaler Select** — These three bits select the frequency of the timer prescaler clock derived from the Bus Clock as shown in Table 8-15. |

**Table 8-15. Timer Clock Selection**

| PR2 | PR1 | PR0 | Timer Clock |
|---|---|---|---|
| 0 | 0 | 0 | Bus Clock / 1 |
| 0 | 0 | 1 | Bus Clock / 2 |
| 0 | 1 | 0 | Bus Clock / 4 |
| 0 | 1 | 1 | Bus Clock / 8 |
| 1 | 0 | 0 | Bus Clock / 16 |
| 1 | 0 | 1 | Bus Clock / 32 |
| 1 | 1 | 0 | Bus Clock / 64 |
| 1 | 1 | 1 | Bus Clock / 128 |

**NOTE**

The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero.

### 8.3.2.12 Main Timer Interrupt Flag 1 (TFLG1)

Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-20. Main Timer Interrupt Flag 1 (TFLG1)**

Read: Anytime

Write: Used in the clearing mechanism (set bits cause corresponding bits to be cleared). Writing a zero will not affect current status of the bit.

**Table 8-16. TRLG1 Field Descriptions**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero.

| Field | Description |
|---|---|
| 7:0<br>C[7:0]F | **Input Capture/Output Compare Channel "x" Flag** — These flags are set when an input capture or output compare event occurs. Clearing requires writing a one to the corresponding flag bit while TEN or PAEN is set to one.<br><br>**Note:** When TFFCA bit in TSCR register is set, a read from an input capture or a write into an output compare channel (0x0010–0x001F) will cause the corresponding channel flag CxF to be cleared. |

### 8.3.2.13 Main Timer Interrupt Flag 2 (TFLG2)

Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 8-21. Main Timer Interrupt Flag 2 (TFLG2)**

TFLG2 indicates when interrupt conditions have occurred. To clear a bit in the flag register, write the bit to one while TEN bit of TSCR1 or PAEN bit of PACTL is set to one.

Read: Anytime

Write: Used in clearing mechanism (set bits cause corresponding bits to be cleared).

Any access to TCNT will clear TFLG2 register if the TFFCA bit in TSCR register is set.

**Table 8-17. TRLG2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — Set when 16-bit free-running timer overflows from 0xFFFF to 0x0000. Clearing this bit requires writing a one to bit 7 of TFLG2 register while the TEN bit of TSCR1 or PAEN bit of PACTL is set to one (See also TCRE control bit explanation) . |

### 8.3.2.14 Timer Input Capture/Output Compare Registers High and Low 0–7(TCxH and TCxL)

Module Base + 0x0010 = TC0H 0x0018=TC4H
0x0012 = TC1H 0x001A=TC5H
0x0014=TC2H 0x001C=TC6H
0x0016=TC3H 0x001E=TC7H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-22. Timer Input Capture/Output Compare Register x High (TCxH)**

Module Base + 0x0011 = TC0L 0x0019 =TC4L
0x0013 = TC1L 0x001B=TC5L
0x0015 =TC2L 0x001D=TC6L
0x0017=TC3L 0x001F=TC7L

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-23. Timer Input Capture/Output Compare Register x Low (TCxL)**

[1] This register is available only when the corresponding channel exists and is reserved if that channel does not exist. Writes to a reserved register have no functional effect. Reads from a reserved register return zeroes.

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

Read: Anytime

Write: Anytime for output compare function.Writes to these registers have no meaning or effect during input capture. All timer input capture/output compare registers are reset to 0x0000.

**NOTE**

Read/Write access in byte mode for high byte should take place before low byte otherwise it will give a different result.

### 8.3.2.15 16-Bit Pulse Accumulator Control Register (PACTL)

Module Base + 0x0020

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 8-24. 16-Bit Pulse Accumulator Control Register (PACTL)**

Read: Any time

Write: Any time

When PAEN is set, the Pulse Accumulator counter is enabled. The Pulse Accumulator counter shares the input pin with IOC7.

**Table 8-18. PACTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>PAEN | **Pulse Accumulator System Enable** — PAEN is independent from TEN. With timer disabled, the pulse accumulator can function unless pulse accumulator is disabled.<br>0 16-Bit Pulse Accumulator system disabled.<br>1 Pulse Accumulator system enabled. |
| 5<br>PAMOD | **Pulse Accumulator Mode** — This bit is active only when the Pulse Accumulator is enabled (PAEN = 1). See Table 8-19.<br>0 Event counter mode.<br>1 Gated time accumulation mode. |
| 4<br>PEDGE | **Pulse Accumulator Edge Control** — This bit is active only when the Pulse Accumulator is enabled (PAEN = 1). For PAMOD bit = 0 (event counter mode). See Table 8-19.<br>0 Falling edges on IOC7 pin cause the count to be increased.<br>1 Rising edges on IOC7 pin cause the count to be increased.<br>For PAMOD bit = 1 (gated time accumulation mode).<br>0 IOC7 input pin high enables M (Bus clock) divided by 64 clock to Pulse Accumulator and the trailing falling edge on IOC7 sets the PAIF flag.<br>1 IOC7 input pin low enables M (Bus clock) divided by 64 clock to Pulse Accumulator and the trailing rising edge on IOC7 sets the PAIF flag. |
| 3:2<br>CLK[1:0] | **Clock Select Bits —** Refer to Table 8-20. |
| 1<br>PAOVI | **Pulse Accumulator Overflow Interrupt Enable**<br>0 Interrupt inhibited.<br>1 Interrupt requested if PAOVF is set. |
| 0<br>PAI | **Pulse Accumulator Input Interrupt Enable**<br>0 Interrupt inhibited.<br>1 Interrupt requested if PAIF is set. |

**Table 8-19. Pin Action**

| PAMOD | PEDGE | Pin Action |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Div. by 64 clock enabled with pin high level |
| 1 | 1 | Div. by 64 clock enabled with pin low level |

**NOTE**

If the timer is not active (TEN = 0 in TSCR), there is no divide-by-64 because the ÷64 clock is generated by the timer prescaler.

**Table 8-20. Timer Clock Selection**

| CLK1 | CLK0 | Timer Clock |
|---|---|---|
| 0 | 0 | Use timer prescaler clock as timer counter clock |
| 0 | 1 | Use PACLK as input to timer counter clock |
| 1 | 0 | Use PACLK/256 as timer counter clock frequency |
| 1 | 1 | Use PACLK/65536 as timer counter clock frequency |

For the description of PACLK please refer Figure 8-30**.**

If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written.

### 8.3.2.16 Pulse Accumulator Flag Register (PAFLG)

[1] .

Module Base + 0x0021

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 8-25. Pulse Accumulator Flag Register (PAFLG)**

Read: Anytime

Write: Anytime

When the TFFCA bit in the TSCR register is set, any access to the PACNT register will clear all the flags in the PAFLG register. Timer module or Pulse Accumulator must stay enabled (TEN=1 or PAEN=1) while clearing these bits.

**Table 8-21. PAFLG Field Descriptions**

| Field | Description |
|---|---|
| 1<br>PAOVF | **Pulse Accumulator Overflow Flag** — Set when the 16-bit pulse accumulator overflows from 0xFFFF to 0x0000.<br>Clearing this bit requires writing a one to this bit in the PAFLG register while TEN bit of TSCR1 or PAEN bit of PACTL register is set to one. |
| 0<br>PAIF | **Pulse Accumulator Input edge Flag** — Set when the selected edge is detected at the IOC7 input pin.In event mode the event edge triggers PAIF and in gated time accumulation mode the trailing edge of the gate signal at the IOC7 input pin triggers PAIF.<br>Clearing this bit requires writing a one to this bit in the PAFLG register while TEN bit of TSCR1 or PAEN bit of PACTL register is set to one. Any access to the PACNT register will clear all the flags in this register when TFFCA bit in register TSCR(0x0006) is set. |

## 8.3.2.17 Pulse Accumulators Count Registers (PACNT)

Module Base + 0x0022

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-26. Pulse Accumulator Count Register High (PACNTH)**

[1] .

Module Base + 0x0023

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-27. Pulse Accumulator Count Register Low (PACNTL)**

Read: Anytime

Write: Anytime

These registers contain the number of active input edges on its input pin since the last reset.

When PACNT overflows from 0xFFFF to 0x0000, the Interrupt flag PAOVF in PAFLG (0x0021) is set.

Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

NOTE

Reading the pulse accumulator counter registers immediately after an active edge on the pulse accumulator input pin may miss the last count because the input has to be synchronized with the Bus clock first.

### 8.3.2.18 Output Compare Pin Disconnect Register(OCPD)

Module Base + 0x002C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-28. Output Compare Pin Disconnect Register (OCPD)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 8-22. OCPD Field Description**

**Note:** Writing to unavailable bits has no effect. Reading from unavailable bits return a zero.

| Field | Description |
|---|---|
| 7:0<br>OCPD[7:0] | **Output Compare Pin Disconnect Bits**<br>0 Enables the timer channel port. Output Compare action will occur on the channel pin. These bits do not affect the input capture or pulse accumulator functions.<br>1 Disables the timer channel port. Output Compare action will not occur on the channel pin, but the output compare flag still become set. |

### 8.3.2.19 Precision Timer Prescaler Select Register (PTPSR)

Module Base + 0x002E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-29. Precision Timer Prescaler Select Register (PTPSR)**

Read: Anytime

Write: Anytime

All bits reset to zero.

**Table 8-23. PTPSR Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTPS[7:0] | **Precision Timer Prescaler Select Bits** — These eight bits specify the division rate of the main Timer prescaler. These are effective only when the PRNT bit of TSCR1 is set to 1. Table 8-24 shows some selection examples in this case.<br>The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal zero. |

The Prescaler can be calculated as follows depending on logical value of the PTPS[7:0] and PRNT bit:

PRNT = 1 : Prescaler = PTPS[7:0] + 1

**Table 8-24. Precision Timer Prescaler Selection Examples when PRNT = 1**

| PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 | Prescale Factor |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 20 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 21 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 22 |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | - | - |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 253 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 254 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 255 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 256 |

## 8.4 Functional Description

This section provides a complete functional description of the timer TIM16B8CV3 block. Please refer to the detailed timer block diagram in Figure 8-30 as necessary.

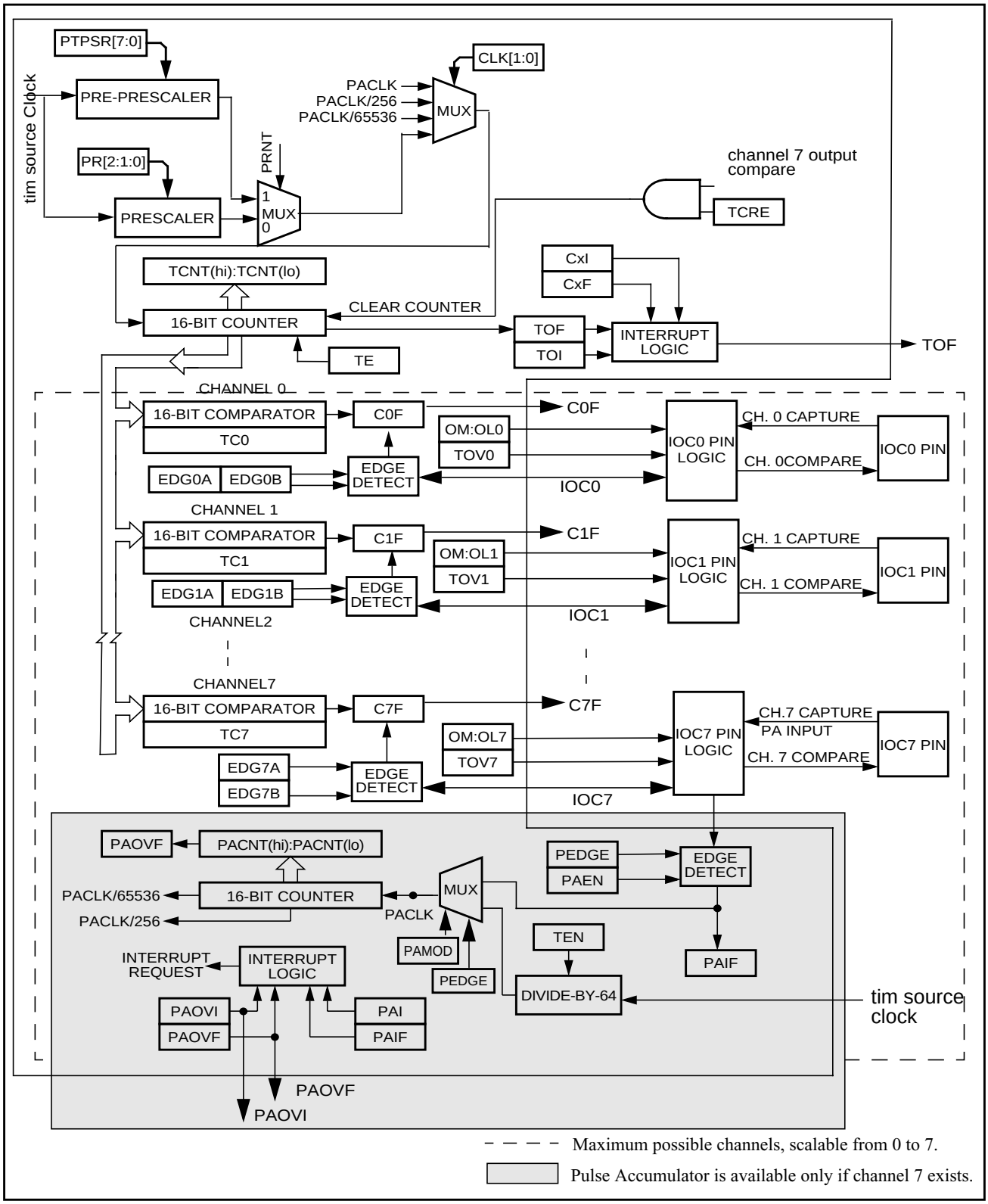



**Figure 8-30. Detailed Timer Block Diagram**

### 8.4.1 Prescaler

The prescaler divides the Bus clock by 1, 2, 4, 8, 16, 32, 64 or 128. The prescaler select bits, PR[2:0], select the prescaler divisor. PR[2:0] are in timer system control register 2 (TSCR2).

The prescaler divides the Bus clock by a prescalar value. Prescaler select bits PR[2:0] of in timer system control register 2 (TSCR2) are set to define a prescalar value that generates a divide by 1, 2, 4, 8, 16, 32, 64 and 128 when the PRNT bit in TSCR1 is disabled.

By enabling the PRNT bit of the TSCR1 register, the performance of the timer can be enhanced. In this case, it is possible to set additional prescaler settings for the main timer counter in the present timer by using PTPSR[7:0] bits of PTPSR register generating divide by 1, 2, 3, 4,....20, 21, 22, 23,.....255, or 256.

### 8.4.2 Input Capture

Clearing the I/O (input/output) select bit, IOSx, configures channel x as an input capture channel. The input capture function captures the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the timer transfers the value in the timer counter into the timer channel registers, TCx.

The minimum pulse width for the input capture input is greater than two Bus clocks.

An input capture on channel x sets the CxF flag. The CxI bit enables the CxF flag to generate interrupt requests. Timer module or Pulse Accumulator must stay enabled (TEN bit of TSCR1 or PAEN bit of PACTL register must be set to one) while clearing CxF (writing one to CxF).

### 8.4.3 Output Compare

Setting the I/O select bit, IOSx, configures channel x when available as an output compare channel. The output compare function can generate a periodic pulse with a programmable polarity, duration, and frequency. When the timer counter reaches the value in the channel registers of an output compare channel, the timer can set, clear, or toggle the channel pin if the corresponding OCPDx bit is set to zero. An output compare on channel x sets the CxF flag. The CxI bit enables the CxF flag to generate interrupt requests. Timer module or Pulse Accumulator must stay enabled (TEN bit of TSCR1 or PAEN bit of PACTL register must be set to one) while clearing CxF (writing one to CxF).

The output mode and level bits, OMx and OLx, select set, clear, toggle on output compare. Clearing both OMx and OLx results in no output compare action on the output compare channel pin.

Setting a force output compare bit, FOCx, causes an output compare on channel x. A forced output compare does not set the channel flag.

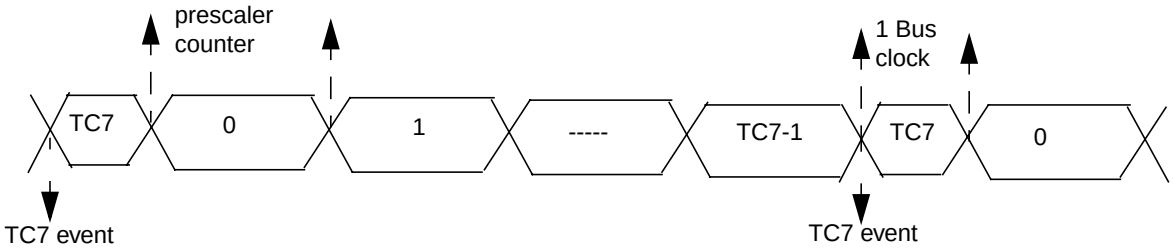
A channel 7 event, which can be a counter overflow when TTOV[7] is set or a successful output compare on channel 7, overrides output compares on all other output compare channels. The output compare 7 mask register masks the bits in the output compare 7 data register. The timer counter reset enable bit, TCRE, enables channel 7 output compares to reset the timer counter. A channel 7 output compare can reset the timer counter even if the IOC7 pin is being used as the pulse accumulator input.

Writing to the timer port bit of an output compare pin does not affect the pin state. The value written is stored in an internal latch. When the pin becomes available for general-purpose output, the last value written to the bit appears at the pin.

When TCRE is set and TC7 is not equal to 0, then TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it will last only one Bus cycle then reset to 0.

Note: in Figure 8-31,if PR[2:0] is equal to 0, one prescaler counter equal to one Bus clock

**Figure 8-31. The TCNT cycle diagram under TCRE=1 condition**




### 8.4.3.1 OC Channel Initialization

The internal register whose output drives OCx can be programmed before the timer drives OCx. The desired state can be programmed to this internal register by writing a one to CFORCx bit with TIOSx, OCPDx and TEN bits set to one.

Set OCx: Write a 1 to FOCx while TEN=1, IOSx=1, OMx=1, OLx=1 and OCPDx=1

Clear OCx: Write a 1 to FOCx while TEN=1, IOSx=1, OMx=1, OLx=0 and OCPDx=1

Setting OCPDx to zero allows the internal register to drive the programmed state to OCx. This allows a glitch free switch over of port from general purpose I/O to timer output once the OCPDx bit is set to zero.

## 8.4.4 Pulse Accumulator

The pulse accumulator (PACNT) is a 16-bit counter that can operate in two modes:

Event counter mode — Counting edges of selected polarity on the pulse accumulator input pin, PAI.

Gated time accumulation mode — Counting pulses from a divide-by-64 clock. The PAMOD bit selects the mode of operation.

The minimum pulse width for the PAI input is greater than two Bus clocks.

### 8.4.5 Event Counter Mode

Clearing the PAMOD bit configures the PACNT for event counter operation. An active edge on the IOC7 pin increments the pulse accumulator counter. The PEDGE bit selects falling edges or rising edges to increment the count.

**NOTE**

The PACNT input and timer channel 7 use the same pin IOC7. To use the IOC7, disconnect it from the output logic by clearing the channel 7 output mode and output level bits, OM7 and OL7. Also clear the channel 7 output compare 7 mask bit, OC7M7.

The Pulse Accumulator counter register reflect the number of active input edges on the PACNT input pin since the last reset.

The PAOVF bit is set when the accumulator rolls over from 0xFFFF to 0x0000. The pulse accumulator overflow interrupt enable bit, PAOVI, enables the PAOVF flag to generate interrupt requests.

**NOTE**

The pulse accumulator counter can operate in event counter mode even when the timer enable bit, TEN, is clear.

### 8.4.6 Gated Time Accumulation Mode

Setting the PAMOD bit configures the pulse accumulator for gated time accumulation operation. An active level on the PACNT input pin enables a divided-by-64 clock to drive the pulse accumulator. The PEDGE bit selects low levels or high levels to enable the divided-by-64 clock.

The trailing edge of the active level at the IOC7 pin sets the PAIF. The PAI bit enables the PAIF flag to generate interrupt requests.

The pulse accumulator counter register reflect the number of pulses from the divided-by-64 clock since the last reset.

**NOTE**

The timer prescaler generates the divided-by-64 clock. If the timer is not active, there is no divided-by-64 clock.

## 8.5 Resets

The reset state of each individual bit is listed within Section 8.3, “Memory Map and Register Definition” which details the registers and their bit fields

## 8.6 Interrupts

This section describes interrupts originated by the TIM16B8CV3 block. Table 8-25 lists the interrupts generated by the TIM16B8CV3 to communicate with the MCU.

**Table 8-25.** TIM16B8CV3 **Interrupts**

| Interrupt | Offset | Vector | Priority | Source | Description |
|---|---|---|---|---|---|
| C[7:0]F | — | — | — | Timer Channel 7–0 | Active high timer channel interrupts 7–0 |
| PAOVI | — | — | — | Pulse Accumulator Input | Active high pulse accumulator input interrupt |
| PAOVF | — | — | — | Pulse Accumulator Overflow | Pulse accumulator overflow interrupt |
| TOF | — | — | — | Timer Overflow | Timer Overflow interrupt |

The TIM16B8CV3 could use up to 11 interrupt vectors. The interrupt vector offsets and interrupt numbers are chip dependent.

## 8.6.1 Channel [7:0] Interrupt (C[7:0]F)

This active high outputs will be asserted by the module to request a timer channel 7 – 0 interrupt. The TIM block only generates the interrupt and does not service it. Only bits related to implemented channels are valid.

## 8.6.2 Pulse Accumulator Input Interrupt (PAOVI)

This active high output will be asserted by the module to request a timer pulse accumulator input interrupt. The TIM block only generates the interrupt and does not service it.

## 8.6.3 Pulse Accumulator Overflow Interrupt (PAOVF)

This active high output will be asserted by the module to request a timer pulse accumulator overflow interrupt. The TIM block only generates the interrupt and does not service it.

## 8.6.4 Timer Overflow Interrupt (TOF)

This active high output will be asserted by the module to request a timer overflow interrupt. The TIM block only generates the interrupt and does not service it.

# Chapter 9
# Pulse-Width Modulator (S12PWM8B8CV2)

**Table 9-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| v02.00 | Feb. 20, 2009 | All | Initial revision of scalable PWM. Started from pwm_8b8c (v01.08). |

## 9.1 Introduction

The Version 2 of S12 PWM module is a channel scalable and optimized implementation of S12 PWM8B8C Version 1. The channel is scalable in pairs from PWM0 to PWM7 and the available channel number is 2, 4, 6 and 8. The shutdown feature has been removed and the flexibility to select one of four clock sources per channel has improved. If the corresponding channels exist and shutdown feature is not used, the Version 2 is fully software compatible to Version 1.

### 9.1.1 Features

The scalable PWM block includes these distinctive features:

- Up to eight independent PWM channels, scalable in pairs (PWM0 to PWM7)
- Available channel number could be 2, 4, 6, 8 (refer to device specification for exact number)
- Programmable period and duty cycle for each channel
- Dedicated counter for each PWM channel
- Programmable PWM enable/disable for each channel
- Software selection of PWM duty pulse polarity for each channel
- Period and duty cycle are double buffered. Change takes effect when the end of the effective period is reached (PWM counter reaches zero) or when the channel is disabled.
- Programmable center or left aligned outputs on individual channels
- Up to eight 8-bit channel or four 16-bit channel PWM resolution
- Four clock sources (A, B, SA, and SB) provide for a wide range of frequencies
- Programmable clock select logic

### 9.1.2 Modes of Operation

There is a software programmable option for low power consumption in wait mode that disables the input clock to the prescaler.

In freeze mode there is a software programmable option to disable the input clock to the prescaler. This is useful for emulation.

Wait: The prescaler keeps on running, unless PSWAI in PWMCTL is set to 1.

Freeze: The prescaler keeps on running, unless PFRZ in PWMCTL is set to 1.

### 9.1.3 Block Diagram

Figure 9-1 shows the block diagram for the 8-bit up to 8-channel scalable PWM block.

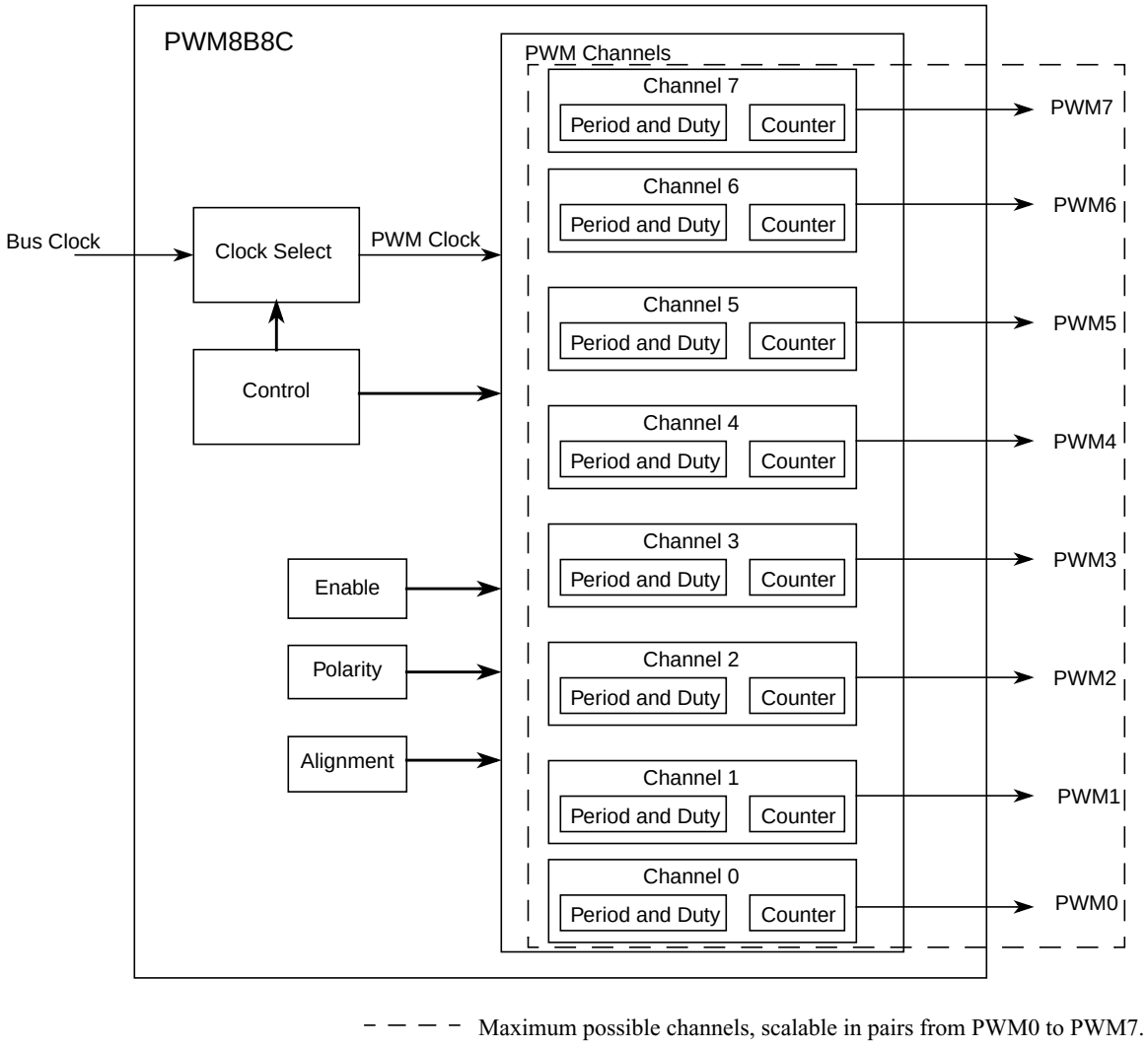



**Figure 9-1. Scalable PWM Block Diagram**

## 9.2 External Signal Description

The scalable PWM module has a selected number of external pins. Refer to device specification for exact number.

### 9.2.1 PWM7 - PWM0 — PWM Channel 7 - 0

Those pins serve as waveform output of PWM channel 7 - 0.

## 9.3 Memory Map and Register Definition

### 9.3.1 Module Memory Map

This section describes the content of the registers in the scalable PWM module. The base address of the scalable PWM module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset. The figure below shows the registers associated with the scalable PWM and their relative offset from the base address. The register detail description follows the order they appear in the register map.

Reserved bits within a register will always read as 0 and the write will be unimplemented. Unimplemented functions are indicated by shading the bit.

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

### 9.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the scalable PWM module.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 PWME[1] | R | PWME7 | PWME6 | PWME5 | PWME4 | PWME3 | PWME2 | PWME1 | PWME0 |
| | W | | | | | | | | |
| 0x0001 PWMPOL[1] | R | PPOL7 | PPOL6 | PPOL5 | PPOL4 | PPOL3 | PPOL2 | PPOL1 | PPOL0 |
| | W | | | | | | | | |
| 0x0002 PWMCLK[1] | R | PCLK7 | PCLKL6 | PCLK5 | PCLK4 | PCLK3 | PCLK2 | PCLK1 | PCLK0 |
| | W | | | | | | | | |
| 0x0003 PWMPRCLK | R | 0 | PCKB2 | PCKB1 | PCKB0 | 0 | PCKA2 | PCKA1 | PCKA0 |
| | W | | | | | | | | |
| 0x0004 PWMCAE[1] | R | CAE7 | CAE6 | CAE5 | CAE4 | CAE3 | CAE2 | CAE1 | CAE0 |
| | W | | | | | | | | |
| 0x0005 PWMCTL[1] | R | CON67 | CON45 | CON23 | CON01 | PSWAI | PFRZ | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 9-2. The scalable PWM Register Summary (Sheet 1 of 4)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 PWMCLKAB [1] | R | PCLKAB7 | PCLKAB6 | PCLKAB5 | PCLKAB4 | PCLKAB3 | PCLKAB2 | PCLKAB1 | PCLKAB0 |
| | W | | | | | | | | |
| 0x0007 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0008 PWMSCLA | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0009 PWMSCLB | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x000A RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000B RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000C PWMCNT0 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000D PWMCNT1 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000E PWMCNT2 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000F PWMCNT3 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0010 PWMCNT4 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0011 PWMCNT5 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0012 PWMCNT6 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0013 PWMCNT7 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0014 PWMPER0 [2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 9-2. The scalable PWM Register Summary (Sheet 2 of 4)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0015 PWMPER1[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0016 PWMPER2[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0017 PWMPER3[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0018 PWMPER4[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0019 PWMPER5[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001A PWMPER6[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001B PWMPER7[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001C PWMDTY0[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001D PWMDTY1[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001E PWMDTY2[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x001F PWMDTY3[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0010 PWMDTY4[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0021 PWMDTY5[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0022 PWMDTY6[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0023 PWMDTY7[2] | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 9-2. The scalable PWM Register Summary (Sheet 3 of 4)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0024 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0025 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0026 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0027 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 9-2. The scalable PWM Register Summary (Sheet 4 of 4)**

1 The related bit is available only if corresponding channel exists.

2 The register is available only if corresponding channel exists.

### 9.3.2.1 PWM Enable Register (PWME)

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source.

**NOTE**

The first PWM cycle after enabling the channel can be irregular.

An exception to this is when channels are concatenated. Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit. In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output lines are disabled.

While in run mode, if all existing PWM channels are disabled (PWMEx–0 = 0), the prescaler counter shuts off for power savings.

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PWME7 | PWME6 | PWME5 | PWME4 | PWME3 | PWME2 | PWME1 | PWME0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-3. PWM Enable Register (PWME)**

Read: Anytime

Write: Anytime

**Table 9-2. PWME Field Descriptions**

**Note:** Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

| Field | Description |
|---|---|
| 7<br>PWME7 | **Pulse Width Channel 7 Enable**<br>0 Pulse width channel 7 is disabled.<br>1 Pulse width channel 7 is enabled. The pulse modulated signal becomes available at PWM output bit 7 when its clock source begins its next cycle. |
| 6<br>PWME6 | **Pulse Width Channel 6 Enable**<br>0 Pulse width channel 6 is disabled.<br>1 Pulse width channel 6 is enabled. The pulse modulated signal becomes available at PWM output bit 6 when its clock source begins its next cycle. If CON67=1, then bit has no effect and PWM output line 6 is disabled. |
| 5<br>PWME5 | **Pulse Width Channel 5 Enable**<br>0 Pulse width channel 5 is disabled.<br>1 Pulse width channel 5 is enabled. The pulse modulated signal becomes available at PWM output bit 5 when its clock source begins its next cycle. |
| 4<br>PWME4 | **Pulse Width Channel 4 Enable**<br>0 Pulse width channel 4 is disabled.<br>1 Pulse width channel 4 is enabled. The pulse modulated signal becomes available at PWM, output bit 4 when its clock source begins its next cycle. If CON45 = 1, then bit has no effect and PWM output line 4 is disabled. |
| 3<br>PWME3 | **Pulse Width Channel 3 Enable**<br>0 Pulse width channel 3 is disabled.<br>1 Pulse width channel 3 is enabled. The pulse modulated signal becomes available at PWM, output bit 3 when its clock source begins its next cycle. |
| 2<br>PWME2 | **Pulse Width Channel 2 Enable**<br>0 Pulse width channel 2 is disabled.<br>1 Pulse width channel 2 is enabled. The pulse modulated signal becomes available at PWM, output bit 2 when its clock source begins its next cycle. If CON23 = 1, then bit has no effect and PWM output line 2 is disabled. |
| 1<br>PWME1 | **Pulse Width Channel 1 Enable**<br>0 Pulse width channel 1 is disabled.<br>1 Pulse width channel 1 is enabled. The pulse modulated signal becomes available at PWM, output bit 1 when its clock source begins its next cycle. |
| 0<br>PWME0 | **Pulse Width Channel 0 Enable**<br>0 Pulse width channel 0 is disabled.<br>1 Pulse width channel 0 is enabled. The pulse modulated signal becomes available at PWM, output bit 0 when its clock source begins its next cycle. If CON01 = 1, then bit has no effect and PWM output line 0 is disabled. |

### 9.3.2.2 PWM Polarity Register (PWMPOL)

The starting polarity of each PWM channel waveform is determined by the associated PPOLx bit in the PWMPOL register. If the polarity bit is one, the PWM channel output is high at the beginning of the cycle and then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PPOL7 | PPOL6 | PPOL5 | PPOL4 | PPOL3 | PPOL2 | PPOL1 | PPOL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-4. PWM Polarity Register (PWMPOL)**

Read: Anytime

Write: Anytime

**NOTE**

PPOLx register bits can be written anytime. If the polarity is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition

**Table 9-3. PWMPOL Field Descriptions**

**Note:** Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

| Field | Description |
|---|---|
| 7–0<br>PPOL[7:0] | P**ulse Width Channel 7–0 Polarity Bits**<br>0 PWM channel 7–0 outputs are low at the beginning of the period, then go high when the duty count is reached.<br>1 PWM channel 7–0 outputs are high at the beginning of the period, then go low when the duty count is reached. |

### 9.3.2.3 PWM Clock Select Register (PWMCLK)

Each PWM channel has a choice of four clocks to use as the clock source for that channel as described below.

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PCLK7 | PCLKL6 | PCLK5 | PCLK4 | PCLK3 | PCLK2 | PCLK1 | PCLK0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-5. PWM Clock Select Register (PWMCLK)**

Read: Anytime

Write: Anytime

**NOTE**

Register bits PCLK0 to PCLK7 can be written anytime. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

**Table 9-4. PWMCLK Field Descriptions**

**Note:** Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

| Field | Description |
|---|---|
| 7-0<br>PCLK[7:0] | **Pulse Width Channel 7-0 Clock Select**<br>0 Clock A or B is the clock source for PWM channel 7-0, as shown in Table 9-5 and Table 9-6.<br>1 Clock SA or SB is the clock source for PWM channel 7-0, as shown in Table 9-5 and Table 9-6. |

The clock source of each PWM channel is determined by PCLKx bits in PWMCLK and PCLKABx bits in PWMCLKAB (see Section 9.3.2.7, “PWM Clock A/B Select Register (PWMCLKAB)). For Channel 0, 1, 4, 5, the selection is shown in Table 9-5; For Channel 2, 3, 6, 7, the selection is shown in Table 9-6.

**Table 9-5. PWM Channel 0, 1, 4, 5 Clock Source Selection**

| PCLKAB[0,1,4,5] | PCLK[0,1,4,5] | Clock Source Selection |
|---|---|---|
| 0 | 0 | Clock A |
| 0 | 1 | Clock SA |
| 1 | 0 | Clock B |
| 1 | 1 | Clock SB |

**Table 9-6. PWM Channel 2, 3, 6, 7 Clock Source Selection**

| PCLKAB[2,3,6,7] | PCLK[2,3,6,7] | Clock Source Selection |
|---|---|---|
| 0 | 0 | Clock B |
| 0 | 1 | Clock SB |
| 1 | 0 | Clock A |
| 1 | 1 | Clock SA |

### 9.3.2.4 PWM Prescale Clock Select Register (PWMPRCLK)

This register selects the prescale clock source for clocks A and B independently.

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PCKB2 | PCKB1 | PCKB0 | 0 | PCKA2 | PCKA1 | PCKA0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-6. PWM Prescale Clock Select Register (PWMPRCLK)**

Read: Anytime

Write: Anytime

**NOTE**

PCKB2–0 and PCKA2–0 register bits can be written anytime. If the clock pre-scale is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

**Table 9-7. PWMPRCLK Field Descriptions**

| Field | Description |
|---|---|
| 6–4<br>PCKB[2:0] | **Prescaler Select for Clock B** — Clock B is one of two clock sources which can be used for all channels. These three bits determine the rate of clock B, as shown in Table 9-8. |
| 2–0<br>PCKA[2:0] | **Prescaler Select for Clock A** — Clock A is one of two clock sources which can be used for all channels. These three bits determine the rate of clock A, as shown in Table 9-8. |

**Table 9-8. Clock A or Clock B Prescaler Selects**

| PCKA/B2 | PCKA/B1 | PCKA/B0 | Value of Clock A/B |
|---|---|---|---|
| 0 | 0 | 0 | Bus clock |
| 0 | 0 | 1 | Bus clock / 2 |
| 0 | 1 | 0 | Bus clock / 4 |
| 0 | 1 | 1 | Bus clock / 8 |
| 1 | 0 | 0 | Bus clock / 16 |
| 1 | 0 | 1 | Bus clock / 32 |
| 1 | 1 | 0 | Bus clock / 64 |
| 1 | 1 | 1 | Bus clock / 128 |

### 9.3.2.5 PWM Center Align Enable Register (PWMCAE)

The PWMCAE register contains eight control bits for the selection of center aligned outputs or left aligned outputs for each PWM channel. If the CAEx bit is set to a one, the corresponding PWM output will be center aligned. If the CAEx bit is cleared, the corresponding PWM output will be left aligned. See Section 9.4.2.5, “Left Aligned Outputs” and Section 9.4.2.6, “Center Aligned Outputs” for a more detailed description of the PWM output modes.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CAE7 | CAE6 | CAE5 | CAE4 | CAE3 | CAE2 | CAE1 | CAE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-7. PWM Center Align Enable Register (PWMCAE)**

Read: Anytime

Write: Anytime

**NOTE**

Write these bits only when the corresponding channel is disabled.

**Table 9-9. PWMCAE Field Descriptions**

**Note:** Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

| Field | Description |
|---|---|
| 7–0<br>CAE[7:0] | **Center Aligned Output Modes on Channels 7–0**<br>0 Channels 7–0 operate in left aligned output mode.<br>1 Channels 7–0 operate in center aligned output mode. |

### 9.3.2.6 PWM Control Register (PWMCTL)

The PWMCTL register provides for various control of the PWM module.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CON67 | CON45 | CON23 | CON01 | PSWAI | PFRZ | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-8. PWM Control Register (PWMCTL)**

Read: Anytime

Write: Anytime

There are up to four control bits for concatenation, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. If the corresponding channels do not exist on a particular derivative, then writes to these bits have no effect and reads will return zeroes. When channels 6 and 7are concatenated, channel 6 registers become the high order bytes of the double byte channel. When channels 4 and 5 are concatenated, channel 4 registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated, channel 2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

See Section 9.4.2.7, “PWM 16-Bit Functions” for a more detailed description of the concatenation PWM Function.

**NOTE**

Change these bits only when both corresponding channels are disabled.

**Table 9-10. PWMCTL Field Descriptions**

**Note:** Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

| Field | Description |
|---|---|
| 7<br>CON67 | **Concatenate Channels 6 and 7**<br>0 Channels 6 and 7 are separate 8-bit PWMs.<br>1 Channels 6 and 7 are concatenated to create one 16-bit PWM channel. Channel 6 becomes the high order byte and channel 7 becomes the low order byte. Channel 7 output pin is used as the output for this 16-bit PWM (bit 7 of port PWMP). Channel 7 clock select control-bit determines the clock source, channel 7 polarity bit determines the polarity, channel 7 enable bit enables the output and channel 7 center aligned enable bit determines the output mode. |
| 6<br>CON45 | **Concatenate Channels 4 and 5**<br>0 Channels 4 and 5 are separate 8-bit PWMs.<br>1 Channels 4 and 5 are concatenated to create one 16-bit PWM channel. Channel 4 becomes the high order byte and channel 5 becomes the low order byte. Channel 5 output pin is used as the output for this 16-bit PWM (bit 5 of port PWMP). Channel 5 clock select control-bit determines the clock source, channel 5 polarity bit determines the polarity, channel 5 enable bit enables the output and channel 5 center aligned enable bit determines the output mode. |
| 5<br>CON23 | **Concatenate Channels 2 and 3**<br>0 Channels 2 and 3 are separate 8-bit PWMs.<br>1 Channels 2 and 3 are concatenated to create one 16-bit PWM channel. Channel 2 becomes the high order byte and channel 3 becomes the low order byte. Channel 3 output pin is used as the output for this 16-bit PWM (bit 3 of port PWMP). Channel 3 clock select control-bit determines the clock source, channel 3 polarity bit determines the polarity, channel 3 enable bit enables the output and channel 3 center aligned enable bit determines the output mode. |
| 4<br>CON01 | **Concatenate Channels 0 and 1**<br>0 Channels 0 and 1 are separate 8-bit PWMs.<br>1 Channels 0 and 1 are concatenated to create one 16-bit PWM channel. Channel 0 becomes the high order byte and channel 1 becomes the low order byte. Channel 1 output pin is used as the output for this 16-bit PWM (bit 1 of port PWMP). Channel 1 clock select control-bit determines the clock source, channel 1 polarity bit determines the polarity, channel 1 enable bit enables the output and channel 1 center aligned enable bit determines the output mode. |
| 3<br>PSWAI | **PWM Stops in Wait Mode —** Enabling this bit allows for lower power consumption in wait mode by disabling the input clock to the prescaler.<br>0 Allow the clock to the prescaler to continue while in wait mode.<br>1 Stop the input clock to the prescaler whenever the MCU is in wait mode. |
| 2<br>PFRZ | **PWM Counters Stop in Freeze Mode** — In freeze mode, there is an option to disable the input clock to the prescaler by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode, the input clock to the prescaler is disabled. This feature is useful during emulation as it allows the PWM function to be suspended. In this way, the counters of the PWM can be stopped while in freeze mode so that once normal program flow is continued, the counters are re-enabled to simulate real-time operations. Since the registers can still be accessed in this mode, to re-enable the prescaler clock, either disable the PFRZ bit or exit freeze mode.<br>0 Allow PWM to continue while in freeze mode.<br>1 Disable PWM input clock to the prescaler whenever the part is in freeze mode. This is useful for emulation. |

### 9.3.2.7 PWM Clock A/B Select Register (PWMCLKAB)

Each PWM channel has a choice of four clocks to use as the clock source for that channel as described below.

Module Base + 0x00006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PCLKAB7 | PCLKAB6 | PCLKAB5 | PCLKAB4 | PCLKAB3 | PCLKAB2 | PCLKAB1 | PCLKAB0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-9. PWM Clock Select Register (PWMCLK)**

Read: Anytime

Write: Anytime

**NOTE**

Register bits PCLKAB0 to PCLKAB7 can be written anytime. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

**Table 9-11. PWMCLK Field Descriptions**

**Note:** Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

| Field | Description |
|---|---|
| 7<br>PCLKAB7 | **Pulse Width Channel 7 Clock A/B Select**<br>0 Clock B or SB is the clock source for PWM channel 7, as shown in Table 9-6.<br>1 Clock A or SA is the clock source for PWM channel 7, as shown in Table 9-6. |
| 6<br>PCLKAB6 | **Pulse Width Channel 6 Clock A/B Select**<br>0 Clock B or SB is the clock source for PWM channel 6, as shown in Table 9-6.<br>1 Clock A or SA is the clock source for PWM channel 6, as shown in Table 9-6. |
| 5<br>PCLKAB5 | **Pulse Width Channel 5 Clock A/B Select**<br>0 Clock A or SA is the clock source for PWM channel 5, as shown in Table 9-5.<br>1 Clock B or SB is the clock source for PWM channel 5, as shown in Table 9-5. |
| 4<br>PCLKAB4 | **Pulse Width Channel 4 Clock A/B Select**<br>0 Clock A or SA is the clock source for PWM channel 4, as shown in Table 9-5.<br>1 Clock B or SB is the clock source for PWM channel 4, as shown in Table 9-5. |
| 3<br>PCLKAB3 | **Pulse Width Channel 3 Clock A/B Select**<br>0 Clock B or SB is the clock source for PWM channel 3, as shown in Table 9-6.<br>1 Clock A or SA is the clock source for PWM channel 3, as shown in Table 9-6. |
| 2<br>PCLKAB2 | **Pulse Width Channel 2 Clock A/B Select**<br>0 Clock B or SB is the clock source for PWM channel 2, as shown in Table 9-6.<br>1 Clock A or SA is the clock source for PWM channel 2, as shown in Table 9-6. |
| 1<br>PCLKAB1 | **Pulse Width Channel 1 Clock A/B Select**<br>0 Clock A or SA is the clock source for PWM channel 1, as shown in Table 9-5.<br>1 Clock B or SB is the clock source for PWM channel 1, as shown in Table 9-5. |
| 0<br>PCLKAB0 | **Pulse Width Channel 0 Clock A/B Select**<br>0 Clock A or SA is the clock source for PWM channel 0, as shown in Table 9-5.<br>1 Clock B or SB is the clock source for PWM channel 0, as shown in Table 9-5. |

The clock source of each PWM channel is determined by PCLKx bits in PWMCLK (see Section 9.3.2.3, “PWM Clock Select Register (PWMCLK)) and PCLKABx bits in PWMCLKAB as shown in Table 9-5 and Table 9-6.

### 9.3.2.8 PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated by taking clock A, dividing it by the value in the PWMSCLA register and dividing that by two.

Clock SA = Clock A / (2 * PWMSCLA)

**NOTE**

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA).

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-10. PWM Scale A Register (PWMSCLA)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLA value)

### 9.3.2.9 PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated by taking clock B, dividing it by the value in the PWMSCLB register and dividing that by two.

Clock SB = Clock B / (2 * PWMSCLB)

**NOTE**

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB).

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-11. PWM Scale B Register (PWMSCLB)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLB value).

### 9.3.2.10 PWM Channel Counter Registers (PWMCNTx)

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source. The counter can be read at any time without affecting the count or the operation of the PWM channel. In left aligned output mode, the counter counts from 0 to the value in the period register - 1. In center aligned output mode, the counter counts from 0 up to the value in the period register and then back down to 0.

Any value written to the counter causes the counter to reset to $00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. The counter is also cleared at the end of the effective period (see Section 9.4.2.5, “Left Aligned Outputs” and Section 9.4.2.6, “Center Aligned Outputs” for more details). When the channel is disabled (PWMEx = 0), the PWMCNTx register does not count. When a channel becomes enabled (PWMEx = 1), the associated PWM counter starts at the count in the PWMCNTx register. For more detailed information on the operation of the counters, see Section 9.4.2.4, “PWM Timer Counters”.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

**NOTE**

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

Module Base + 0x000C = PWMCNT0, 0x000D = PWMCNT1, 0x000E = PWMCNT2, 0x000F = PWMCNT3
Module Base + 0x0010 = PWMCNT4, 0x0011 = PWMCNT5, 0x0012 = PWMCNT6, 0x0013 = PWMCNT7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-12. PWM Channel Counter Registers (PWMCNTx)**

[1] This register is available only when the corresponding channel exists and is reserved if that channel does not exist. Writes to a reserved register have no functional effect. Reads from a reserved register return zeroes.

Read: Anytime

Write: Anytime (any value written causes PWM counter to be reset to $00).

### 9.3.2.11 PWM Channel Period Registers (PWMPERx)

There is a dedicated period register for each channel. The value in this register determines the period of the associated PWM channel.

The period registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends

- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period register will go directly to the latches as well as the buffer.

**NOTE**

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active period due to the double buffering scheme.

See Section 9.4.2.3, “PWM Period and Duty” for more information.

To calculate the output period, take the selected clock source period for the channel of interest (A, B, SA, or SB) and multiply it by the value in the period register for that channel:

- Left aligned output (CAEx = 0)

  PWMx Period = Channel Clock Period * PWMPERx
- Center Aligned Output (CAEx = 1)

  PWMx Period = Channel Clock Period * (2 * PWMPERx)

For boundary case programming values, please refer to Section 9.4.2.8, “PWM Boundary Cases”.

Module Base + 0x0014 = PWMPER0, 0x0015 = PWMPER1, 0x0016 = PWMPER2, 0x0017 = PWMPER3
Module Base + 0x0018 = PWMPER4, 0x0019 = PWMPER5, 0x001A = PWMPER6, 0x001B = PWMPER7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 9-13. PWM Channel Period Registers (PWMPERx)**

1 This register is available only when the corresponding channel exists and is reserved if that channel does not exist. Writes to a reserved register have no functional effect. Reads from a reserved register return zeroes.

Read: Anytime

Write: Anytime

### 9.3.2.12 PWM Channel Duty Registers (PWMDTYx)

There is a dedicated duty register for each channel. The value in this register determines the duty of the associated PWM channel. The duty value is compared to the counter and if it is equal to the counter value a match occurs and the output changes state.

The duty registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)

- The channel is disabled

In this way, the output of the PWM will always be either the old duty waveform or the new duty waveform, not some variation in between. If the channel is not enabled, then writes to the duty register will go directly to the latches as well as the buffer.

**NOTE**

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active duty due to the double buffering scheme.

See Section 9.4.2.3, “PWM Period and Duty” for more information.

**NOTE**

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time. If the polarity bit is one, the output starts high and then goes low when the duty count is reached, so the duty registers contain a count of the high time. If the polarity bit is zero, the output starts low and then goes high when the duty count is reached, so the duty registers contain a count of the low time.

To calculate the output duty cycle (high time as a% of period) for a particular channel:

- Polarity = 0 (PPOL x =0)

  Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
- Polarity = 1 (PPOLx = 1)

  Duty Cycle = [PWMDTYx / PWMPERx] * 100%

For boundary case programming values, please refer to Section 9.4.2.8, “PWM Boundary Cases”.

Module Base + 0x001C = PWMDTY0, 0x001D = PWMDTY1, 0x001E = PWMDTY2, 0x001F = PWMDTY3
Module Base + 0x0020 = PWMDTY4, 0x0021 = PWMDTY5, 0x0022 = PWMDTY6, 0x0023 = PWMDTY7

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 9-14. PWM Channel Duty Registers (PWMDTYx)**

[1] This register is available only when the corresponding channel exists and is reserved if that channel does not exist. Writes to a reserved register have no functional effect. Reads from a reserved register return zeroes.

Read: Anytime

Write: Anytime

# 9.4 Functional Description

## 9.4.1 PWM Clock Select

There are four available clocks: clock A, clock B, clock SA (scaled A), and clock SB (scaled B). These four clocks are based on the bus clock.

Clock A and B can be software selected to be 1, 1/2, 1/4, 1/8,..., 1/64, 1/128 times the bus clock. Clock SA uses clock A as an input and divides it further with a reloadable counter. Similarly, clock SB uses clock B as an input and divides it further with a reloadable counter. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB. Each PWM channel has the capability of selecting one of four clocks, clock A, Clock B, clock SA or clock SB.

The block diagram in Figure 9-15 shows the four different clocks and how the scaled clocks are created.

### 9.4.1.1 Prescale

The input clock to the PWM prescaler is the bus clock. It can be disabled whenever the part is in freeze mode by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode (freeze mode signal active) the input clock to the prescaler is disabled. This is useful for emulation in order to freeze the PWM. The input clock can also be disabled when all available PWM channels are disabled (PWMEx-0 = 0). This is useful for reducing power by disabling the prescale counter.

Clock A and clock B are scaled values of the input clock. The value is software selectable for both clock A and clock B and has options of 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, or 1/128 times the bus clock. The value selected for clock A is determined by the PCKA2, PCKA1, PCKA0 bits in the PWMPRCLK register. The value selected for clock B is determined by the PCKB2, PCKB1, PCKB0 bits also in the PWMPRCLK register.

### 9.4.1.2 Clock Scale

The scaled A clock uses clock A as an input and divides it further with a user programmable value and then divides this by 2. The scaled B clock uses clock B as an input and divides it further with a user programmable value and then divides this by 2. The rates available for clock SA are software selectable to be clock A divided by 2, 4, 6, 8,..., or 512 in increments of divide by 2. Similar rates are available for clock SB.

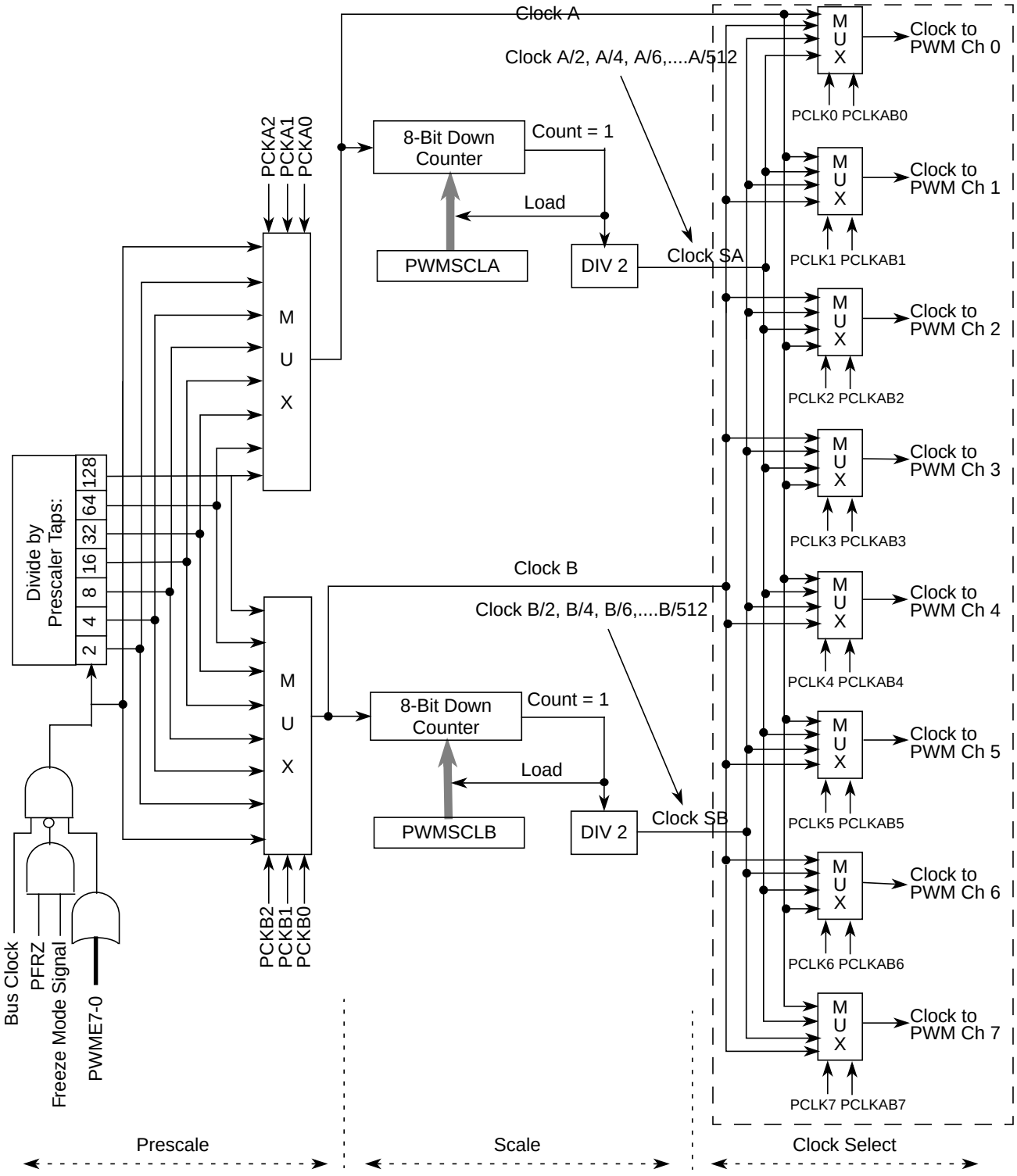



**Figure 9-15. PWM Clock Select Block Diagram**

Clock A is used as an input to an 8-bit down counter. This down counter loads a user programmable scale value from the scale register (PWMSCLA). When the down counter reaches one, a pulse is output and the 8-bit counter is re-loaded. The output signal from this circuit is further divided by two. This gives a greater range with only a slight reduction in granularity. Clock SA equals clock A divided by two times the value in the PWMSCLA register.

**NOTE**

Clock SA = Clock A / (2 * PWMSCLA)

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Similarly, clock B is used as an input to an 8-bit down counter followed by a divide by two producing clock SB. Thus, clock SB equals clock B divided by two times the value in the PWMSCLB register.

**NOTE**

Clock SB = Clock B / (2 * PWMSCLB)

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

As an example, consider the case in which the user writes $FF into the PWMSCLA register. Clock A for this case will be E (bus clock) divided by 4. A pulse will occur at a rate of once every 255x4 E cycles. Passing this through the divide by two circuit produces a clock signal at an E divided by 2040 rate. Similarly, a value of $01 in the PWMSCLA register when clock A is E divided by 4 will produce a clock at an E divided by 8 rate.

Writing to PWMSCLA or PWMSCLB causes the associated 8-bit down counter to be re-loaded. Otherwise, when changing rates the counter would have to count down to $01 before counting at the proper rate. Forcing the associated counter to re-load the scale register value every time PWMSCLA or PWMSCLB is written prevents this.

**NOTE**

Writing to the scale registers while channels are operating can cause irregularities in the PWM outputs.

#### 9.4.1.3 Clock Select

Each PWM channel has the capability of selecting one of four clocks, clock A, clock SA, clock B or clock SB. The clock selection is done with the PCLKx control bits in the PWMCLK register and PCLKABx control bits in PWMCLKAB register. For backward compatibility consideration, the reset value of PWMCLK and PWMCLKAB configures following default clock selection.

For channels 0, 1, 4, and 5 the clock choices are clock A.

For channels 2, 3, 6, and 7 the clock choices are clock B.

**NOTE**

Changing clock control bits while channels are operating can cause irregularities in the PWM outputs.

## 9.4.2 PWM Channel Timers

The main part of the PWM module are the actual timers. Each of the timer channels has a counter, a period register and a duty register (each are 8-bit). The waveform output period is controlled by a match between the period register and the value in the counter. The duty is controlled by a match between the duty register and the counter value and causes the state of the output to change during the period. The starting polarity of the output is also selectable on a per channel basis. Shown below in Figure 9-16 is the block diagram for the PWM timer.

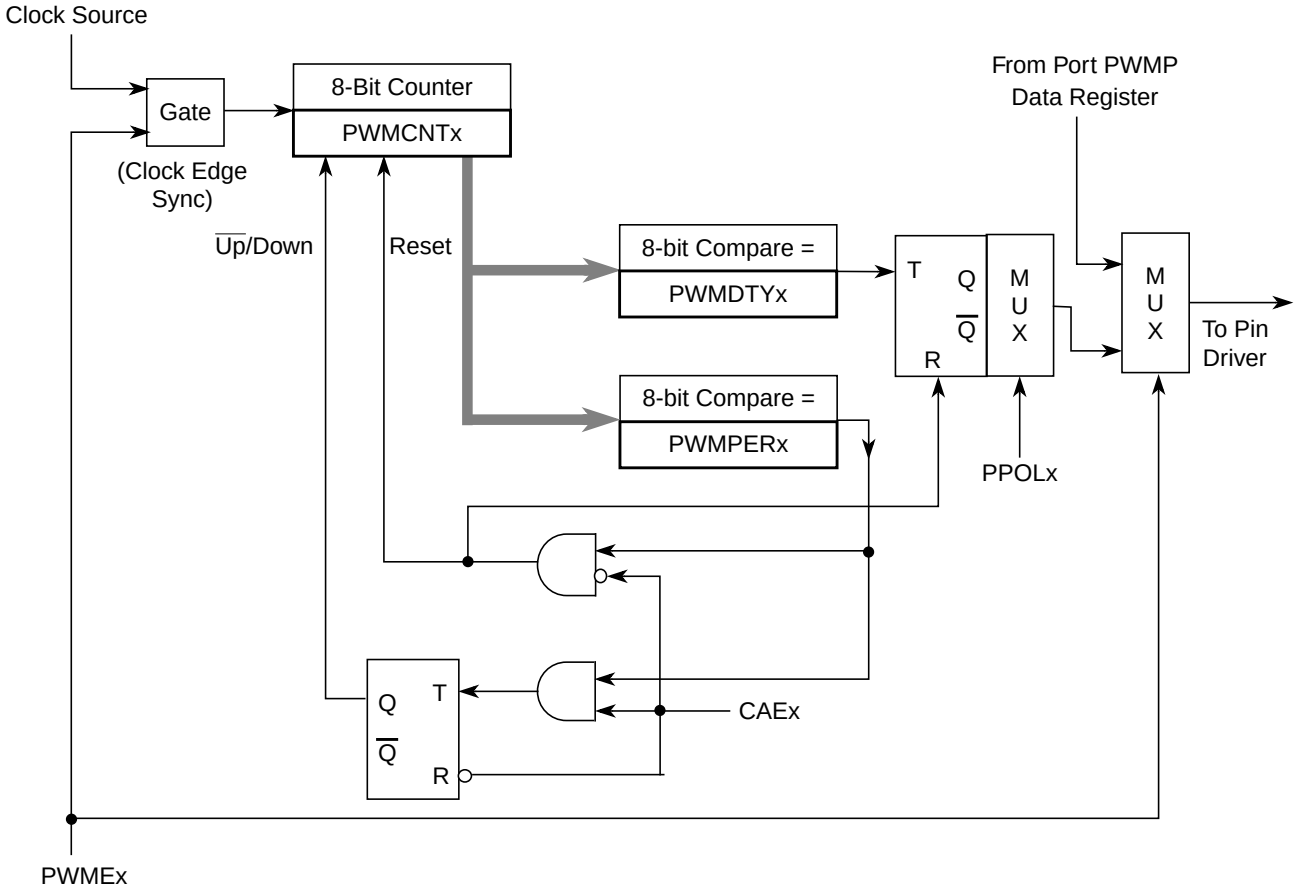



**Figure 9-16. PWM Timer Channel Block Diagram**

### 9.4.2.1 PWM Enable

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output signal is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source. An exception to this is when channels are concatenated. Refer to Section 9.4.2.7, “PWM 16-Bit Functions” for more detail.

**NOTE**

The first PWM cycle after enabling the channel can be irregular.

On the front end of the PWM timer, the clock is enabled to the PWM circuit by the PWMEx bit being high. There is an edge-synchronizing circuit to guarantee that the clock will only be enabled or disabled at an edge. When the channel is disabled (PWMEx = 0), the counter for the channel does not count.

#### 9.4.2.2 PWM Polarity

Each channel has a polarity bit to allow starting a waveform cycle with a high or low signal. This is shown on the block diagram Figure 9-16 as a mux select of either the Q output or the $\overline{Q}$ output of the PWM output flip flop. When one of the bits in the PWMPOL register is set, the associated PWM channel output is high at the beginning of the waveform, then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

#### 9.4.2.3 PWM Period and Duty

Dedicated period and duty registers exist for each channel and are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period and duty registers will go directly to the latches as well as the buffer.

A change in duty or period can be forced into effect “immediately” by writing the new value to the duty and/or period registers and then writing to the counter. This forces the counter to reset and the new duty and/or period values to be latched. In addition, since the counter is readable, it is possible to know where the count is with respect to the duty value and software can be used to make adjustments

**NOTE**

When forcing a new period or duty into effect immediately, an irregular PWM cycle can occur.

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time.

#### 9.4.2.4 PWM Timer Counters

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source (see Section 9.4.1, “PWM Clock Select” for the available clock sources and rates). The counter compares to two registers, a duty register and a period register as shown in Figure 9-16. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in Figure 9-16 and described in Section 9.4.2.5, “Left Aligned Outputs” and Section 9.4.2.6, “Center Aligned Outputs”.

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

Any value written to the counter causes the counter to reset to $00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. When the channel is disabled (PWMEx = 0), the counter stops. When a channel becomes enabled (PWMEx = 1), the associated PWM counter continues from the count in the PWMCNTx register. This allows the waveform to continue where it left off when the channel is re-enabled. When the channel is disabled, writing “0” to the period register will cause the counter to reset on the next selected clock.

**NOTE**

If the user wants to start a new “clean” PWM waveform without any “history” from the old waveform, the user must write to channel counter (PWMCNTx) prior to enabling the PWM channel (PWMEx = 1).

Generally, writes to the counter are done prior to enabling a channel in order to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled, except that the new period is started immediately with the output set according to the polarity bit.

**NOTE**

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

The counter is cleared at the end of the effective period (see Section 9.4.2.5, “Left Aligned Outputs” and Section 9.4.2.6, “Center Aligned Outputs” for more details).

**Table 9-12. PWM Timer Counter Conditions**

| Counter Clears ($00) | Counter Counts | Counter Stops |
|---|---|---|
| When PWMCNTx register written to any value | When PWM channel is enabled (PWMEx = 1). Counts from last value in PWMCNTx. | When PWM channel is disabled (PWMEx = 0) |
| Effective period ends | | |

### 9.4.2.5 Left Aligned Outputs

The PWM timer provides the choice of two types of outputs, left aligned or center aligned. They are selected with the CAEx bits in the PWMCAE register. If the CAEx bit is cleared (CAEx = 0), the corresponding PWM output will be left aligned.

In left aligned output mode, the 8-bit counter is configured as an up counter only. It compares to two registers, a duty register and a period register as shown in the block diagram in Figure 9-16. When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register resets the counter and the output flip-flop, as shown in Figure 9-16, as well as performing a load from the double buffer period and duty register to the associated registers, as described in Section 9.4.2.3, “PWM Period and Duty”. The counter counts from 0 to the value in the period register – 1.

**NOTE**

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

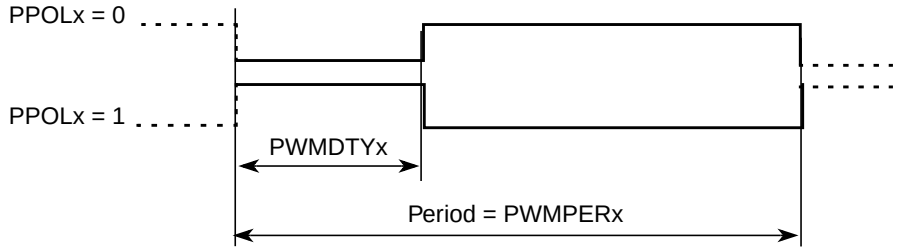



**Figure 9-17. PWM Left Aligned Output Waveform**

To calculate the output frequency in left aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / PWMPERx
- PWMx Duty Cycle (high time as a% of period):
  - Polarity = 0 (PPOLx = 0)

    Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - Polarity = 1 (PPOLx = 1)

    Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a left aligned output, consider the following case:

Clock Source = E, where E = 10 MHz (100 ns period)

PPOLx = 0

PWMPERx = 4

PWMDTYx = 1

PWMx Frequency = 10 MHz/4 = 2.5 MHz

PWMx Period = 400 ns

PWMx Duty Cycle = 3/4 *100% = 75%

The output waveform generated is shown in Figure 9-18.

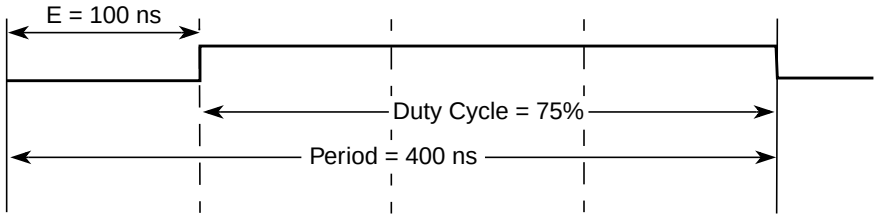



**Figure 9-18. PWM Left Aligned Output Example Waveform**

### 9.4.2.6 Center Aligned Outputs

For center aligned output mode selection, set the CAEx bit (CAEx = 1) in the PWMCAE register and the corresponding PWM output will be center aligned.

The 8-bit counter operates as an up/down counter in this mode and is set to up whenever the counter is equal to $00. The counter compares to two registers, a duty register and a period register as shown in the block diagram in Figure 9-16. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register changes the counter direction from an up-count to a down-count. When the PWM counter decrements and matches the duty register again, the output flip-flop changes state causing the PWM output to also change state. When the PWM counter decrements and reaches zero, the counter direction changes from a down-count back to an up-count and a load from the double buffer period and duty registers to the associated registers is performed, as described in Section 9.4.2.3, “PWM Period and Duty”. The counter counts from 0 up to the value in the period register and then back down to 0. Thus the effective period is PWMPERx*2.

**NOTE**

Changing the PWM output mode from left aligned to center aligned output (or vice versa) while channels are operating can cause irregularities in the PWM output. It is recommended to program the output mode before enabling the PWM channel.

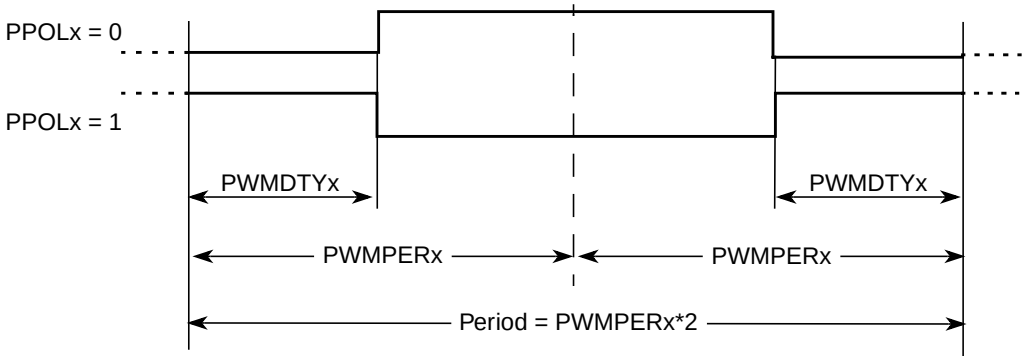



**Figure 9-19. PWM Center Aligned Output Waveform**

To calculate the output frequency in center aligned output mode for a particular channel, take the selected clock source frequency for the channel (A, B, SA, or SB) and divide it by twice the value in the period register for that channel.

- PWMx Frequency = Clock (A, B, SA, or SB) / (2*PWMPERx)
- PWMx Duty Cycle (high time as a% of period):
  - Polarity = 0 (PPOLx = 0)
    Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
  - Polarity = 1 (PPOLx = 1)
    Duty Cycle = [PWMDTYx / PWMPERx] * 100%

As an example of a center aligned output, consider the following case:

Clock Source = E, where E = 10 MHz (100 ns period)

PPOLx = 0

PWMPERx = 4

PWMDTYx = 1

PWMx Frequency = 10 MHz/8 = 1.25 MHz

PWMx Period = 800 ns

PWMx Duty Cycle = 3/4 *100% = 75%

Shown in Figure 9-20 is the output waveform generated.

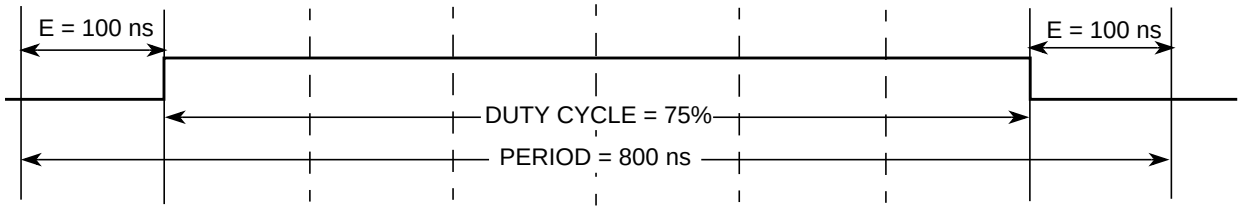



**Figure 9-20. PWM Center Aligned Output Example Waveform**

### 9.4.2.7 PWM 16-Bit Functions

The scalable PWM timer also has the option of generating up to 8-channels of 8-bits or 4-channels of 16-bits for greater PWM resolution. This 16-bit channel option is achieved through the concatenation of two 8-bit channels.

The PWMCTL register contains four control bits, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. Channels 6 and 7 are concatenated with the CON67 bit, channels 4 and 5 are concatenated with the CON45 bit, channels 2 and 3 are concatenated with the CON23 bit, and channels 0 and 1 are concatenated with the CON01 bit.

**NOTE**

Change these bits only when both corresponding channels are disabled.

When channels 6 and 7 are concatenated, channel 6 registers become the high order bytes of the double byte channel, as shown in Figure 9-21. Similarly, when channels 4 and 5 are concatenated, channel 4

registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated, channel 2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

When using the 16-bit concatenated mode, the clock source is determined by the low order 8-bit channel clock select control bits. That is channel 7 when channels 6 and 7 are concatenated, channel 5 when channels 4 and 5 are concatenated, channel 3 when channels 2 and 3 are concatenated, and channel 1 when channels 0 and 1 are concatenated. The resulting PWM is output to the pins of the corresponding low order 8-bit channel as also shown in Figure 9-21. The polarity of the resulting PWM output is controlled by the PPOLx bit of the corresponding low order 8-bit channel as well.

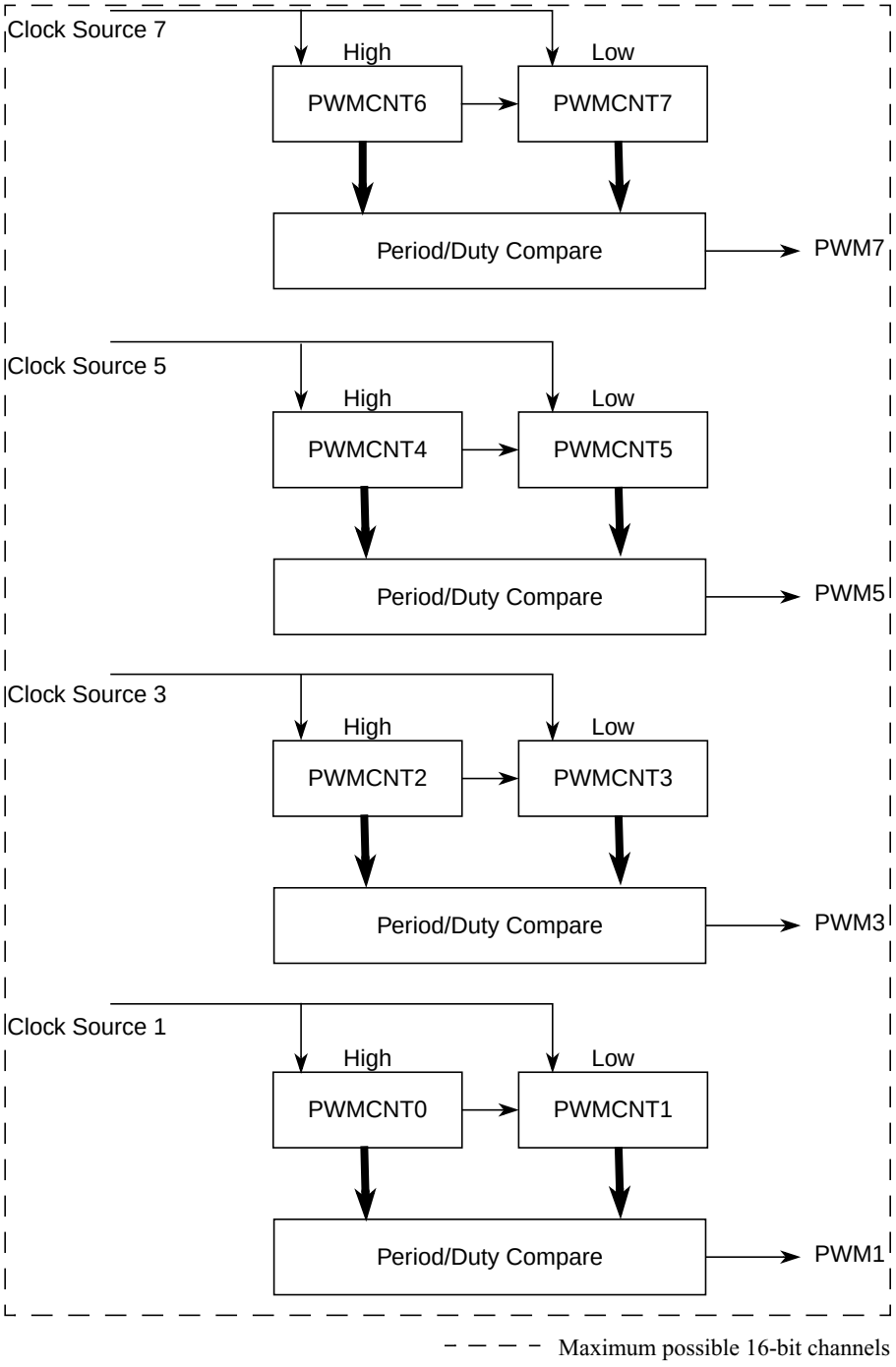



**Figure 9-21. PWM 16-Bit Mode**

Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit. In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output is disabled.

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

Either left aligned or center aligned output mode can be used in concatenated mode and is controlled by the low order CAEx bit. The high order CAEx bit has no effect.

Table 9-13 is used to summarize which channels are used to set the various control bits when in 16-bit mode.

**Table 9-13. 16-bit Concatenation Mode Summary**

**Note:** Bits related to available channels have functional significance.

| CONxx | PWMEx | PPOLx | PCLKx | CAEx | PWMx Output |
|---|---|---|---|---|---|
| CON67 | PWME7 | PPOL7 | PCLK7 | CAE7 | PWM7 |
| CON45 | PWME5 | PPOL5 | PCLK5 | CAE5 | PWM5 |
| CON23 | PWME3 | PPOL3 | PCLK3 | CAE3 | PWM3 |
| CON01 | PWME1 | PPOL1 | PCLK1 | CAE1 | PWM1 |

### 9.4.2.8 PWM Boundary Cases

Table 9-14 summarizes the boundary conditions for the PWM regardless of the output mode (left aligned or center aligned) and 8-bit (normal) or 16-bit (concatenation).

**Table 9-14. PWM Boundary Cases**

| PWMDTYx | PWMPERx | PPOLx | PWMx Output |
|---|---|---|---|
| $00 (indicates no duty) | >$00 | 1 | Always low |
| $00 (indicates no duty) | >$00 | 0 | Always high |
| XX | $00[1] (indicates no period) | 1 | Always high |
| XX | $00[1] (indicates no period) | 0 | Always low |
| >= PWMPERx | XX | 1 | Always high |
| >= PWMPERx | XX | 0 | Always low |

[1] Counter = $00 and does not count.

## 9.5 Resets

The reset state of each individual bit is listed within the Section 9.3.2, “Register Descriptions” which details the registers and their bit-fields. All special functions or modes which are initialized during or just following reset are described within this section.

- The 8-bit up/down counter is configured as an up counter out of reset.
- All the channels are disabled and all the counters do not count.

- For channels 0, 1, 4, and 5 the clock choices are clock A.
- For channels 2, 3, 6, and 7 the clock choices are clock B.

## 9.6 Interrupts

The PWM module has no interrupt.

# Chapter 10
# Analog-to-Digital Converter (ADC12B_LBA_V1)

## Revision History

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.15 | 20 Aug 2011 | | Initial version |
| V01.16 | 24 Aug 2011 | | Updated bit information for STR_SEQA, RIDX_IMD, SEQAD_IF and CONIF_OIF |
| V01.17 | 14. Sep 2011 | 10.4.1/10-355 | Added reserved register space 0x002A - 0x003F in Module Memory Map |
| V01.18 | 16. Sep 2011 | 10.4.1/10-355; 10.4.2.15/10-377; 10.4.2.16/10-379; 10.4.2.17/10-381; 10.4.2.18/10-383 1.4.2.20/1-44 10.2.1.2/10-350 | Updated Figures of registers in regard to write accessibility for ADCCMD_0 to ADCCMD_3 and ADCCMPHT and ADCCMPLT. Removed feature of interrupt flag clearing at Stop Mode or Wait Mode exit. |
| V01.19 | 21. Sep 2011 | 10.2.1.2/10-350 | Updated Stop Mode and Wait Mode description regarding conversion result storage and corresponding conversion flag setting. Added a Note regarding ADC behavior in case Wait Mode phase being shorter than a Sequence Abort Event takes to be executed. |
| V01.20 | 23. Sep 2011 | 10.2.1.2/10-350 10.4.2.3/10-361 | Added bit READY to ADCSTS register. |
| V01.21 | 05. Oct 2011 | 10.1/10-347 10.4.2.1/10-358, 10.4.2.6/10-364 | Added latency time for ADCFLWCTL register after ADC gets enabled. Added ADC disable time $t_{DISABLE}$ when ADC gets disabled and before ADC can be re-enabled. |
| V01.22 | 12. Oct 2011 | - | Minor corrections (wording, spaces, text format). Corrected cross references. |
| V01.23 | 18. Oct 2011 | 10.4.2.6/10-364 | Added information for timing considerations (Trigger Event - channel sample start) depending on ADC mode configuration. |
| V01.24 | 21. Oct 2011 | 10.4.2.6/10-364 | Updated format of footnote in Table 10-9. |
| V01.25 | 27. Oct 2011 | 10.2.1.2/10-350 | Updated information about RVL buffer select after exit from low power mode (Wait mode with bit SWAI set or Stop Mode). |

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.26 | 31. Jan 2012 | 10.2.1.2/10-350, 10.4.2.6/10-364, 10.5.2.2/10-390, 10.5.3.2.6/10-402, 10.8.7.2/10-410, 10.8.7.3/10-411, 10.8.8/10-412, 10.8.9/10-413, 10.8.10/10-414, | Listed sections are updated or newly added to provide more information about timing considerations, sampling phase details and conversion flow control application use cases |
| V01.27 | 14. Feb 2012 | 10.4.2.1/10-358, Table 10-9, 10.4.2.9/10-370, 10.5.3.2.4/10-398, 10.5.3.2.5/10-398, 10.5.3.2.6/10-402 | Moved parts of the Note information of section 10.4.2.1/10-358 to the new section 10.5.3.2.4/10-398. Updated Table 10-9./10-367 by adding Footnote 6. Updated TRIG_EIF bit description in section 10.4.2.9/10-370. Updated section 10.5.3.2.5/10-398. |
| V01.28 | 27. Feb 2012 | | Corrected Footnote information in Table 10-9. Re-arranged sections with use case information to end of document section Section 10.8, “Use Cases and Application Informationand following. Corrected typos. |
| V01.29 | 09. Mar 2012 | 10.8/10-406, 10.8.6/10-408, re-ordering sections below 10.5.3/10-391, add Figure 10-40 | Reorganized section order and grouped sections containing application and use case information underneath Section 10.8, “Use Cases and Application Information. Added new Section 10.8.6, “RVL swapping in RVL double buffer mode and related registers ADCIMDRI and ADCEOLRI and Figure 10-40. Corrected typos. |
| V1.30 | 04. May 2012 | 10.5.3.2.3/10-396 | Added conversion result justification information depending on bit DJM at Section 10.5.3.2.3, “Introduction of the two Result Value Lists (RVLs) |
| V1.31 | 27. July 2012 | 10.4.2.5/10-363, 10.4.2.3/10-361, 10.4.2.15/10-377 to 10.4.2.17/10-381 | Added footnote at Table 10-7 providing information in case of usage of reserved settings. Added Notes to the ADCCMD register descriptions in case of modifying these registers when debugging (bit SMOD_ACC set).Correction of typos and wording in the bit description for the flow control bits SEQA and RSTA. Description of bit DBECC_ERR updated. |
| V1.32 | 12. Sep 2012 | 10.5.3.2.6/10-402, 10.4.2.3/10-361, 10.8.5/10-408 | Added information for flag RSTAR_EIF. Updated information of RVL_SEL bit description. Added information in ADC Status Register that bits CSL_SEL and RVL_SEL are influenced by ADC Soft-Reset (information already given in ADC_SR description of previous releases).Updated use case and application information Section 10.8.5, “List Usage - CSL double buffer mode and RVL double buffer mode. |
| V1.33 | 13. Sep 2012 | 10.4.2.8/10-369, 10.4.2.10/10-372 | Unspecified flow control issue added (case of Trigger Event occurring before Restart Event to start execution of a Command Sequence List. Added Note containing detailed information regarding flag CONIF_OIF |
| V1.34 | 11. Oct 2012 | 10.5.3.2.6/10-402 | Added information to “Trigger Overrun” scenario: Restart Request Overrun in “Trigger Mode” and the related impact on Trigger Request Overrun. See Section 10.5.3.2.6, “Conversion flow control in case of conversion sequence control bit overrun scenarios. |
| V1.35 | 06. Nov 2012 | 10.4.2.9/10-370 | Modified bit description of flag LDOK_EIF for better understanding. |
| V1.36 | 08 Nov 2012 | 10.4.2.13/10-375 | Updated description of bits RIDX_IMD for better understanding. |
| V1.37 | 19. Apr 2013 | - | Updates from review of Reference Manual to fix typos etc. |

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V1.38 | 30. Apr 2013 | 10.4.2.13/10-375 | Provided more detailed information regarding captured information in bits RIDX_IMD[5:0] for different scenarios of Sequence Abort Event execution. |
| V1.39 | 02. Jul 2013 | 10.4.2.6/10-364 | Update of: Timing considerations for Restart Mode |

## 10.1 Introduction

The ADC12B_LBA is an n-channel multiplexed input successive approximation analog-to-digital converter. Refer to device electrical specifications for ADC parameters and accuracy.

The List Based Architecture (LBA) provides flexible conversion sequence definition as well as flexible oversampling. The order of channels to be converted can be freely defined. Also multiple instantiations of the module can be triggered simultaneously (matching sampling point across multiple module instantiations).

There are four register bits which control the conversion flow (please refer to the description of register ADCFLWCTL).
The four conversion flow control bits of register ADCFLWCTL can be modified in two different ways:

- Via Data-Bus accesses
- Via Internal Interface Signals (Trigger, Restart, LoadOK and Seq_Abort, see also Figure 10-2). Each Interface Signal is associated with one conversion flow control bit.

For information regarding Internal Interface connectivity related to the conversion flow control please refer to the device overview of the Reference Manual.

Register ADCFLWCTL can be controlled via Internal Interface only or via Data-Bus only or by both depending on the register access configuration bits ACC_CFG[1:0].

The four bits of register ADCFLWCTL reflect the captured request and status of the four Internal Interface Signals (LoadOK, Trigger, Restart and Seq_abort, see also Figure 10-2) if access configuration is set accordingly and indicate event progress (when an event is processed and when it is finished).

Conversion flow error situations are captured by corresponding interrupt flags in the ADCEIF register.

There are two Conversion Flow Control Modes (Restart Mode, Trigger Mode). Each mode causes a certain behavior of the conversion flow control bits which can be selected according to the application needs.

Please refer to Section 10.4.2.1, “ADC Control Register 0 (ADCCTL_0) and Section 10.5.3.2.4, “The two conversion flow control Mode Configurations for more information regarding conversion flow control.

Because internal components of the ADC are turned on/off with bit ADC_EN, the ADC requires a recovery time period ($t_{REC}$) after ADC is enabled until the first conversion can be launched via a Trigger.

When bit ADC_EN gets cleared (transition from 1’b1 to 1’b0) any ongoing conversion sequence will be aborted and pending results or the result of current conversion gets discarded (not stored). The ADC can

not be re-enabled before any pending action or action in process is finished respectively aborted which could take up to a maximum latency time of $t_{DISABLE}$ (see device Reference Manual for more details).

## 10.2 Key Features

- Programmers model with List Based Architecture for conversion command and result value organization
- Selectable resolution of 8-bit, 10-bit or 12-bit
- Channel select control for n external analog input channels
- Provides up to eight device internal channels (please see the device Reference Manual for connectivity information and Figure 10-2)
- Programmable sample time
- Asample buffer amplifier for channel sampling (improved performance in view to influence of channel input path resistance versus conversion accuracy)
- Left/right justified result data
- Individual selectable VRH_0/1 and VRL_0/1 inputs on a conversion command basis (please see Figure 10-2)
- Special conversions for selected VRH_0/1, VRL_0/1, (VRL_0/1 + VRH_0/1) / 2
- 15 conversion interrupts with flexible interrupt organization per conversion result
- One dedicated interrupt for “End Of List” type commands
- Command Sequence List (CSL) with a maximum number of 64 command entries
- Provides conversion sequence abort
- Restart from top of active Command Sequence List (CSL)
- The Command Sequence List and Result Value List are implemented in double buffered manner (two lists in parallel for each function)
- Conversion Command (CSL) loading possible from System RAM or NVM
- Single conversion flow control register with software selectable access path
- Two conversion flow control modes optimized to different application use cases

## 10.2.1 Modes of Operation

### 10.2.1.1 Conversion Modes

This architecture provides **single**, **multiple** or **continuous conversion** on a **single channel** or on **multiple channels based on the Command Sequence List**.

### 10.2.1.2 MCU Operating Modes

- **MCU Stop Mode**
  Before issuing an MCU Stop Mode request the ADC should be idle (no conversion or conversion sequence or Command Sequence List ongoing).
  If an MCU Stop Mode request is issued whilst a conversion or a conversion sequence or CSL is in progress a Sequence Abort Event is issued automatically and a possible ongoing conversion will be finished.
  After the Sequence Abort Event is finished depending on bit STR_SEQA the conversion result is stored and corresponding flags are set (STR_SEQA=1) or conversion result is not stored and corresponding flags are not set (STR_SEQA=0). Following the MCU Stop Mode is entered without bit SEQAD_IF being set.
  Alternatively the Sequence Abort Event can be issued by software before MCU Stop Mode request. As soon as flag SEQAD_IF is set the MCU Stop Mode request can be is issued.

  With MCU Stop Mode Request until exit from Stop Mode all Flow Control Signals (RSTA, SEQA, LDOK, TRIG) are cleared.

  After exit from MCU Stop Mode the following happens in the order given with expected event(s) depending on the conversion flow control mode:
  — In ADC conversion flow control mode “Trigger Mode” a Restart Event is expected to simultaneously set bits TRIG and RSTA causing the ADC to execute the Restart Event (CMD_IDX and RVL_IDX cleared) followed by the Trigger Event. The Restart Event can be generated automatically after exit from MCU Stop Mode if bit AUT_RSTA is set.
  — In ADC conversion flow control mode “Restart Mode” a Restart Event is expected to set bit RSTA only (ADC already aborted at MCU Stop Mode entry hence bit SEQA must not be set simultaneously) causing the ADC to execute the Restart Event (CDM_IDX and RVL_IDX cleared). The Restart Event can be generated automatically after exit from MCU Stop Mode if bit AUT_RSTA is set.
  — The RVL buffer select (RVL_SEL) is not changed if a CSL was in process at MCU Stop Mode request. Hence the same buffer will be used after exit from Stop Mode that was used when Stop Mode request occurred.

- **MCU Wait Mode**
  Depending on the ADC Wait Mode configuration bit SWAI the ADC either continues conversion in MCU Wait Mode or freezes conversion at next conversion boundary before the MCU Wait Mode is entered.

  *The ADC behavior for configuration SWAI =1'b0:*
  The ADC continues conversion during Wait Mode according to the conversion flow control sequence. It is assumed that the conversion flow control sequence is continued (conversion flow control bits TRIG, RSTA, SEQA and LDOK are serviced accordingly).

  *ADC behavior for configuration SWAI = 1'b1:*
  At MCU Wait Mode request the ADC should be idle (no conversion or conversion sequence or Command Sequence List ongoing).
  If at MCU Wait Mode request a conversion or a conversion sequence or CSL is in progress a Sequence Abort Event is issued automatically and a possible ongoing conversion will be finished. After the Sequence Abort Event is finished depending on bit STR_SEQA the conversion result is stored and corresponding flags are set (STR_SEQA=1) or the conversion result is not stored and corresponding flags are not set (STR_SEQA=0). Following the MCU Wait Mode is entered without bit SEQAD_IF being set.
  Alternatively the Sequence Abort Event can be issued by software before MCU Wait Mode request. As soon as flag SEQAD_IF is set the MCU Wait Mode request can be issued.
  With MCU Wait Mode Request until exit from Wait Mode all Flow Control Signals (RSTA, SEQA, LDOK, TRIG) are cleared.
  After exit from MCU Wait Mode the following happens in the order given with expected event(s) depending on the conversion flow control mode:
  — In ADC conversion flow control mode "Trigger Mode" a Restart Event is expected to occur. This simultaneously sets bit TRIG and RSTA causing the ADC to execute the Restart Event (CMD_IDX and RVL_IDX cleared) followed by the Trigger Event. The Restart Event can be generated automatically after exit from MCU Wait Mode if bit AUT_RSTA is set.
  — In ADC conversion flow control mode "Restart Mode" a Restart Event is expected to set bit RSTA only (ADC already aborted at MCU Wait Mode entry hence bit SEQA must not be set simultaneously) causing the ADC to execute the Restart Event (CDM_IDX and RVL_IDX cleared). The Restart Event can be generated automatically after exit from MCU Wait Mode if bit AUT_RSTA is set.
  — The RVL buffer select (RVL_SEL) is not changed if a CSL was in process at MCU Wait Mode request. Hence the same RVL buffer will be used after exit from Wait Mode that was used when Wait Mode request occurred.

**NOTE**

In principle the MCU could stay in Wait Mode for a shorter period of time than the ADC needs to abort an ongoing conversion (range of us). Therefore in case a Sequence Abort Event is issued automatically due to MCU Wait Mode request a following Restart Event after exit from MCU Wait Mode can not be executed before ADC has finished this Sequence Abort Event. The Restart Event is detected but it is pending.
This applies in case the MCU Wait Mode is exited before ADC has finished the Sequence Abort Event and a Restart Event is issued immediately after exit from MCU Wait Mode. Bit READY can be used by software to detect when the Restart Event can be issued without latency time in processing the event (see also Figure 10-1).

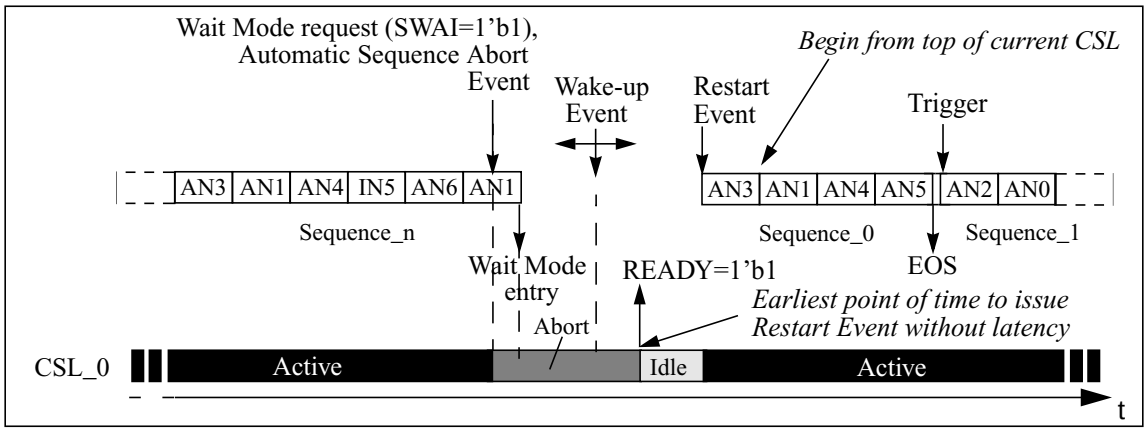



**Figure 10-1. Conversion Flow Control Diagram - Wait Mode (SWAI=1'b1, AUT_RSTA=1'b0)**

- **MCU Freeze Mode**
  Depending on the ADC Freeze Mode configuration bit FRZ_MOD the ADC either continues conversion in Freeze Mode or freeze conversion at next conversion boundary before the MCU Freeze Mode is entered. After exit from MCU Freeze Mode with previously frozen conversion sequence the ADC continues the conversion with the next conversion command and all ADC interrupt flags are unchanged during MCU Freeze Mode.

## 10.2.2 Block Diagram

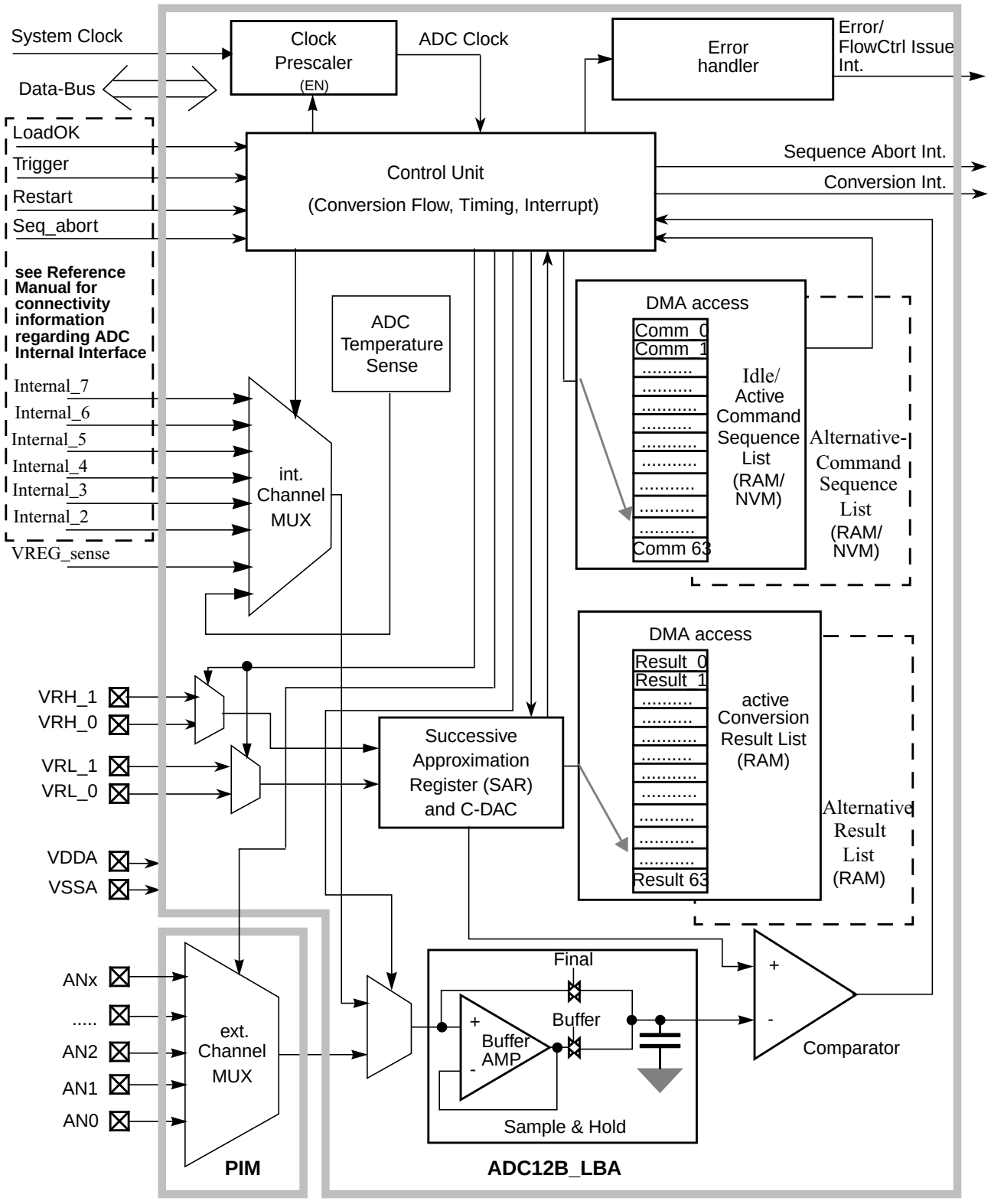



**Figure 10-2. ADC12B_LBA Block Diagram**

## 10.3 Signal Description

This section lists all inputs to the ADC12B_LBA block.

### 10.3.1 Detailed Signal Descriptions

#### 10.3.1.1 AN*x* (*x* = *n*,..., 2, 1, 0)

This pin serves as the analog input Channel *x.* The maximum input channel number is *n*. Please refer to the device Reference Manual for the maximum number of input channels.

#### 10.3.1.2 VRH_0, VRH_1, VRL_0, VRL_1

VRH_0/1 are the high reference voltages, VRL0/1 are the low reference voltages for a ADC conversion selectable on a conversion command basis. Please refer to the device Reference Manual for availability and connectivity of these pins.

#### 10.3.1.3 VDDA, VSSA

These pins are the power supplies for the analog circuitry of the ADC12B_LBA block.

## 10.4 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the ADC12B_LBA.

### 10.4.1 Module Memory Map

Figure 10-3 gives an overview of all ADC12B_LBA registers.

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | ADCCTL_0 | R<br>W | ADC_EN | ADC_SR | FRZ_MOD | SWAI | ACC_CFG[1:0] | | STR_SEQA | MOD_CFG |
| 0x0001 | ADCCTL_1 | R<br>W | CSL_BMOD | RVL_BMOD | SMOD_ACC | AUT_RSTA | 0 | 0 | 0 | 0 |
| 0x0002 | ADCSTS | R<br>W | CSL_SEL | RVL_SEL | DBECC_ERR | Reserved | READY | 0 | 0 | 0 |
| 0x0003 | ADCTIM | R<br>W | 0 | PRS[6:0] | | | | | | |
| 0x0004 | ADCFMT | R<br>W | DJM | 0 | 0 | 0 | 0 | SRES[2:0] | | |
| 0x0005 | ADCFLWCTL | R<br>W | SEQA | TRIG | RSTA | LDOK | 0 | 0 | 0 | 0 |
| 0x0006 | ADCEIE | R<br>W | IA_EIE | CMD_EIE | EOL_EIE | Reserved | TRIG_EIE | RSTAR_EIE | LDOK_EIE | 0 |
| 0x0007 | ADCIE | R<br>W | SEQAD_IE | CONIF_0IE | Reserved | 0 | 0 | 0 | 0 | 0 |
| 0x0008 | ADCEiF | R<br>W | IA_EIF | CMD_EIF | EOL_EIF | Reserved | TRIG_EIF | RSTAR_EIF | LDOK_EIF | 0 |
| 0x0009 | ADCIF | R<br>W | SEQAD_IF | CONIF_0IF | Reserved | 0 | 0 | 0 | 0 | 0 |
| 0x000A | ADCCONIE_0 | R<br>W | CON_IE[15:8] | | | | | | | |
| 0x000B | ADCCONIE_1 | R<br>W | CON_IE[7:1] | | | | | | | EOL_IE |
| 0x000C | ADCCONIF_0 | R<br>W | CON_IF[15:8] | | | | | | | |
| 0x000D | ADCCONIF_1 | R<br>W | CON_IF[7:1] | | | | | | | EOL_IF |
| 0x000E | ADCIMDRI_0 | R | CSL_IMD | RVL_IMD | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000F | ADCIMDRI_1 | R<br>W | 0 | 0 | RIDX_IMD[5:0] | | | | | |

= Unimplemented or Reserved

**Figure 10-3. ADC12B_LBA Register Summary (Sheet 1 of 3)**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0010 | ADCEOLRI | R | CSL_EOL | RVL_EOL | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0011 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0012 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0013 | Reserved | R | Reserved | Reserved | | | | | 0 | 0 |
| | | W | | | | | | | | |
| 0x0014 | ADCCMD_0 | R | CMD_SEL | | 0 | 0 | INTFLG_SEL[3:0] | | | |
| | | W | | | | | | | | |
| 0x0015 | ADCCMD_1 | R | VRH_SEL | VRL_SEL | CH_SEL[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x0016 | ADCCMD_2 | R | SMP[4:0] | | | | | 0 | 0 | Reserved |
| | | W | | | | | | | | |
| 0x0017 | ADCCMD_3 | R | Reserved | Reserved | Reserved | | | | | |
| | | W | | | | | | | | |
| 0x0018 | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x0019 | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x001A | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x001B | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x001C | ADCCIDX | R | 0 | 0 | CMD_IDX[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x001D | ADCCBP_0 | R | CMD_PTR[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x001E | ADCCBP_1 | R | CMD_PTR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x001F | ADCCBP_2 | R | CMD_PTR[7:2] | | | | | | 0 | 0 |
| | | W | | | | | | | | |
| 0x0020 | ADCRIDX | R | 0 | 0 | RES_IDX[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x0021 | ADCRBP_0 | R | 0 | 0 | 0 | 0 | RES_PTR[19:16] | | | |
| | | W | | | | | | | | |
| 0x0022 | ADCRBP_1 | R | RES_PTR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0023 | ADCRBP_2 | R | RES_PTR[7:2] | | | | | | 0 | 0 |
| | | W | | | | | | | | |
| 0x0024 | ADCCROFF0 | R | 0 | CMDRES_OFF0[6:0] | | | | | | |
| | | W | | | | | | | | |
| 0x0025 | ADCCROFF1 | R | 0 | CMDRES_OFF1[6:0] | | | | | | |
| | | W | | | | | | | | |
| 0x0026 | Reserved | R | 0 | 0 | 0 | 0 | Reserved | | | |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 10-3. ADC12B_LBA Register Summary (Sheet 2 of 3)**

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0027 | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x0028 | Reserved | R | Reserved | | | | | | 0 | 0 |
| | | W | | | | | | | | |
| 0x0029 | Reserved | R | Reserved | 0 | Reserved | | | | | |
| | | W | | | | | | | | |
| 0x002A-0x003F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 10-3. ADC12B_LBA Register Summary (Sheet 3 of 3)**

## 10.4.2 Register Descriptions

This section describes in address order all the ADC12B_LBA registers and their individual bits.

### 10.4.2.1 ADC Control Register 0 (ADCCTL_0)

Module Base + 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ADC_EN | ADC_SR | FRZ_MOD | SWAI | ACC_CFG[1:0] | | STR_SEQA | MOD_CFG |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-4. ADC Control Register 0 (ADCCTL_0)**

Read: Anytime

Write:

- Bits ADC_EN, ADC_SR, FRZ_MOD and SWAI writable anytime
- Bits MOD_CFG, STR_SEQA and ACC_CFG[1:0] writable if bit ADC_EN clear or bit SMOD_ACC set

**Table 10-1. ADCCTL_0 Field Descriptions**

| Field | Description |
|---|---|
| 15<br>ADC_EN | **ADC Enable Bit** — This bit enables the ADC (e.g. sample buffer amplifier etc.) and controls accessibility of ADC register bits. When this bit gets cleared any ongoing conversion sequence will be aborted and pending results or the result of current conversion gets discarded (not stored). The ADC can not be re-enabled before any pending action or action in process is finished respectively aborted which could take up to a maximum latency time of $t_{DISABLE}$ (see device Reference Manual for more details).<br>Because internal components of the ADC are turned on/off with this bit, the ADC requires a recovery time period ($t_{REC}$) after ADC is enabled until the first conversion can be launched via a Trigger.<br>0 ADC disabled.<br>1 ADC enabled. |
| 14<br>ADC_SR | **ADC Soft-Reset** — This bit causes an ADC Soft-Reset if set after a severe error occurred (see list of severe errors at Section 10.4.2.9, “ADC Error Interrupt Flag Register (ADCEIF) which causes the ADC to cease operation). It clears all overrun flags and error flags and forces ADC statemachine to its idle state as well as clearing the command index register, result index register and CSL_SEL and RVL_SEL bit (to be ready for a new control sequence to load new command and start execution again from top of selected CSL).<br>A severe error occurs if an error flag is set which cause the ADC to cease operation.<br>In order to make the ADC operational again an ADC Soft-Reset must be issued.<br>Once this bit is set it can not be cleared by writing any value. It is cleared only by ADC hardware after the soft reset has been executed.<br>0 No ADC soft reset issued.<br>1 Issue ADC soft reset. |
| 13<br>FRZ_MOD | **Freeze Mode Configuration** — This bit influences conversion flow during Freeze Mode.<br>0 ADC continues conversion in Freeze Mode.<br>1 ADC freezes the conversion at next conversion boundary at Freeze Mode entry. |
| 12<br>SWAI | **Wait Mode Configuration** — This bit influences conversion flow during Wait Mode.<br>0 ADC continues conversion in Wait Mode.<br>1 ADC halts the conversion at next conversion boundary at Wait Mode entry. |

**Table 10-1. ADCCTL_0 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 11-10<br>ACC_CFG[1:0] | **ADCFLWCTL Register Access Configuration** — These bits define if the register ADCFLWCTL is controlled via Internal Interface only or Data-Bus only or both. See Table 10-2. for more details. |
| 9<br>STR_SEQA | **Control Of Conversion Result Storage and RSTAR_EIF flag setting at Sequence Abort or Restart Event** — This bit controls conversion result storage and RSTAR_EIF flag setting when a Sequence Abort Event or Restart Event occurs as follows:<br>*If STR_SEQA = 1'b0 and if a:*<br>• Sequence Abort Event or Restart Event is issued during a conversion the data of this conversion is not stored and the respective conversion complete flag is not set<br>• Restart Event only is issued before the last conversion of a CSL is finished and no Sequence Abort Event is in process (SEQA clear) causes the RSTA_EIF error flag to be asserted and bit SEQA gets set by hardware<br>*If STR_SEQA = 1'b1 and if a:*<br>• Sequence Abort Event or Restart Event is issued during a conversion the data of this conversion is stored and the respective conversion complete flag is set and Intermediate Result Information Register is updated.<br>• Restart Event only occurs during the last conversion of a CSL and no Sequence Abort Event is in process (SEQA clear) does not set the RSTA_EIF error flag<br>• Restart Event only is issued before the CSL is finished and no Sequence Abort Event is in process (SEQA clear) causes the RSTA_EIF error flag to be asserted and bit SEQA gets set by hardware |
| 8<br>MOD_CFG | **(Conversion Flow Control) Mode Configuration** — This bit defines the conversion flow control after a Restart Event and after execution of the "End Of List" command type:<br>- Restart Mode<br>- Trigger Mode<br>(For more details please see also section Section 10.5.3.2, "Introduction of the Programmers Model and following.)<br>0 "Restart Mode" selected.<br>1 "Trigger Mode" selected. |

**Table 10-2. ADCFLWCTL Register Access Configurations**

| ACC_CFG[1] | ACC_CFG[0] | ADCFLWCTL Access Mode |
|---|---|---|
| 0 | 0 | None of the access paths is enabled<br>(default configuration) |
| 0 | 1 | Single Access Mode - Internal Interface<br>(ADCFLWCTL access via Internal Interface only) |
| 1 | 0 | Single Access Mode - Data Bus<br>(ADCFLWCTL access via Data Bus only) |
| 1 | 1 | Dual Access Mode<br>(ADCFLWCTL register access via Internal Interface and Data Bus) |

## NOTE

Each conversion flow control bit (SEQA, RSTA, TRIG, LDOK) must be controlled by software or Internal Interface according to the requirements described in Section 10.5.3.2.4, "The two conversion flow control Mode Configurations and overview summary in Table 10-9.

### 10.4.2.2 ADC Control Register 1 (ADCCTL_1)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CSL_BMOD | RVL_BMOD | SMOD_ACC | AUT_RSTA | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-5. ADC Control Register 1 (ADCCTL_1)**

Read: Anytime

Write:

- Bit CSL_BMOD and RVL_BMOD writable if bit ADC_EN clear or bit SMOD_ACC set
- Bit SMOD_ACC only writable in MCU Special Mode
- Bit AUT_RSTA writable anytime

**Table 10-3. ADCCTL_1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CSL_BMOD | **CSL Buffer Mode Select Bit** — This bit defines the CSL buffer mode. This bit is only writable if ADC_EN is clear.<br>0 CSL single buffer mode.<br>1 CSL double buffer mode. |
| 6<br>RVL_BMOD | **RVL Buffer Mode Select Bit** — This bit defines the RVL buffer mode.<br>0 RVL single buffer mode<br>1 RVL double buffer mode |
| 5<br>SMOD_ACC | **Special Mode Access Control Bit** — This bit controls register access rights in MCU Special Mode. This bit is automatically cleared when leaving MCU Special Mode.<br>Note: When this bit is set also the ADCCMD register is writeable via the data bus to allow modification of the current command for debugging purpose. But this is only possible if the current command is not already processed (conversion not started).<br>Please see access details given for each register.<br>Care must be taken when modifying ADC registers while bit SMOD_ACC is set to not corrupt a possible ongoing conversion.<br>0 Normal user access - Register write restrictions exist as specified for each bit.<br>1 Special access - Register write restrictions are lifted. |
| 4<br>AUT_RSTA | **Automatic Restart Event after exit from MCU Stop and Wait Mode (SWAI set)** — This bit controls if a Restart Event is automatically generated after exit from MCU Stop Mode or Wait Mode with bit SWAI set. It can be configured for ADC conversion flow control mode “Trigger Mode” and “Restart Mode” (anytime during application runtime).<br>0 No automatic Restart Event after exit from MCU Stop Mode.<br>1 Automatic Restart Event occurs after exit from MCU Stop Mode. |

### 10.4.2.3 ADC Status Register (ADCSTS)

It is important to note that if flag DBECC_ERR is set the ADC ceases operation. In order to make the ADC operational again an ADC Soft-Reset must be issued. An ADC Soft-Reset clears bits CSL_SEL and RVL_SEL.

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CSL_SEL | RVL_SEL | DBECC_ERR | Reserved | READY | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-6. ADC Status Register (ADCSTS)**

Read: Anytime

Write:

- Bits CSL_SEL and RVL_SEL anytime if bit ADC_EN is clear or bit SMOD_ACC is set
- Bits DBECC_ERR and READY not writable

**Table 10-4. ADCSTS Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CSL_SEL | **Command Sequence List Select bit** — This bit controls and indicates which ADC Command List is active. This bit can only be written if ADC_EN bit is clear. This bit toggles in CSL double buffer mode when no conversion or conversion sequence is ongoing and bit LDOK is set and bit RSTA is set. In CSL single buffer mode this bit is forced to 1'b0 by bit CSL_BMOD.<br>0 ADC Command List 0 is active.<br>1 ADC Command List 1 is active. |
| 6<br>RVL_SEL | **Result Value List Select Bit** — This bit controls and indicates which ADC Result List is active. This bit can only be written if bit ADC_EN is clear. After storage of the initial Result Value List this bit toggles in RVL double buffer mode whenever the conversion result of the first conversion of the current CSL is stored or a CSL got aborted. In RVL single buffer mode this bit is forced to 1'b0 by bit RVL_BMOD.<br>Please see also Section 10.2.1.2, "MCU Operating Modes for information regarding Result List usage in case of Stop or Wait Mode.<br>0 ADC Result List 0 is active.<br>1 ADC Result List 1 is active. |
| 5<br>DBECC_ERR | **Double Bit ECC Error Flag** — This flag indicates that a double bit ECC error occurred during conversion command load or result storage and ADC ceases operation.<br>In order to make the ADC operational again an ADC Soft-Reset must be issued.<br>This bit is cleared if bit ADC_EN is clear.<br>0 No double bit ECC error occurred.<br>1 A double bit ECC error occurred. |
| 3<br>READY | **Ready For Restart Event Flag** — This flag indicates that ADC is in its idle state and ready for a Restart Event. It can be used to verify after exit from Wait Mode if a Restart Event can be issued and processed immediately without any latency time due to an ongoing Sequence Abort Event after exit from MCU Wait Mode (see also the Note in Section 10.2.1.2, "MCU Operating Modes).<br>0 ADC not in idle state.<br>1 ADC is in idle state. |

### 10.4.2.4 ADC Timing Register (ADCTIM)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PRS[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

= Unimplemented or Reserved

**Figure 10-7. ADC Timing Register (ADCTIM))**

Read: Anytime

Write: These bits are writable if bit ADC_EN is clear or bit SMOD_ACC is set

**Table 10-5. ADCTIM Field Descriptions**

| Field | Description |
|---|---|
| 6-0<br>PRS[6:0] | **ADC Clock Prescaler** — These 7bits are the binary prescaler value PRS. The ADC conversion clock frequency is calculated as follows:<br>$f_{ATDCLK} = \frac{f_{BUS}}{2 \times (PRS + 1)}$<br>Refer to Device Specification for allowed frequency range of $f_{ATDCLK}$. |

### 10.4.2.5 ADC Format Register (ADCFMT)

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DJM | 0 | 0 | 0 | 0 | SRES[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-8. ADC Format Register (ADCFMT)**

Read: Anytime

Write: Bits DJM and SRES[2:0] are writable if bit ADC_EN clear or bit SMOD_ACC set

**Table 10-6. ADCFMT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DJM | **Result Register Data Justification** — Conversion result data format is always unsigned. This bit controls justification of conversion result data in the conversion result list.<br>0 Left justified data in the conversion result list.<br>1 Right justified data in the conversion result list. |
| 2-0<br>SRES[2:0] | **ADC Resolution Select** — These bits select the resolution of conversion results. See Table 10-7 for coding. |

**Table 10-7. Selectable Conversion Resolution**

| SRES[2] | SRES[1] | SRES[0] | ADC Resolution |
|---|---|---|---|
| 0 | 0 | 0 | 8-bit data |
| 0 | 0 | 1 | Reserved[1] |
| 0 | 1 | 0 | 10-bit data |
| 0 | 1 | 1 | Reserved[1] |
| 1 | 0 | 0 | 12-bit data |
| 1 | x | x | Reserved[1] |

1 Reserved settings cause a severe error at ADC conversion start whereby the CMD_EIF flag is set and ADC ceases operation

### 10.4.2.6 ADC Conversion Flow Control Register (ADCFLWCTL)

Bit set and bit clear instructions should not be used to access this register.

When the ADC is enabled the bits of ADCFLWCTL register can be modified after a latency time of three Bus Clock cycles.
All bits are cleared if bit ADC_EN is clear or via ADC soft-reset.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SEQA | TRIG | RSTA | LDOK | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-9. ADC Conversion Flow Control Register (ADCFLWCTL)**

Read: Anytime

Write:

- Bits SEQA, TRIG, RSTA, LDOK can only be set if bit ADC_EN is set.
- Writing 1’b0 to any of these bits does not have an effect

Timing considerations (Trigger Event - channel sample start) depending on ADC mode configuration:

- Restart Mode
  When the Restart Event has been processed (initial command of current CSL is loaded) it takes two Bus Clock cycles plus two ADC conversion clock cycles (pump phase) from the Trigger Event (bit TRIG set) until the select channel starts to sample.
  During a conversion sequence (back to back conversions) it takes five Bus Clock cycles plus two ADC conversion clock cycles (pump phase) from current conversion period end until the newly selected channel is sampled in the following conversion period.
- Trigger Mode
  When a Restart Event occurs a Trigger Event is issued simultaneously. The time required to process the Restart Event is mainly defined by the internal read data bus availability and therefore can vary. In this mode the Trigger Event is processed immediately after the Restart Event is finished and both conversion flow control bits are cleared simultaneously. From de-assert of bit TRIG until sampling begins five Bus Clock cycles are required. Hence from occurrence of a Restart Event until channel sampling it takes five Bus Clock cycles plus an uncertainty of a few Bus Clock cycles.

For more details regarding the sample phase please refer to Section 10.5.2.2, “Sample and Hold Machine with Sample Buffer Amplifier.

**Table 10-8. ADCFLWCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SEQA | **Conversion Sequence Abort Event** — This bit indicates that a conversion sequence abort event is in progress. When this bit is set the ongoing conversion sequence and current CSL will be aborted at the next conversion boundary. This bit gets cleared when the ongoing conversion sequence is aborted and ADC is idle.<br>This bit can only be set if bit ADC_EN is set.<br>This bit is cleared if bit ADC_EN is clear.<br>*Data-Bus Control:*<br>This bit can be controlled via the Data Bus if access control is configured accordingly via ACC_CFG[1:0].<br>Writing a value of 1'b0 does not clear the flag.<br>Writing a one to this bit does not clear it but causes an overrun if the bit has already been set. See Section 10.5.3.2.6, "Conversion flow control in case of conversion sequence control bit overrun scenarios for more details.<br>*Internal Interface Control:*<br>This bit can be controlled via the Internal Interface Signal "Seq_Abort" if access control is configured accordingly via ACC_CFG[1:0]. After being set an additional request via the Internal Interface Signal "Seq_Abort" causes an overrun. See also conversion flow control in case of overrun situations.<br>*General:*<br>In both conversion flow control modes (Restart Mode and Trigger Mode) when bit RSTA gets set automatically bit SEQA gets set when the ADC has not reached one of the following scenarios:<br>- A Sequence Abort request is about to be executed or has been executed.<br>- "End Of List" command type has been executed or is about to be executed<br>In case bit SEQA is set automatically the Restart error flag RSTA_EIF is set to indicate an unexpected Restart Request.<br>0 No conversion sequence abort request.<br>1 Conversion sequence abort request. |
| 6<br>TRIG | **Conversion Sequence Trigger Bit** — This bit starts a conversion sequence if set and no conversion or conversion sequence is ongoing. This bit is cleared when the first conversion of a sequence starts to sample.<br>This bit can only be set if bit ADC_EN is set.<br>This bit is cleared if bit ADC_EN is clear.<br>*Data-Bus Control:*<br>This bit can be controlled via the Data Bus if access control is configured accordingly via ACC_CFG[1:0].<br>Writing a value of 1'b0 does not clear the flag.<br>After being set this bit can not be cleared by writing a value of 1'b1 instead the error flag TRIG_EIF is set. See also Section 10.5.3.2.6, "Conversion flow control in case of conversion sequence control bit overrun scenarios for more details.<br>*Internal Interface Control:*<br>This bit can be controlled via the Internal Interface Signal "Trigger" if access control is configured accordingly via ACC_CFG[1:0]. After being set an additional request via Internal Interface Signal "Trigger" causes the flag TRIG_EIF to be set.<br>0 No conversion sequence trigger.<br>1 Trigger to start conversion sequence. |

**Table 10-8. ADCFLWCTL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>RSTA | **Restart Event (Restart from Top of Command Sequence List)** — This bit indicates that a Restart Event is executed. The ADC loads the conversion command from top of the active Sequence Command List when no conversion or conversion sequence is ongoing. This bit is cleared when the first conversion command of the sequence from top of active Sequence Command List has been loaded into the ADCCMD register.<br>This bit can only be set if bit ADC_EN is set.<br>This bit is cleared if bit ADC_EN is clear.<br>*Data-Bus Control:*<br>This bit can be controlled via the Data Bus if access control is configured accordingly via ACC_CFG[1:0].<br>Writing a value of 1'b0 does not clear the flag.<br>Writing a one to this bit does not clear it but causes an overrun if the bit has already been set. See also Section 10.5.3.2.6, "Conversion flow control in case of conversion sequence control bit overrun scenarios for more details.<br>*Internal Interface Control:*<br>This bit can be controlled via the Internal Interface Signal "Restart" if access control is configured accordingly via ACC_CFG[1:0]. After being set an additional request via Internal Interface Signal "Restart" causes an overrun. See conversion flow control in case of overrun situations for more details.<br>General:<br>In conversion flow control mode "Trigger Mode" when bit RSTA gets set bit TRIG is set simultaneously if one of the following has been executed:<br>- "End Of List" command type has been executed or is about to be executed<br>- Sequence Abort Event<br>0 Continue with commands from active Sequence Command List.<br>1 Restart from top of active Sequence Command List. |
| 4<br>LDOK | **Load OK for alternative Command Sequence List** — This bit indicates if the preparation of the alternative Sequence Command List is done and Command Sequence List must be swapped with the Restart Event. This bit is cleared when bit RSTA is set (Restart Event executed) and the Command Sequence List got swapped.<br>This bit can only be set if bit ADC_EN is set.<br>This bit is cleared if bit ADC_EN is clear.<br>This bit is forced to zero if bit CSL_BMOD is clear.<br>*Data-Bus Control:*<br>This bit can be controlled via the Data Bus if access control is configured accordingly via ACC_CFG[1:0].<br>Writing a value of 1'b0 does not clear the flag.<br>To set bit LDOK the bits LDOK and RSTA must be written simultaneously.<br>After being set this bit can not be cleared by writing a value of 1'b1. See also Section 10.5.3.2.6, "Conversion flow control in case of conversion sequence control bit overrun scenarios for more details.<br>*Internal Interface Control:*<br>This bit can be controlled via the Internal Interface Signal "LoadOK" and "Restart" if access control is configured accordingly via ACC_CFG[1:0]. With the assertion of Interface Signal "Restart" the interface Signal "LoadOK" is evaluated and bit LDOK set accordingly (bit LDOK set if Interface Signal "LoadOK" asserted when Interface Signal "Restart" asserts).<br>General:<br>Only in "Restart Mode" if a Restart Event occurs without bit LDOK being set the error flag LDOK_EIF is set except when the respective Restart Request occurred after or simultaneously with a Sequence Abort Request.<br>The LDOK_EIF error flag is also not set in "Restart Mode" if the first Restart Event occurs after:<br>- ADC got enabled<br>- Exit from Stop Mode<br>- ADC Soft-Reset<br>0 Load of alternative list done.<br>1 Load alternative list. |

**Table 10-9. Summary of Conversion Flow Control Bit Scenarios**

| RSTA | TRIG | SEQA | LDOK | Conversion Flow Control Mode | Conversion Flow Control Scenario |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Both Modes | Valid |
| 0 | 0 | 0 | 1 | Both Modes | Can Not Occur |
| 0 | 0 | 1 | 0 | Both Modes | Valid[5] |
| 0 | 0 | 1 | 1 | Both Modes | Can Not Occur |
| 0 | 1 | 0 | 0 | Both Modes | Valid[2] |
| 0 | 1 | 0 | 1 | Both Modes | Can Not Occur |
| 0 | 1 | 1 | 0 | Both Modes | Can Not Occur |
| 0 | 1 | 1 | 1 | Both Modes | Can Not Occur |
| 1 | 0 | 0 | 0 | Both Modes | Valid[4] |
| 1 | 0 | 0 | 1 | Both Modes | Valid[1 4] |
| 1 | 0 | 1 | 0 | Both Modes | Valid[3 4 5] |
| 1 | 0 | 1 | 1 | Both Modes | Valid[1 3 4 5] |
| 1 | 1 | 0 | 0 | “Restart Mode” | Error flag TRIG_EIF set |
| | | | | “Trigger Mode” | Valid[2 4 6] |
| 1 | 1 | 0 | 1 | “Restart Mode” | Error flag TRIG_EIF set |
| | | | | “Trigger Mode” | Valid[1 2 4 6] |
| 1 | 1 | 1 | 0 | “Restart Mode” | Error flag TRIG_EIF set |
| | | | | “Trigger Mode” | Valid[2 3 4 5 6] |
| 1 | 1 | 1 | 1 | “Restart Mode” | Error flag TRIG_EIF set |
| | | | | “Trigger Mode” | Valid[1 2 3 4 5 6] |

1 Swap CSL buffer
2 Start conversion sequence
3 Prevent RSTA_EIF and LDOK_EIF
4 Load conversion command from top of CSL
5 Abort any ongoing conversion, conversion sequence and CSL
6 Bit TRIG set automatically in Trigger Mode

For a detailed description of all conversion flow control bit scenarios please see also Section 10.5.3.2.4, “The two conversion flow control Mode Configurations, Section 10.5.3.2.5, “The four ADC conversion flow control bits and Section 10.5.3.2.6, “Conversion flow control in case of conversion sequence control bit overrun scenarios

### 10.4.2.7 ADC Error Interrupt Enable Register (ADCEIE)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IA_EIE | CMD_EIE | EOL_EIE | Reserved | TRIG_EIE | RSTAR_EIE | LDOK_EIE | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-10. ADC Error Interrupt Enable Register (ADCEIE)**

Read: Anytime

Write: Anytime

**Table 10-10. ADCEIE Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IA_EIE | **Illegal Access Error Interrupt Enable Bit** — This bit enables the illegal access error interrupt.<br>0 Illegal access error interrupt disabled.<br>1 Illegal access error interrupt enabled. |
| 6<br>CMD_EIE | **Command Value Error Interrupt Enable Bit** — This bit enables the command value error interrupt.<br>0 Command value interrupt disabled.<br>1 Command value interrupt enabled. |
| 5<br>EOL_EIE | **"End Of List" Error Interrupt Enable Bit** — This bit enables the "End Of List" error interrupt.<br>0 "End Of List" error interrupt disabled.<br>1 "End Of List" error interrupt enabled. |
| 3<br>TRIG_EIE | **Conversion Sequence Trigger Error Interrupt Enable Bit** — This bit enables the conversion sequence trigger error interrupt.<br>0 Conversion sequence trigger error interrupt disabled.<br>1 Conversion sequence trigger error interrupt enabled. |
| 2<br>RSTAR_EIE | **Restart Request Error Interrupt Enable Bit**— This bit enables the restart request error interrupt.<br>0 Restart Request error interrupt disabled.<br>1 Restart Request error interrupt enabled. |
| 1<br>LDOK_EIE | **Load OK Error Interrupt Enable Bit** — This bit enables the Load OK error interrupt.<br>0 Load OK error interrupt disabled.<br>1 Load OK error interrupt enabled. |

### 10.4.2.8 ADC Interrupt Enable Register (ADCIE)

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SEQAD_IE | CONIF_OIE | Reserved | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-11. ADC Interrupt Enable Register (ADCIE)**

Read: Anytime

Write: Anytime

**Table 10-11. ADCIE Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SEQAD_IE | **Conversion Sequence Abort Done Interrupt Enable Bit** — This bit enables the conversion sequence abort event done interrupt.<br>0 Conversion sequence abort event done interrupt disabled.<br>1 Conversion sequence abort event done interrupt enabled. |
| 6<br>CONIF_OIE | **ADCCONIF Register Flags Overrun Interrupt Enable** — This bit enables the flag which indicates if an overrun situation occurred for one of the CON_IF[15:1] flags or for the EOL_IF flag.<br>0 No ADCCONIF Register Flag overrun occurred.<br>1 ADCCONIF Register Flag overrun occurred. |

### 10.4.2.9 ADC Error Interrupt Flag Register (ADCEIF)

If one of the following error flags is set the ADC ceases operation:
- IA_EIF
- CMD_EIF
- EOL_EIF
- TRIG_EIF

In order to make the ADC operational again an ADC Soft-Reset must be issued which clears above listed error interrupt flags.

The error interrupt flags RSTAR_EIF and LDOK_EIF do not cause the ADC to cease operation. If set the ADC continues operation. Each of the two bits can be cleared by writing a value of 1’b1. Both bits are also cleared if an ADC Soft-Reset is issued.

All bits are cleared if bit ADC_EN is clear. Writing any flag with value 1’b0 does not clear a flag. Writing any flag with value 1’b1 does not set the flag.

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IA_EIF | CMD_EIF | EOL_EIF | Reserved | TRIG_EIF | RSTAR_EIF | LDOK_EIF | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-12. ADC Error Interrupt Flag Register (ADCEIF)**

Read: Anytime

Write:

- Bits RSTAR_EIF and LDOK_EIF are writable anytime
- Bits IA_EIF, CMD_EIF, EOL_EIF and TRIG_EIF are not writable

**Table 10-12. ADCEIF Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IA_EIF | **Illegal Access Error Interrupt Flag** — This flag indicates that storing the conversion result caused an illegal access error or conversion command loading from outside system RAM or NVM area occurred.<br>The ADC ceases operation if this error flag is set (issue of type severe).<br>0 No illegal access error occurred.<br>1 An illegal access error occurred. |
| 6<br>CMD_EIF | **Command Value Error Interrupt Flag** — This flag indicates that an invalid command is loaded (Any command that contains reserved bit settings) or illegal format setting selected (reserved SRES[2:0] bit settings).<br>The ADC ceases operation if this error flag is set (issue of type severe).<br>0 Valid conversion command loaded.<br>1 Invalid conversion command loaded. |
| 5<br>EOL_EIF | **“End Of List” Error Interrupt Flag** — This flag indicates a missing “End Of List” command type in current executed CSL.<br>The ADC ceases operation if this error flag is set (issue of type severe).<br>0 No “End Of List” error.<br>1 “End Of List” command type missing in current executed CSL. |

**Table 10-12. ADCEIF Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>TRIG_EIF | **Trigger Error Interrupt Flag** — This flag indicates that a trigger error occurred.<br>This flag is set in “Restart” Mode when a conversion sequence got aborted and no Restart Event occurred before the Trigger Event or if the Trigger Event occurred before the Restart Event was finished (conversion command has been loaded).<br>This flag is set in “Trigger” Mode when a Trigger Event occurs before the Restart Event is issued to start conversion of the initial Command Sequence List. In “Trigger” Mode only a Restart Event is required to start conversion of the initial Command Sequence List.<br>This flag is set when a Trigger Event occurs before a conversion sequence got finished.<br>This flag is also set if a Trigger occurs while a Trigger Event is just processed - first conversion command of a sequence is beginning to sample (see also Section 10.5.3.2.6, “Conversion flow control in case of conversion sequence control bit overrun scenarios).<br>This flag is also set if the Trigger Event occurs automatically generated by hardware in “Trigger Mode” due to a Restart Event and simultaneously a Trigger Event is generated via Data Bus or Internal Interface.<br>The ADC ceases operation if this error flag is set (issue of type severe).<br>0 No trigger error occurred.<br>1 A trigger error occurred. |
| 2<br>RSTAR_EIF | **Restart Request Error Interrupt Flag** — This flag indicates a flow control issue. It is set when a Restart Request occurs after a Trigger Event and before one of the following conditions was reached:<br>- The “End Of List” command type has been executed<br>- Depending on bit STR_SEQA if the “End Of List” command type is about to be executed<br>- The current CSL has been aborted or is about to be aborted due to a Sequence Abort Request.<br>The ADC continues operation if this error flag is set.<br>This flag is not set for Restart Request overrun scenarios (see also Section 10.5.3.2.6, “Conversion flow control in case of conversion sequence control bit overrun scenarios).<br>0 No Restart request error situation occurred.<br>1 Restart request error situation occurred. |
| 1<br>LDOK_EIF | **Load OK Error Interrupt Flag** — This flag can only be set in “Restart Mode” . It indicates that a Restart Request occurred without LDOK. This flag is not set if a Sequence Abort Event is already in process (bit SEQA set) when the Restart Request occurs or a Sequence Abort Request occurs simultaneously with the Restart Request.<br>The LDOK_EIF error flag is also not set in “Restart Mode” if the first Restart Event occurs after:<br>- ADC got enabled<br>- Exit from Stop Mode<br>- ADC Soft-Reset<br>- ADC used in CSL single buffer mode<br>The ADC continues operation if this error flag is set.<br>0 No Load OK error situation occurred.<br>1 Load OK error situation occurred. |

### 10.4.2.10 ADC Interrupt Flag Register (ADCIF)

After being set any of these bits can be cleared by writing a value of 1’b1 or via ADC soft-reset (bit ADC_SR). All bits are cleared if bit ADC_EN is clear. Writing any flag with value 1’b0 does not clear the flag. Writing any flag with value 1’b1 does not set the flag.

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SEQAD_IF | CONIF_OIF | Reserved | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-13. ADC Interrupt Flag Register (ADCIF)**

Read: Anytime

Write: Anytime

**Table 10-13. ADCIF Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SEQAD_IF | **Conversion Sequence Abort Done Interrupt Flag** — This flag is set when the Sequence Abort Event has been executed except the Sequence Abort Event occurred by hardware in order to be able to enter MCU Stop Mode or Wait Mode with bit SWAI set.This flag is also not set if the Sequence Abort request occurs during execution of the last conversion command of a CSL and bit STR_SEQA being set.<br>0 No conversion sequence abort request occurred.<br>1 A conversion sequence abort request occurred. |
| 6<br>CONIF_OIF | **ADCCONIF Register Flags Overrun Interrupt Flag** — This flag indicates if an overrun situation occurred for one of the CON_IF[15:1] flags or for the EOL_IF flag. In RVL single buffer mode (RVL_BMOD clear) an overrun of the EOL_IF flag is not indicated (For more information please see Note below).<br>0 No ADCCONIF Register Flag overrun occurred.<br>1 ADCCONIF Register Flag overrun occurred. |

**NOTE**

In RVL double buffer mode a conversion interrupt flag (CON_IF[15:1]) or End Of List interrupt flag (EOL_IF) overrun is detected if one of these bits is set when it should be set again due to conversion command execution.

In RVL single buffer mode a conversion interrupt flag (CON_IF[15:1]) overrun is detected only. The overrun is detected if any of the conversion interrupt flags (CON_IF[15:1]) is set while the first conversion result of a CSL is stored (result of first conversion from top of CSL is stored).

### 10.4.2.11 ADC Conversion Interrupt Enable Register (ADCCONIE)

Module Base + 0x000A

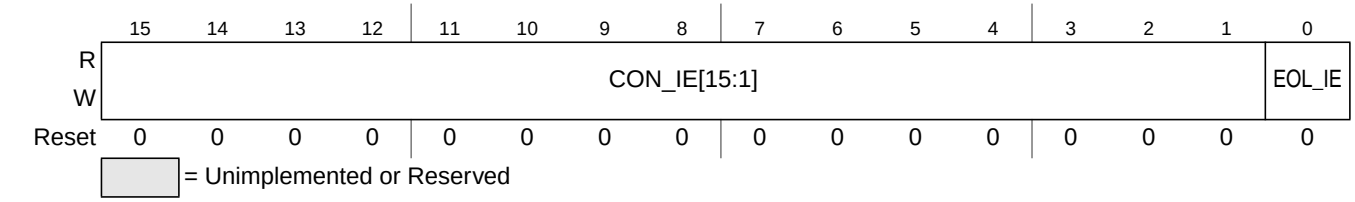



**Figure 10-14. ADC Conversion Interrupt Enable Register (ADCCONIE)**

Read: Anytime

Write: Anytime

**Table 10-14. ADCCONIE Field Descriptions**

| Field | Description |
|---|---|
| 15-1<br>CON_IE[15:1] | **Conversion Interrupt Enable Bits** — These bits enable the individual interrupts which can be triggered via interrupt flags CON_IF[15:1].<br>0 ADC conversion interrupt disabled.<br>1 ADC conversion interrupt enabled. |
| 0<br>EOL_IE | **End Of List Interrupt Enable Bit** — This bit enables the end of conversion sequence list interrupt.<br>0 End of list interrupt disabled.<br>1 End of list interrupt enabled. |

### 10.4.2.12 ADC Conversion Interrupt Flag Register (ADCCONIF)

After being set any of these bits can be cleared by writing a value of 1’b1. All bits are cleared if bit ADC_EN is clear or via ADC soft-reset (bit ADC_SR set). Writing any flag with value 1’b0 does not clear the flag. Writing any flag with value 1’b1 does not set the flag.

Module Base + 0x000C

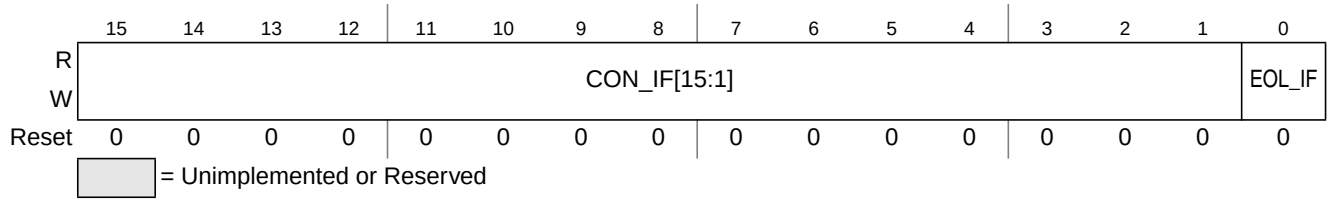



**Figure 10-15. ADC Conversion Interrupt Flag Register (ADCCONIF)**

Read: Anytime

Write: Anytime

**Table 10-15. ADCCONIF Field Descriptions**

| Field | Description |
|---|---|
| 15-1<br>CON_IF[15:1] | **Conversion Interrupt Flags** — These bits could be set by the binary coded interrupt select bits INT_SEL[3:0] when the corresponding conversion command has been processed and related data has been stored to RAM. See also notes below. |
| 0<br>EOL_IF | **End Of List Interrupt Flag** — This bit is set by the binary coded conversion command type select bits CMD_SEL[1:0] for “end of list” type of commands and after such a command has been processed and the related data has been stored RAM.<br>See also second note below |

**NOTE**

These bits can be used to indicate if a certain packet of conversion results is available. Clearing a flag indicates that conversion results have been retrieved by software and the flag can be used again (see also Section 10.8.6, “RVL swapping in RVL double buffer mode and related registers ADCIMDRI and ADCEOLRI.

**NOTE**

Overrun situation of a flag CON_IF[15:1] and EOL_IF are indicated by flag CONIF_OIF.

### 10.4.2.13 ADC Intermediate Result Information Register (ADCIMDRI)

This register is cleared when bit ADC_SR is set or bit ADC_EN is clear.

Module Base + 0x000E

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CSL_IMD | RVL_IMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RIDX_IMD[5:0] | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-16. ADC Intermediate Result Information Register (ADCIMDRI)**

Read: Anytime

Write: Never

**Table 10-16. ADCIMDRI Field Descriptions**

| Field | Description |
|---|---|
| 15<br>CSL_IMD | **Active CSL At Intermediate Event** — This bit indicates the active (used) CSL at the occurrence of a conversion interrupt flag (CON_IF[15:1]) (occurrence of an intermediate result buffer fill event) or when a Sequence Abort Event gets executed.<br>0 CSL_0 active (used) when a conversion interrupt flag (CON_IF[15:1]) got set.<br>1 CSL_1 active (used) when a conversion interrupt flag (CON_IF[15:1]) got set. |
| 14<br>RVL_IMD | **Active RVL At Intermediate Event** — This bit indicates the active (used) RVL buffer at the occurrence of a conversion interrupt flag (CON_IF[15:1]) (occurrence of an intermediate result buffer fill event) or when a Sequence Abort Event gets executed.<br>0 RVL_0 active (used) when a conversion interrupt flag (CON_IF[15:1]) got set.<br>1 RVL_1 active (used) when a conversion interrupt flag (CON_IF[15:1]) got set. |
| 5-0<br>RIDX_IMD[5:0] | **RES_IDX Value At Intermediate Event** — These bits indicate the result index (RES_IDX) value at the occurrence of a conversion interrupt flag (CON_IF[15:1]) (occurrence of an intermediate result buffer fill event) or occurrence of EOL_IF flag or when a Sequence Abort Event gets executed to abort an ongoing conversion (the result index RES_IDX is captured at the occurence of a result data store).<br><br>When a Sequence Abort Event has been processed flag SEQAD_IF is set and the RES_IDX value of the last stored result is provided. Hence in case an ongoing conversion is aborted the RES_IDX value captured in RIDX_IMD bits depends on bit STORE_SEQA:<br>- STORE_SEQA =1 : The result index of the aborted conversion is provided<br>- STORE_SEQA =0 : The result index of the last stored result at abort execution time is provided<br>In case a CSL is aborted while no conversion is ongoing (ADC waiting for a Trigger Event) the last captured result index is provided.<br>In case a Sequence Abort Event was initiated by hardware due to MCU entering Stop Mode or Wait Mode with bit SWAI set, the result index of the last stored result is captured by bits RIDX_IMD but flag SEQAD_IF is not set. |

**NOTE**

The register ADCIMDRI is updated and simultaneously a conversion interrupt flag CON_IF[15:1] occurs when the corresponding conversion command (conversion command with INTFLG_SEL[3:0] set) has been processed and related data has been stored to RAM.

### 10.4.2.14 ADC End Of List Result Information Register (ADCEOLRI)

This register is cleared when bit ADC_SR is set or bit ADC_EN is clear.

Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CSL_EOL | RVL_EOL | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-17. ADC End Of List Result Information Register (ADCEOLRI)**

Read: Anytime

Write: Never

**Table 10-17. ADCEOLRI Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CSL_EOL | **Active CSL When “End Of List” Command Type Executed** — This bit indicates the active (used) CSL when a “End Of List” command type has been executed and related data has been stored to RAM.<br>0 CSL_0 active when “End Of List” command type executed.<br>1 CSL_1 active when “End Of List” command type executed. |
| 6<br>RVL_EOL | **Active RVL When “End Of List” Command Type Executed** — This bit indicates the active (used) RVL when a “End Of List” command type has been executed and related data has been stored to RAM.<br>0 RVL_0 active when “End Of List” command type executed.<br>1 RVL_1 active when “End Of List” command type executed. |

**NOTE**

The conversion interrupt EOL_IF occurs and simultaneously the register ADCEOLRI is updated when the “End Of List” conversion command type has been processed and related data has been stored to RAM.

### 10.4.2.15 ADC Command Register 0 (ADCCMD_0)

Module Base + 0x0014

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| R | CMD_SEL | | 0 | 0 | INTFLG_SEL[3:0] | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-18. ADC Command Register 0 (ADCCMD_0)**

Read: Anytime

Write: Only writable if bit SMOD_ACC is set
(see also Section 10.4.2.2, “ADC Control Register 1 (ADCCTL_1) bit SMOD_ACC description for more details)

**Table 10-18. ADCCMD_0 Field Descriptions**

| Field | Description |
|---|---|
| 31-30 CMD_SEL[1:0] | **Conversion Command Select Bits** — These bits define the type of current conversion described in Table 10-19. |
| 27-24 INTFLG_SEL[3:0] | **Conversion Interrupt Flag Select Bits** — These bits define which interrupt flag is set in the ADCIFH/L register at the end of current conversion.The interrupt flags ADCIF[15:1] are selected via binary coded bits INT_SEL[3:0]. See also Table 10-20 |

**NOTE**

If bit SMOD_ACC is set modifying this register must be done carefully - only when no conversion and conversion sequence is ongoing.

**Table 10-19. Conversion Command Type Select**

| CMD_SEL[1] | CMD_SEL[0] | Conversion Command Type Description |
|---|---|---|
| 0 | 0 | Normal Conversion |
| 0 | 1 | End Of Sequence<br>(Wait for Trigger to execute next sequence or for a Restart) |
| 1 | 0 | End Of List<br>(Automatic wrap to top of CSL<br>and Continue Conversion) |
| 1 | 1 | End Of List<br>(Wrap to top of CSL and:<br>- In “Restart Mode” wait for Restart Event followed by a Trigger<br>- In “Trigger Mode” wait for Trigger or Restart Event) |

**Table 10-20. Conversion Interrupt Flag Select**

| CON_IF[15:1] | INTFLG_SEL[3] | INTFLG_SEL[2] | INTFLG_SEL[1] | INTFLG_SEL[0] | Comment |
|---|---|---|---|---|---|
| 0x0000 | 0 | 0 | 0 | 0 | No flag set |
| 0x0001 | 0 | 0 | 0 | 1 | Only one flag can be set (one hot coding) |
| 0x0002 | 0 | 0 | 1 | 0 | |
| 0x0004 | 0 | 0 | 1 | 1 | |
| 0x0008 | 0 | 1 | 0 | 0 | |
| 0x0010 | 0 | 1 | 0 | 1 | |
| .... | ... | ... | ... | ... | |
| 0x0800 | 1 | 1 | 0 | 0 | |
| 0x1000 | 1 | 1 | 0 | 1 | |
| 0x2000 | 1 | 1 | 1 | 0 | |
| 0x4000 | 1 | 1 | 1 | 1 | |

### 10.4.2.16 ADC Command Register 1 (ADCCMD_1)

A command which contains reserved bit settings causes the error flag CMD_EIF being set and ADC cease operation.

Module Base + 0x0015

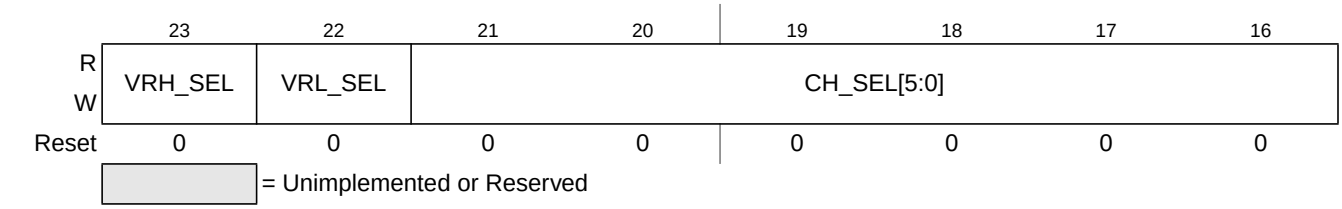



**Figure 10-19. ADC Command Register 1 (ADCCMD_1)**

Read: Anytime

Write: Only writable if bit SMOD_ACC is set
(see also Section 10.4.2.2, “ADC Control Register 1 (ADCCTL_1) bit SMOD_ACC description for more details)

**Table 10-21. ADCCMD_1 Field Descriptions**

| Field | Description |
|---|---|
| 23<br>VRH_SEL | **Reference High Voltage Select Bit** — This bit selects the high voltage reference for current conversion.<br>0 VRH_0 input selected as high voltage reference.<br>1 VRH_1 input selected as high voltage reference. |
| 22<br>VRL_SEL | **Reference Low Voltage Select Bit** — This bit selects the voltage reference for current conversion.<br>0 VRL_0 input selected as low voltage reference.<br>1 VRL_1 input selected as low voltage reference. |
| 21-16<br>CH_SEL[5:0] | **ADC Input Channel Select Bits** — These bits select the input channel for the current conversion. See Table 10-22 for channel coding information. |

**NOTE**

If bit SMOD_ACC is set modifying this register must be done carefully - only when no conversion and conversion sequence is ongoing.

**Table 10-22. Analog Input Channel Select**

| CH_SEL[5] | CH_SEL[4] | CH_SEL[3] | CH_SEL[2] | CH_SEL[1] | CH_SEL[0] | Analog Input Channel |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | VRL_0/1 |
| 0 | 0 | 0 | 0 | 0 | 1 | VRH_0/1 |
| 0 | 0 | 0 | 0 | 1 | 0 | (VRH_0/1 + VRL_0/1) / 2 |
| 0 | 0 | 0 | 0 | 1 | 1 | Reserved |
| 0 | 0 | 0 | 1 | 0 | 0 | Reserved |
| 0 | 0 | 0 | 1 | 0 | 1 | Reserved |
| 0 | 0 | 0 | 1 | 1 | 0 | Reserved |

**Table 10-22. Analog Input Channel Select**

| CH_SEL[5] | CH_SEL[4] | CH_SEL[3] | CH_SEL[2] | CH_SEL[1] | CH_SEL[0] | Analog Input Channel |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | Reserved |
| 0 | 0 | 1 | 0 | 0 | 0 | Internal_0<br>(ADC temperature sense) |
| 0 | 0 | 1 | 0 | 0 | 1 | Internal_1<br>(Vreg_3v3 sense) |
| 0 | 0 | 1 | 0 | 1 | 0 | Internal_2 |
| 0 | 0 | 1 | 0 | 1 | 1 | Internal_3 |
| 0 | 0 | 1 | 1 | 0 | 0 | Internal_4 |
| 0 | 0 | 1 | 1 | 0 | 1 | Internal_5 |
| 0 | 0 | 1 | 1 | 1 | 0 | Internal_6 |
| 0 | 0 | 1 | 1 | 1 | 1 | Internal_7 |
| 0 | 1 | 0 | 0 | 0 | 0 | AN0 |
| 0 | 1 | 0 | 0 | 0 | 1 | AN1 |
| 0 | 1 | 0 | 0 | 1 | 0 | AN2 |
| 0 | 1 | 0 | 0 | 1 | 1 | AN3 |
| 0 | 1 | 0 | 1 | 0 | 0 | AN4 |
| 0 | 1 | x | x | x | x | ANx |
| 1 | x | x | x | x | x | Reserved |

**NOTE**

ANx in Table 10-22 is the maximum number of implemented analog input channels on the device. Please refer to the device overview of the Reference Manual for details regarding number of analog input channels.

### 10.4.2.17 ADC Command Register 2 (ADCCMD_2)

A command which contains reserved bit settings causes the error flag CMD_EIF being set and ADC cease operation.

Module Base + 0x0016

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | SMP[4:0] | | | | | 0 | 0 | Reserved |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-20. ADC Command Register 2 (ADCCMD_2)**

Read: Anytime

Write: Only writable if bit SMOD_ACC is set
(see also Section 10.4.2.2, “ADC Control Register 1 (ADCCTL_1) bit SMOD_ACC description for more details)

**Table 10-23. ADCCMD_2 Field Descriptions**

| Field | Description |
|---|---|
| 15-11 SMP[4:0] | **Sample Time Select Bits** — These four bits select the length of the sample time in units of ADC conversion clock cycles. Note that the ADC conversion clock period is itself a function of the prescaler value (bits PRS[6:0]). Table 10-24 lists the available sample time lengths. |

**NOTE**

If bit SMOD_ACC is set modifying this register must be done carefully - only when no conversion and conversion sequence is ongoing.

**Table 10-24. Sample Time Select**

| SMP[4] | SMP[3] | SMP[2] | SMP[1] | SMP[0] | Sample Time in Number of ADC Clock Cycles |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 4 |
| 0 | 0 | 0 | 0 | 1 | 5 |
| 0 | 0 | 0 | 1 | 0 | 6 |
| 0 | 0 | 0 | 1 | 1 | 7 |
| 0 | 0 | 1 | 0 | 0 | 8 |
| 0 | 0 | 1 | 0 | 1 | 9 |
| 0 | 0 | 1 | 1 | 0 | 10 |
| 0 | 0 | 1 | 1 | 1 | 11 |
| 0 | 1 | 0 | 0 | 0 | 12 |
| 0 | 1 | 0 | 0 | 1 | 13 |

**Table 10-24. Sample Time Select**

| SMP[4] | SMP[3] | SMP[2] | SMP[1] | SMP[0] | Sample Time in Number of ADC Clock Cycles |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 14 |
| 0 | 1 | 0 | 1 | 1 | 15 |
| 0 | 1 | 1 | 0 | 0 | 16 |
| 0 | 1 | 1 | 0 | 1 | 17 |
| 0 | 1 | 1 | 1 | 0 | 18 |
| 0 | 1 | 1 | 1 | 1 | 19 |
| 1 | 0 | 0 | 0 | 0 | 20 |
| 1 | 0 | 0 | 0 | 1 | 21 |
| 1 | 0 | 0 | 1 | 0 | 22 |
| 1 | 0 | 0 | 1 | 1 | 23 |
| 1 | 0 | 1 | 0 | 0 | 24 |
| 1 | 0 | 1 | 0 | 1 | Reserved |
| 1 | 0 | 1 | 1 | 0 | Reserved |
| 1 | 0 | 1 | 1 | 1 | Reserved |
| 1 | 1 | x | x | x | Reserved |

## 10.4.2.18 ADC Command Register 3 (ADCCMD_3)

Module Base + 0x0017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-21. ADC Command Register 3 (ADCCMD_3)**

### 10.4.2.19 ADC Command Index Register (ADCCIDX)

It is important to note that these bits do not represent absolute addresses instead it is a sample index (object size 32bit).

Module Base + 0x001C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | CMD_IDX[5:0] | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-22. ADC Command Index Register (ADCCIDX)**

Read: Anytime

Write: NA

**Table 10-25. ADCCIDX Field Descriptions**

| Field | Description |
|---|---|
| 5-0<br>CMD_IDX<br>[5:0] | **ADC Command Index Bits** — These bits represent the command index value for the conversion commands relative to the two CSL start addresses in the memory map. These bits do not represent absolute addresses instead it is a sample index (object size 32bit). See also Section 10.5.3.2.2, “Introduction of the two Command Sequence Lists (CSLs) for more details. |

### 10.4.2.20 ADC Command Base Pointer Register (ADCCBP)

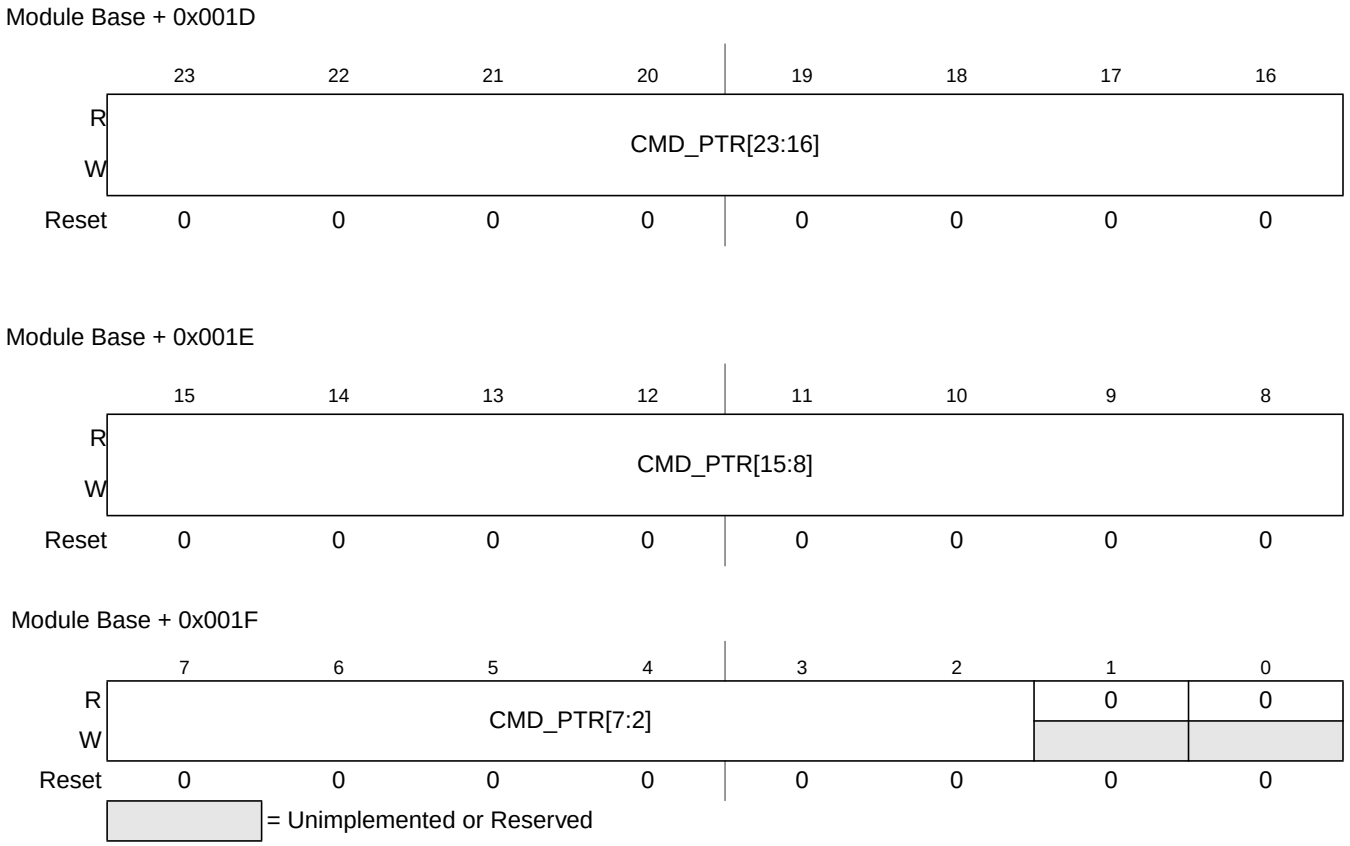



**Figure 10-23. ADC Command Base Pointer Registers (ADCCBP_0, ADCCBP_1, ADCCBP_2))**

Read: Anytime

Write: Bits CMD_PTR[23:2] writable if bit ADC_EN clear or bit SMOD_ACC set

**Table 10-26. ADCCBP Field Descriptions**

| Field | Description |
|---|---|
| 23-2<br>CMD_PTR [23:2] | **ADC Command Base Pointer Address** — These bits define the base address of the two CSL areas inside the system RAM or NVM of the memory map. They are used to calculate the final address from which the conversion commands will be loaded depending on which list is active. For more details see Section 10.5.3.2.2, “Introduction of the two Command Sequence Lists (CSLs). |

### 10.4.2.21 ADC Result Index Register (ADCRIDX)

It is important to note that these bits do not represent absolute addresses instead it is a sample index (object size 16bit).

Module Base + 0x0020

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | RES_IDX[5:0] | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-24. ADC Result Index Register (ADCRIDX)**

Read: Anytime

Write: NA

**Table 10-27. ADCRIDX Field Descriptions**

| Field | Description |
|---|---|
| 5-0<br>RES_IDX[5:0] | **ADC Result Index Bits** — These read only bits represent the index value for the conversion results relative to the two RVL start addresses in the memory map. These bits do not represent absolute addresses instead it is a sample index (object size 16bit). See also Section 10.5.3.2.3, “Introduction of the two Result Value Lists (RVLs)” for more details. |

### 10.4.2.22 ADC Result Base Pointer Register (ADCRBP)

Module Base + 0x0021

| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | RES_PTR[19:16] | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Module Base + 0x0022

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | RES_PTR[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Module Base + 0x0023

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RES_PTR[7:2] | | | | | | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-25. ADC Result Base Pointer Registers (ADCRBP_0, ADCRBP_1, ADCRBP_2))**

Read: Anytime

Write: Bits RES_PTR[19:2] writeable if bit ADC_EN clear or bit SMOD_ACC set

**Table 10-28. ADCRBP Field Descriptions**

| Field | Description |
|---|---|
| 19-2<br>RES_PTR[19:2] | **ADC Result Base Pointer Address** — These bits define the base address of the list areas inside the system RAM of the memory map to which conversion results will be stored to at the end of a conversion. These bits can only be written if bit ADC_EN is clear. See also Section 10.5.3.2.3, “Introduction of the two Result Value Lists (RVLs). |

### 10.4.2.23 ADC Command and Result Offset Register 0 (ADCCROFF0)

Module Base + 0x0024

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | CMDRES_OFF0[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-26. ADC Command and Result Offset Register 0 (ADCCROFF0)**

Read: Anytime

Write: NA

**Table 10-29. ADCCROFF0 Field Descriptions**

| Field | Description |
|---|---|
| 6-0<br>CMDRES_OFF0[6:0] | **ADC Command and Result Offset Value** — These read only bits represent the conversion command and result offset value relative to the conversion command base pointer address and result base pointer address in the memory map to refer to CSL_0 and RVL_0. It is used to calculate the address inside the system RAM to which the result at the end of the current conversion is stored to and the area (RAM or NVM) from which the conversion commands are loaded from. This is a zero offset (null offset) which can not be modified. These bits do not represent absolute addresses instead it is a sample offset (object size 16bit for RVL, object size 32bit for CSL). See also Section 10.5.3.2.2, “Introduction of the two Command Sequence Lists (CSLs) and Section 10.5.3.2.3, “Introduction of the two Result Value Lists (RVLs) for more details. |

### 10.4.2.24 ADC Command and Result Offset Register 1 (ADCCROFF1)

It is important to note that these bits do not represent absolute addresses instead it is an sample offset (object size 16bit for RVL, object size 32bit for CSL).

Module Base + 0x0025

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | CMDRES_OFF1[6:0] | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-27. ADC Command and Result Offset Register 1 (ADCCROFF1)**

Read: Anytime

Write: These bits are writable if bit ADC_EN clear or bit SMOD_ACC set

**Table 10-30. ADCCROFF1 Field Descriptions**

| Field | Description |
|---|---|
| 6-0<br>CMDRES_OFF1[6:0] | **ADC Result Address Offset Value** — These bits represent the conversion command and result offset value relative to the conversion command base pointer address and result base pointer address in the memory map to refer to CSL_1 and RVL_1. It is used to calculate the address inside the system RAM to which the result at the end of the current conversion is stored to and the area (RAM or NVM) from which the conversion commands are loaded from. These bits do not represent absolute addresses instead it is an sample offset (object size 16bit for RVL, object size 32bit for CSL).,These bits can only be modified if bit ADC_EN is clear. See also Section 10.5.3.2.2, “Introduction of the two Command Sequence Lists (CSLs)” and Section 10.5.3.2.3, “Introduction of the two Result Value Lists (RVLs)” for more details. |

# 10.5 Functional Description

## 10.5.1 Overview

The ADC12B_LBA consists of an analog sub-block and a digital sub-block. It is a successive approximation analog-to-digital converter including a Sample-and Hold mechanism and an internal charge scaled C-DAC (switched capacitor scaled digital-to-analog converter) with a comparator to realize the successive approximation algorithm.

## 10.5.2 Analog Sub-Block

The analog sub-block contains all analog circuits (Sample&Hold, C-DAC, analog Comparator, etc.) required to perform a single conversion. Separate power supplies VDDA and VSSA allow to isolate noise of other MCU circuitry from the analog sub-block for improved accuracy.

### 10.5.2.1 Analog Input Multiplexer

The analog input multiplexers connect one of the external or internal analog input channels to the sample and hold storage node.

### 10.5.2.2 Sample and Hold Machine with Sample Buffer Amplifier

The Sample and Hold Machine controls the storage and charge of the storage node (sample capacitor) to the voltage level of the analog signal at the selected ADC input channel. This architecture employs the advantage of reduced crosstalk between channels.

The sample buffer amplifier is used to raise the effective input impedance of the A/D machine, so that external components (higher bandwidth or higher impedance connected as specified) are less significant to accuracy degradation.

During the sample phase the analog input connects first via a sample buffer amplifier with the storage node always for two ADC clock cycles (“Buffer” Sample Time). The remaining sample time (“Final” sample time) the storage node is directly connected to the analog input source. Please see also Figure 10-28 for illustration and the Appendix of the device Reference Manual for more details.
The input analog signals are unipolar and must be within the potential range of VSSA to VDDA.
During the hold process the analog input is disconnected from the storage node.

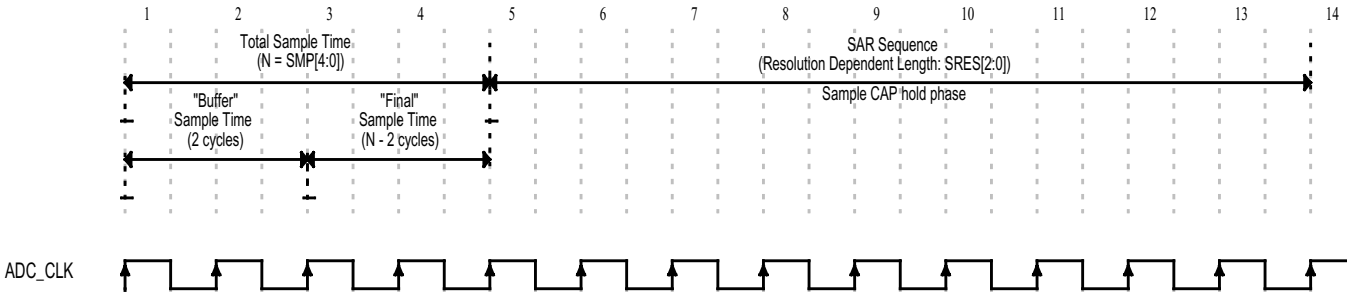



**Figure 10-28. Sampling and Conversion Timing Example (8-bit Resolution, 4 Cycle Sampling)**

Please note that there is always a pump phase of two ADC_CLK cycles before the sample phase begins, hence glitches during the pump phase could impact the conversion accuracy for short sample times.

## 10.5.3 Digital Sub-Block

The digital sub-block contains a list based programmers model and the control logic for the analog sub-block circuits.

### 10.5.3.1 Analog-to-Digital (A/D) Machine

The A/D Machine performs the analog to digital conversion. The resolution is program selectable to be either 8- or 10- or 12 bits. The A/D machine uses a successive approximation architecture. It functions by comparing the sampled and stored analog voltage with a series of binary coded discrete voltages.

By following a binary search algorithm, the A/D machine identifies the discrete voltage that is nearest to the sampled and stored voltage.

Only analog input signals within the potential range of VRL_0/1 to VRH_0/1 (A/D reference potentials) will result in a non-railed digital output code.

### 10.5.3.2 Introduction of the Programmers Model

The ADC_LBA provides a programmers model which uses a system memory list based architecture for definition of the conversion command sequence and conversion result handling.
The Command Sequence List (CSL) and Result Value List (RVL) are implemented in double buffered manner and the buffer mode is user selectable for each list (bits CSL_BMOD, RVL_BMOD). The 32-bit wide conversion command is double buffered and the currently active command is visible in the ADC register map at ADCCMD register space.

#### 10.5.3.2.1 Introduction of The Command Sequence List (CSL) Format

A Command Sequence List (CSL) contains up to 64 conversion commands maximum. A user selectable number of successive conversion commands in the CSL can be grouped as a command sequence. This sequence of conversion commands is successively executed by the ADC at the occurrence of a Trigger Event. The commands of a sequence are successively executed until an “End Of Sequence” or “End Of List” command type identifier in a command is detected (command type is coded via bits CMD_SEL[1:0]). The number of successive conversion commands which belong to a command sequence and the number of command sequences inside the CSL can be freely defined by the user and is limited by the 64 conversion commands a CSL can contain. A CSL must contain at least one conversion command and one “end of list” command type identifier. The minimum number of command sequences inside a CSL is zero and the maximum number of command sequences is 63. A command sequence is defined with bits CMD_SEL[1:0] in the register ADCCMD_M by defining the end of a conversion sequence. The Figure 10-29 and Figure 10-30 provides examples of a CSL.

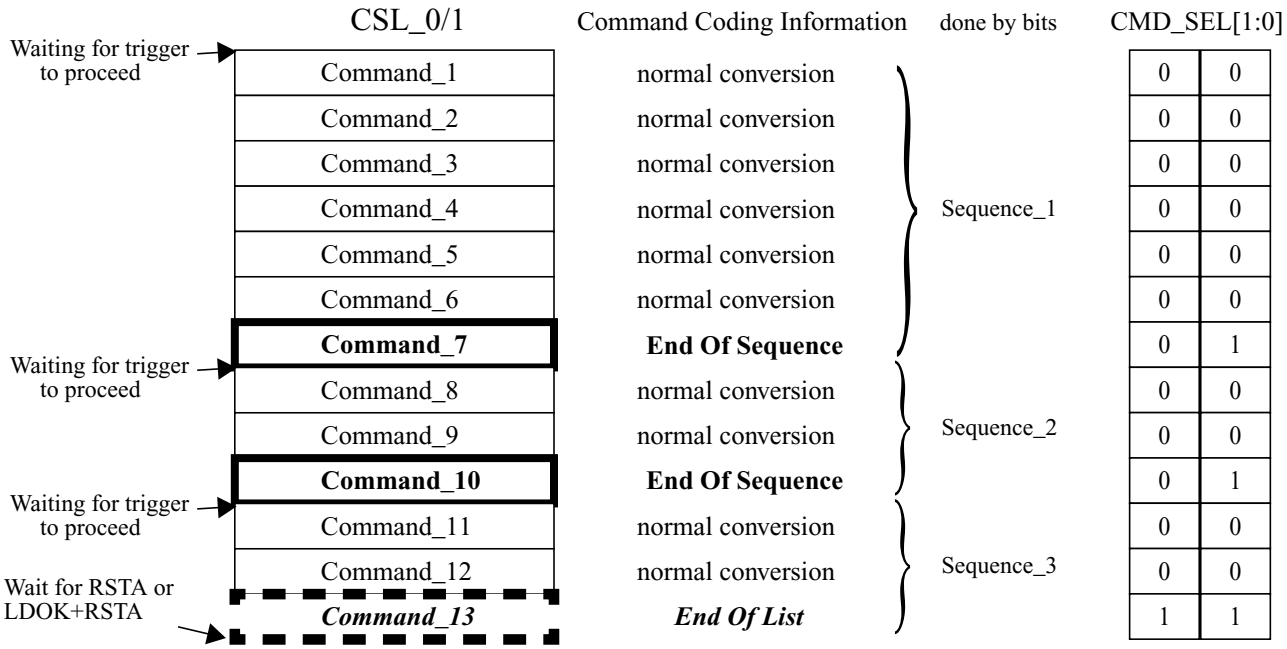



**Figure 10-29. Example CSL with sequences and an “End Of List” command type identifier**

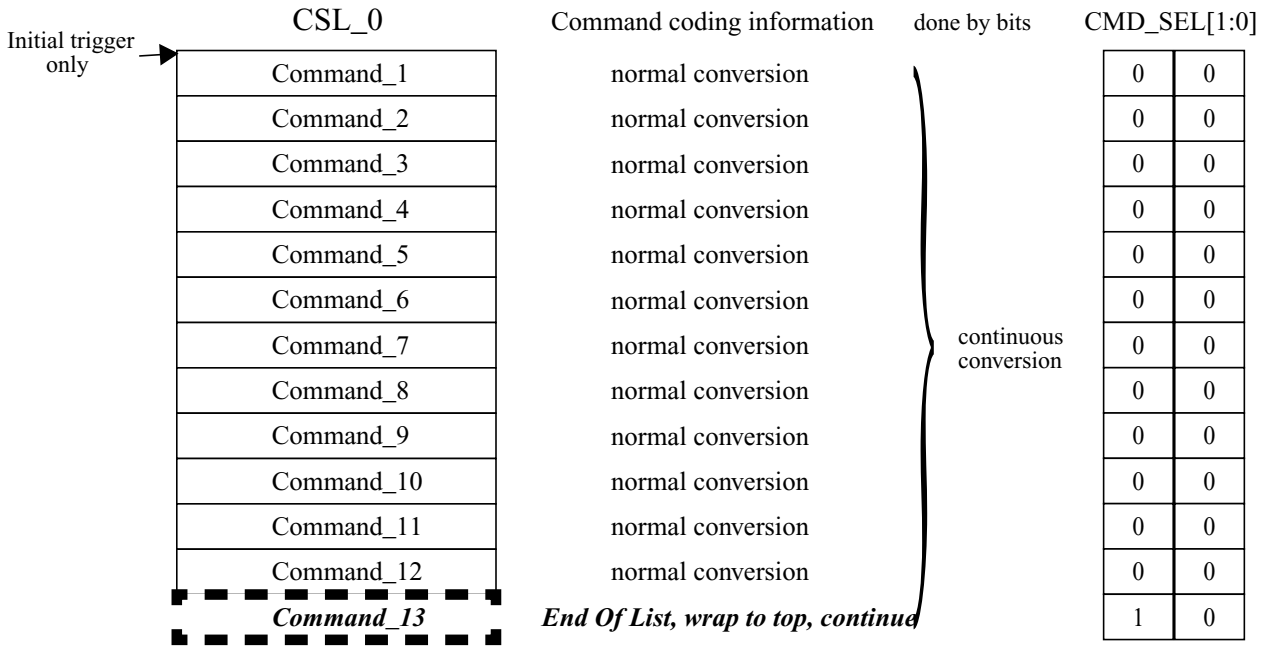
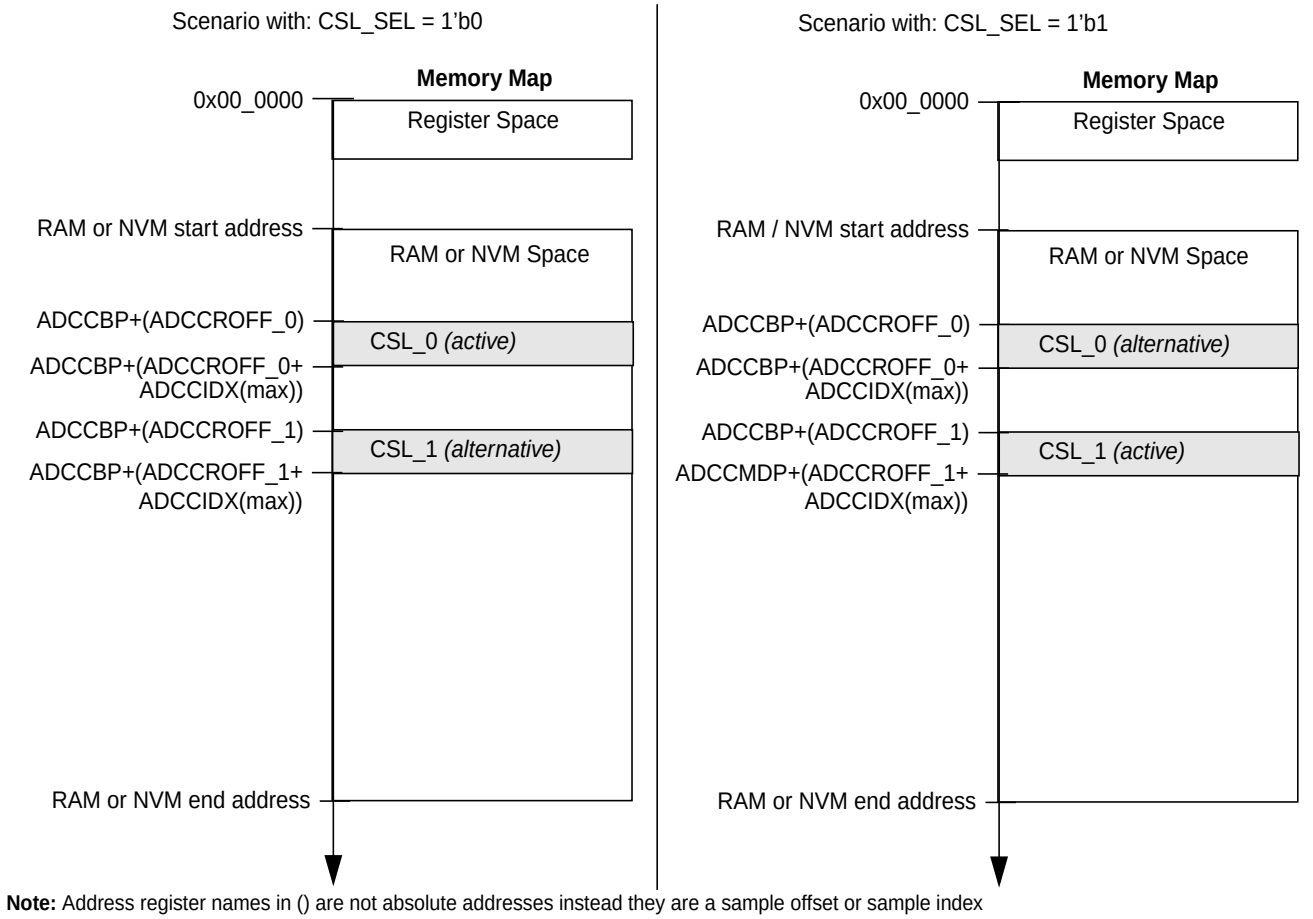



**Figure 10-30. Example CSL for continues conversion**

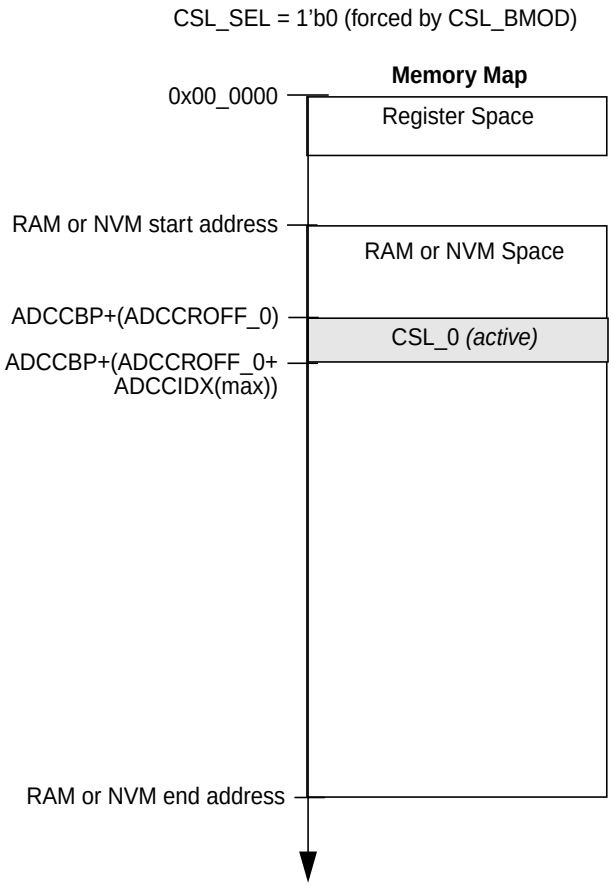
#### 10.5.3.2.2 Introduction of the two Command Sequence Lists (CSLs)

The two Command Sequence Lists (CSLs) can be referred to via the *Command Base Pointer Register* plus the *Command and Result Offset Registers* plus the *Command Index Register* (ADCCBP, ADCCROFF_0/1, ADCCIDX).
The final address for conversion command loading is calculated by the sum of these registers (e.g.: ADCCBP+ADCCROFF_0+ADCCIDX or ADCCBP+ADCCROFF_1+ADCCIDX).
Bit CSL_BMOD selects if the CSL is used in double buffer or single buffer mode. In double buffer mode the CSL can be swapped by flow control bits LDOK and RSTA. For detailed information about when and how the CSL is swapped please refer to Section 10.5.3.2.5, “The four ADC conversion flow control bits - description of Restart Event + CSL Swap, Section 10.8.7.1, “Initial Start of a Command Sequence List and Section 10.8.7.3, “Restart CSL execution with new/other CSL (alternative CSL becomes active CSL) - CSL swapping
Which list is actively used for ADC command loading is indicated by bit CSL_SEL. The register to define the CSL start addresses (ADCCBP) can be set to any even location of the system RAM or NVM area. It is the users responsibility to make sure that the different ADC lists do not overlap or exceed the system RAM respectively exceed the NVM area. The error flag IA_EIF will be set for accesses to ranges outside system RAM area and cause an error interrupt if enabled.




**Note:** Address register names in () are not absolute addresses instead they are a sample offset or sample index

**Figure 10-31. Command Sequence List Schema in Double Buffer Mode**




**Note:** Address register names in () are not absolute addresses instead they are a sample offset or sample index

**Figure 10-32. Command Sequence List Schema in Single Buffer Mode**

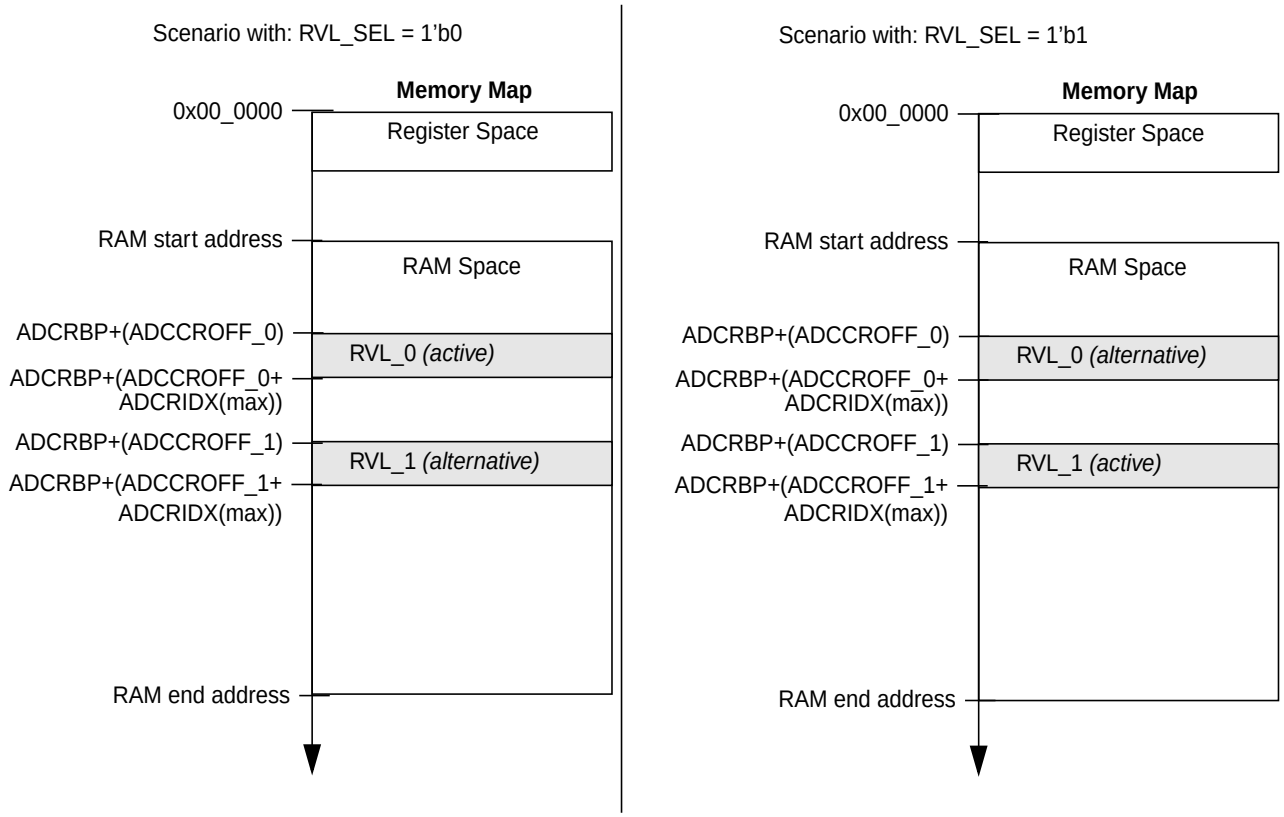
While the ADC is enabled one CSL is active (indicated by bit CSL_SEL) and corresponding list should not be modified anymore. At the same time the alternative CSL can be modified to prepare the ADC for new conversion sequences in CSL double buffered mode. When the ADC is enabled the command address registers (ADCCBP, ADCCROFF_0/2, ADCCIDX) are read only and register ADCCIDX is under control of the ADC.

#### 10.5.3.2.3 Introduction of the two Result Value Lists (RVLs)

The same list based architecture as describe above for the CSL has been implemented for the Result Value List (RVL) with corresponding address registers (ADCRBP, ADCCROFF_0/1, ADCRIDX).
The final address for conversion result storage is calculated by the sum of these registers (e.g.: ADCRBP+ADCCROFF_0+ADCRIDX or ADCRBP+ADCCROFF_1+ADCRIDX).
By bit RVL_BMOD it can be selected if the RVL is used in double buffer or single buffer mode. In double buffer mode the RVL is swapped:

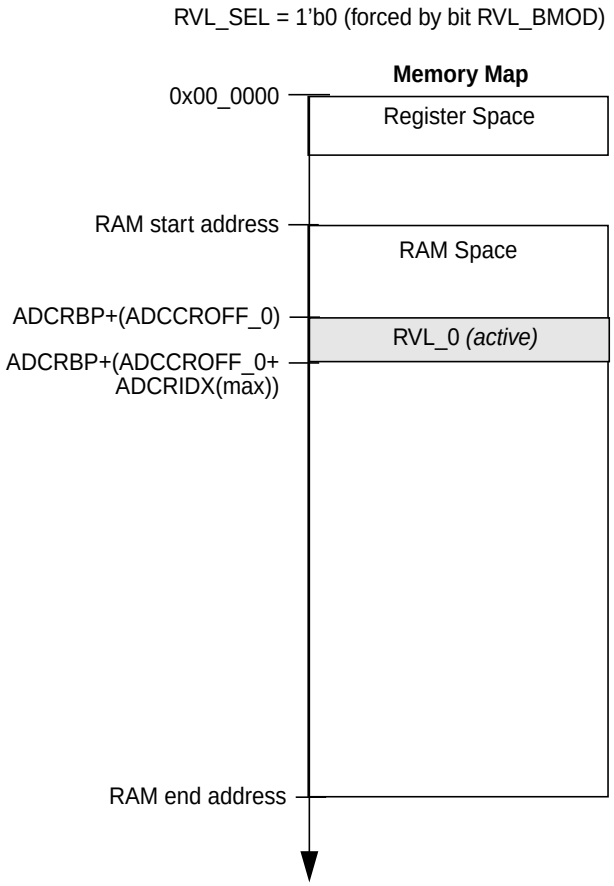
- Each time an “End Of List” command type got executed followed by the first conversion from top of the next CSL and related (first) result is about to be stored
- A CSL got aborted (bit SEQA=1’b1) and ADC enters idle state (becomes ready for new flow control events)

Using the RVL in double buffer mode the RVL is not swapped after exit from Stop Mode or Wait Mode with bit SWAI set. Hence the RVL used before entry of Stop or Wait Mode with bit SWAI set is overwritten after exit from the MCU Operating Mode (see also Section 10.2.1.2, “MCU Operating Modes).
Which list is actively used for ADC conversion result storage is indicated by bit RVL_SEL. The register to define the RVL start addresses (ADCRBP) can be set to any even location of the system RAM area. It is the users responsibility to make sure that the different ADC lists do not overlap or exceed the system RAM area. The error flag IA_EIF will be set for accesses to ranges outside system RAM area and cause an error interrupt if enabled.




**Note:** Address register names in () are not absolute addresses instead they are a sample offset or sample index

**Figure 10-33. Result Value List Schema in Double Buffer Mode**




**Note:** Address register names in () are not absolute addresses instead they are a sample offset or sample index

**Figure 10-34. Result Value List Schema in Single Buffer Mode**

While ADC is enabled one Result Value List is active (indicated by bit RVL_SEL). The conversion Result Value List can be read anytime. When the ADC is enabled the conversion result address registers (ADCRBP, ADCCROFF_0/1, ADCRIDX) are read only and register ADCRIDX is under control of the ADC.

A conversion result is always stored as 16bit entity in unsigned data representation. Left and right justification inside the entity is selected via the DJM control bit. Unused bits inside an entity are stored zero.

**Table 10-31. Conversion Result Justification Overview**

| Conversion Resolution (SRES[1:0]) | Left Justified Result (DJM = 1'b0) | Right Justified Result (DJM = 1'b1) |
|---|---|---|
| 8 bit | {Result[7:0],8'b00000000} | {8'b00000000,Result[7:0]} |
| 10 bit | {Result[9:0],6'b000000} | {6'b000000,Result[9:0]} |
| 12 bit | {Result[11:0],4'b0000} | {4'b0000,Result[11:0]} |

#### 10.5.3.2.4 The two conversion flow control Mode Configurations

The ADC provides two Modes (“Trigger Mode” and “Restart Mode”) which are different in the conversion control flow. The “Restart Mode” provides precise timing control about the sample start point but is more complex from the flow control perspective, while the “Trigger Mode” is more simple from flow control point of view but is less controllable regarding conversion sample start.

Following are the key differences:

In “Trigger Mode” configuration when conversion flow control bit RSTA gets set the bit TRIG gets set automatically. Hence in “Trigger Mode” the applications should not set the bit TRIG and bit RSTA simultaneously (via Data Bus or Internal Interface), because it is a flow control failure and the ADC will cease operation.
In “Trigger Mode” configuration after the execution of the initial Restart Event the current CSL can be executed and controlled via Trigger Events only. Hence if the “End Of List” command is reached a restart of conversion flow from top of current CSL does not require to set bit RSTA because returning to the top of current CSL is done automatically. Therefore the current CSL can be executed again after the “End Of List” command type is executed by a Trigger Event only.

In “Restart Mode” configuration the execution of a CSL is controlled via Trigger Events and Restart Events. After execution of the “End Of List” command the conversion flow must be continued by a Restart Event followed by a Trigger Event and the Trigger Event must not occur before the Restart Event has finished.

For more details and examples regarding flow control and application use cases please see following section and Section 10.8.7, “Conversion flow control application information

#### 10.5.3.2.5 The four ADC conversion flow control bits

There are four bits to control conversion flow (execution of a CSL and CSL exchange in double buffer mode). Each bit is controllable via the Data-Bus and Internal Interface depending on the setting of ACC_CFG[1:0] bits (see also Figure 10-2). In the following the conversion control event to control the conversion flow is given with the related Internal Interface Signal and corresponding register bit name together with information regarding:

- Function of the conversion control event
- How to request the event
- When is the event finished
- Mandatory requirements to executed the event

A summary of all event combinations is provided by Table 10-9.

- **Trigger Event**
  Internal Interface Signal: Trigger
  Corresponding Bit Name: TRIG
  - *Function:*
    Start the first conversion of a conversion sequence which is defined in the active Command Sequence List;

– *Requested by:*
Positive edge of Internal Interface signal Trigger;
Write Access via Data Bus to set control bit TRIG;

– *When finished:*
This bit is cleared by the ADC when the first conversion of the sequence is beginning to sample;

– *Mandatory Requirements:*
- In all ADC conversion flow control modes bit TRIG is only set (Trigger Event executed) if the Trigger Event occurs while no conversion or conversion sequence is ongoing (ADC idle);
- In ADC conversion flow control mode “Restart Mode” with a Restart Event in progress it is not allowed that a Trigger Event occurs before the background command load phase has finished (Restart Event has been executed) else the error flag TRIG_EIF is set;
- In ADC conversion flow control mode “Trigger Mode” a Restart Event causes bit TRIG being set automatically. Bit TRIG is set when no conversion or conversion sequence is ongoing (ADC idle) and the RVL done condition is reached by one of the following:
* A “End Of List” command type has been executed
* A Sequence Abort Event is in progress or has been executed
The ADC executes the Restart Event followed by the Trigger Event;
- In ADC conversion flow control mode “Trigger Mode” a Restart Event and a simultaneous Trigger Event via Internal Interface or Data Bus causes the TRIG_EIF bit being set and ADC cease operation.

• **Restart Event** (with current active CSL)
Internal Interface Signal: Restart
Corresponding Bit Name: RSTA

– *Function:*
Go to top of active CSL (clear index register for CSL);
Load one background command register and wait for Trigger (CSL offset register is not switched independent of bit CSL_BMOD);
Set error flag RSTA_EIF when a Restart Request occurs before one of the following conditions was reached:
* The "End Of List" command type has been executed
*Depending on bit STR_SEQA if the "End Of List" command type is about to be executed
* The current CSL has been aborted or is about to be aborted due to a Sequence Abort Request.

– *Requested by:*
Positive edge of Internal Interface signal Restart;
Write Access via Data Bus to set control bit RSTA;

– *When finished:*
This bit is cleared when the first conversion command of the sequence from top of active Sequence Command List is loaded;

– *Mandatory Requirement:*
- In all ADC conversion flow control modes a Restart Event causes bit RSTA to be set. Bit

SEQA is set simultaneously by ADC hardware if:
* ADC not idle (a conversion or conversion sequence is ongoing and current CSL not finished) and no Sequence Abort Event in progress (bit SEQA not already set or set simultaneously via Internal Interface or Data Bus)
* ADC idle but RVL done condition not reached
The RVL done condition is reached by one of the following:
* A “End Of List” command type has been executed
* A Sequence Abort Event is in progress or has been executed (bit SEQA already set or set simultaneously via Internal Interface or Data Bus)
The ADC executes the Sequence Abort Event followed by the Restart Event for the conditions described before or only a Restart Event.
- In ADC conversion flow control mode “Trigger Mode” a Restart Event causes bit TRIG being set automatically. Bit TRIG is set when no conversion or conversion sequence is ongoing (ADC idle) and the RVL done condition is reached by one of the following:
* A “End Of List” command type has been executed
* A Sequence Abort Event is in progress or has been executed
The ADC executes the Restart Event followed by the Trigger Event;
- In ADC conversion flow control mode “Trigger Mode” a Restart Event and a simultaneous Trigger Event via Internal Interface or Data Bus causes the TRIG_EIF bit being set and ADC cease operation.

- **Restart Event + CSL Exchange (Swap)**
  Internal Interface Signals: Restart + LoadOK
  Corresponding Bit Names: RSTA + LDOK
  – *Function:*
    Go to top of active CSL (clear index register for CSL) and switch to other offset register for address calculation if configured for double buffer mode (exchange the CSL list);
    *Requested by:*
    Internal Interface with the assertion of Interface Signal Restart the interface Signal LoadOK is evaluated and bit LDOK is set accordingly (bit LDOK set if Interface Signal LoadOK asserted when Interface Signal Restart asserts);
    Write Access via Data Bus to set control bit RSTA simultaneously with bit LDOK;
  – *When finished:*
    Bit LDOK can only be cleared if it was set as described before and both bits (LDOK, RSTA) are cleared when the first conversion command from top of active Sequence Command List is loaded;
  – *Mandatory Requirement:*
    No ongoing conversion or conversion sequence;
    Details if using the Internal Interface:
    If signal Restart is asserted before signal LoadOK is set the conversion starts from top of currently active CSL at the next Trigger Event (no exchange of CSL list);
    If signal Restart is asserted after or simultaneously with signal LoadOK the conversion starts from top of the other CSL at the next Trigger Event (CSL is switched) if CSL is configured

for double buffer mode;

- **Sequence Abort Event**
  Internal Interface Signal: Seq_Abort
  Corresponding Bit Name: SEQA
  - *Function:*
    Abort any possible ongoing conversion at next conversion boundary and abort current conversion sequence and active CSL;
  - *Requested by:*
    Positive edge of Internal Interface signal Seq_Abort;
    Write Access via Data Bus to set control bit SEQA;
  - *When finished:*
    This bit gets cleared when an ongoing conversion is finished and result is stored and/or an ongoing conversion sequence is aborted and current active CSL is aborted (ADC idle, RVL done);
  - *Mandatory Requirement:*
    - In all ADC conversion flow control modes bit SEQA can only be set if:
    * ADC not idle (a conversion or conversion sequence is ongoing)
    * ADC idle but RVL done condition not reached
    The RVL done condition is not reached if:
    * A “End Of List” command type has not been executed
    * A Sequence Abort Event has not been executed (bit SEQA not already set)
    - In all ADC conversion flow control modes a Sequence Abort Event can be issued at any time;
    - In ADC conversion flow control mode “Restart Mode” after a conversion sequence abort request has been executed it is mandatory to set bit RSTA; If a Trigger Event occurs before a Restart Event is executed (bit RSTA set and cleared by hardware) bit TRIG is set and error flag TRIG_EIF is set and the ADC can only be continued by a Soft-Reset; After the Restart Event the ADC accepts new Trigger Events (bit TRIG set) and begins conversion from top of the currently active CSL;
    - In ADC conversion flow control mode “Restart Mode” after a Sequence Abort Event has been executed a Restart Event causes bit RSTA being set only; The ADC executes a Restart Event only;
  - In both conversion flow control modes (“Restart Mode” and “Trigger Mode”) when conversion flow control bit RSTA gets set automatically bit SEQA gets set when the ADC has not reached one of the following scenarios:
    * A “End Of List” command type has been executed or is about to be executed
    * A Sequence Abort request is about to be executed or has been executed .
    In case bit SEQA is set automatically the Restart error flag RSTA_EIF is set to indicate an unexpected Restart Request.

#### 10.5.3.2.6 Conversion flow control in case of conversion sequence control bit overrun scenarios

Restart Request Overrun:
If a legal Restart Request is detected and no Restart Event is in progress it causes bit RSTA being set which indicates that a Restart Event is detected and processed. In case further Restart Requests occur while bit RSTA being set this is defined a overrun situation. This scenario is likely to occur when bit STR_SEQA is set or when a Restart Event causes a Sequence Abort Event. The request overrun is captured in a background register which stores always the last detected overrun request. Hence if the overrun situation occurs more than once while a Restart Event is in progress only the latest overrun request is pending and will be processed as soon as bit RSTA is cleared and therefore causes bit RSTA being set again one cycle later. The background register is cleared as soon as the pending Restart Request gets processed.

LoadOK Overrun:
Simultaneously at any Restart Request overrun situation the LoadOK input is evaluated and the status is captured in a background register which is alternated anytime a Restart Request Overrun occurs while Load OK Request is asserted. The Load OK background register is cleared as soon as the pending Restart Request gets processed.

Trigger Overrun:
If a Trigger occurs whilst bit TRIG is already set, this is defined as a Trigger overrun situation and causes the ADC to cease conversion at the next conversion boundary and to set bit TRIG_EIF. A overrun is also detected if the Trigger Event occurs automatically generated by hardware in “Trigger Mode” due to a Restart Event and simultaneously a Trigger Event is generated via Data Bus or Internal Interface. In this case the ADC ceases operation before conversion begins to sample. In “Trigger Mode” a Restart Request Overrun does not cause a Trigger Overrun (bit TRIG_EIF not set).

Sequence Abort Request Overrun:
If a Sequence Abort Request occurs whilst bit SEQA is already set, this is defined as a Sequence Abort Request Overrun situation and the overrun request is ignored.

#### 10.5.3.3 ADC List Usage and Conversion/Conversion Sequence Flow Description

It is the users responsibility to make sure that the different lists do not overlap or exceed the system RAM area respectively the CSL does not exceed the NVM area if located in the NVM. The error flag IA_EIF will be set for accesses done outside system RAM area and cause an error interrupt if enabled for lists which are located in the system RAM.

Generic flow for ADC register load at conversion sequence start/restart:

- It is mandatory that the ADC is idle (no ongoing conversion or conversion sequence)
- It is mandatory to have at least one CSL with valid entries. See also Section 10.8.7.2, “Restart CSL execution with currently active CSL or Section 10.8.7.3, “Restart CSL execution with new/other CSL (alternative CSL becomes active CSL) - CSL swapping for more details on possible scenarios
- A Restart Event occurs which causes the index registers getting cleared (register ADCCIDX and ADCRIDX are cleared) to point to the top of the corresponding lists (top of active RVL and CSL)
- Load conversion command to background conversion command register 1
- The control bit(s) RSTA (and LDOK if set) are cleared.
- Wait for Trigger Event to start conversion

Generic flow for ADC register load during conversion:

- The index registers ADCCIDX is incremented
- The inactive background command register is loaded with a new conversion command

Generic flow for ADC result storage at end of conversion:

- Index register ADCRIDX is incremented and conversion result to system RAM gets stored. As soon as the result is successfully stored any conversion interrupt flags are set accordingly.
- At the conversion boundary the other background command register becomes active and visible in the ADC register map.
- If the last executed conversion command was of type “End Of Sequence” the ADC waits for the Trigger Event;
- If the last executed conversion command was of type “End Of List” and ADC configured in “Restart Mode” the ADC set all related flags and stays idle awaiting a Restart Event to continue;
- If the last executed conversion command was of type “End Of List” and ADC configured in “Trigger Mode” the ADC set all related flags and automatically returns to top of current CSL and is awaiting a Trigger Event to continue;
- If the last executed conversion command was of type “Normal Conversion” the ADC continues command execution in the order of the current CSL (continues conversion);

## 10.6 Resets

At reset the ADC12B_LBA is disabled and in a power down state. The reset state of each individual bit is listed within the Register Description section (see Section 10.4.2, “Register Descriptions”) which details the registers and their bit-fields.

## 10.7 Interrupts

The ADC supports three types of interrupts:

- Conversion Interrupt
- Sequence Abort Interrupt
- Error and Conversion Flow Control Issue Interrupt

Each of the interrupt types is associated with individual interrupt enable bits and interrupt flags.

### 10.7.1 ADC Conversion Interrupt

The ADC provides one conversion interrupt associated to 16 interrupt enable bits with dedicated interrupt flags. The 16 interrupt flags consist of:

- 15 conversion interrupt flags which can be associated to any conversion completion.
- One additional interrupt flag which is fixed to the “End Of List” conversion command type within the active CSL.

The association of the conversion number with the interrupt flag number is done in the conversion command.

### 10.7.2 ADC Sequence Abort Done Interrupt

The ADC provides one sequence abort done interrupt associated with the sequence abort request for conversion flow control. Hence there is only one dedicated interrupt flag and interrupt enable bit for conversion sequence abort and occurs when the sequence abort is done.

### 10.7.3 ADC Error and Conversion Flow Control Issue Interrupt

The ADC provides one error interrupt for four error classes related to conversion interrupt overflow, command validness, DMA access status and Conversion Flow Control issues and CSL failure. The following error interrupt flags belong to the group of severe issues which cause an error interrupt if enabled and cease ADC operation:

- IA_EIF
- CMD_EIF
- EOL_EIF
- TRIG_EIF

In order to make the ADC operational again an ADC Soft-Reset must be issued which clears the above listed error interrupt flags.

**NOTE**

It is important to note that if flag DBECC_ERR is set the ADC ceases operation as well but does not cause an ADC error interrupt. Instead a machine exception is issued. In order to make the ADC operational again an ADC Soft-Reset must be issued.

Remaining error interrupt flags cause an error interrupt if enabled but ADC continues operation. The related interrupt flags are:

- RSTAR_EIF
- LDOK_EIF
- CONIF_OIF

# 10.8 Use Cases and Application Information

## 10.8.1 List Usage - CSL single buffer mode and RVL single buffer mode

In this use case both list types are configured for single buffer mode (CSL_BMOD=1’b0 and RVL_BMOD=1’b0, CSL_SEL and RVL_SEL are forced to 1’b0). Every time the conversion flow reaches the command containing the “End Of List” command type identifier or a Restart occurs at a sequence boundary or after an aborted conversion or conversion sequence the index register for the CSL and RVL are cleared to start from top of list with the next conversion command and result storage.

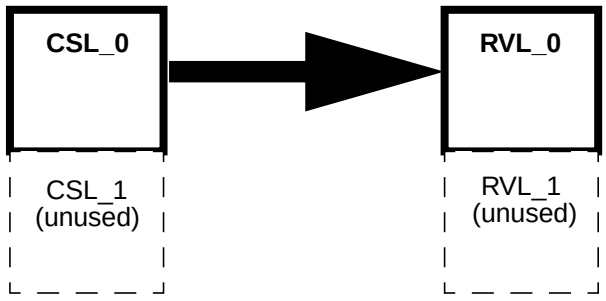



**Figure 10-35. CSL single buffer mode - RVL single buffer mode Diagram**

## 10.8.2 List Usage - CSL single buffer mode and RVL double buffer mode

In this use case the CSL is configured for single buffer mode (CSL_BMOD=1’b0) and the RVL is configured for double buffer mode (RVL_BMOD=1’b1). In this buffer configuration only the result list RVL is switched when the first conversion result of a CSL is stored after a CSL was successfully finished. or a CSL got aborted.

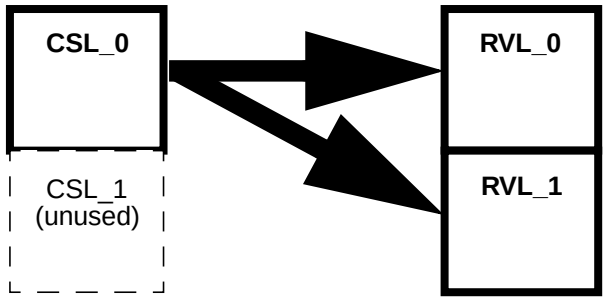



**Figure 10-36. CSL single buffer mode - RVL single buffer mode Diagram**

The last entirely filled RVL (a RVL where the corresponding CSL has been executed including the “End Of List “ command type) is shown by register ADCEOLRI.

The CSL is used in single buffer mode and bit CSL_SEL is forced to 1’b0.

### 10.8.3 List Usage - CSL double buffer mode and RVL double buffer mode

In this use case both list types are configured for double buffer mode (CSL_BMOD=1’b1 and RVL_BMOD=1’b1) and whenever a Command Sequence List (CSL) is finished or aborted the command Sequence List is swapped by simultaneous assertion of bits LDOK and RSTA.

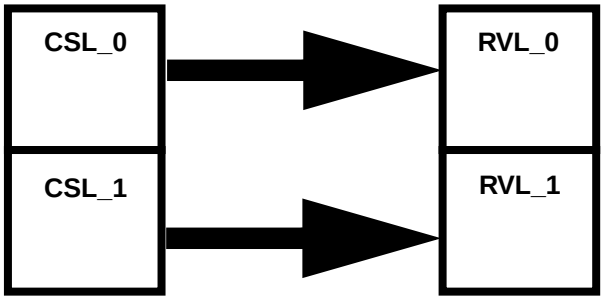



**Figure 10-37. CSL double buffer mode - RVL double buffer mode Diagram**

This use case can be used if the channel order or CSL length varies very frequently in an application.

### 10.8.4 List Usage - CSL double buffer mode and RVL single buffer mode

In this use case the CSL is configured for double buffer mode (CSL_BMOD=1’b1) and the RVL is configured for single buffer mode (RVL_BMOD=1’b0).

The two command lists can be of different size and the allocated result list memory area in the RAM must be able to hold as many entries as the larger of the two command lists. Each time when the end of a Command Sequence List is reached and if bits LDOK and RSTA are set the commands list is swapped.

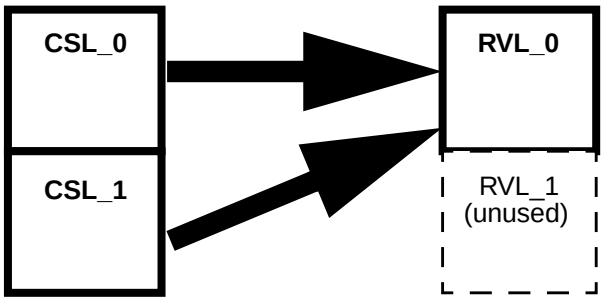



**Figure 10-38. CSL double buffer mode - RVL single buffer mode Diagram**

### 10.8.5 List Usage - CSL double buffer mode and RVL double buffer mode

In this use case both list types are configured for double buffer mode (CSL_BMOD=1’b1) and RVL_BMOD=1’b1).

This setup is the same as Section 10.8.3, “List Usage - CSL double buffer mode and RVL double buffer mode but at the end of a CSL the CSL is not always swapped (bit LDOK not always set with bit RSTA). The Result Value List is swapped whenever a CSL is finished or a CSL got aborted.

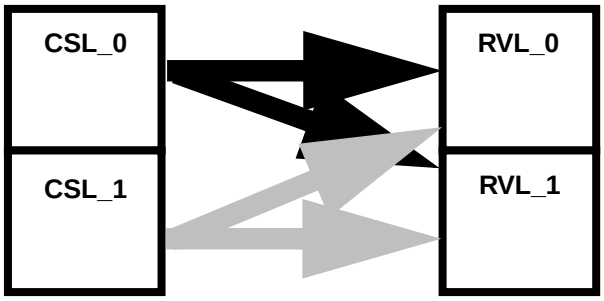
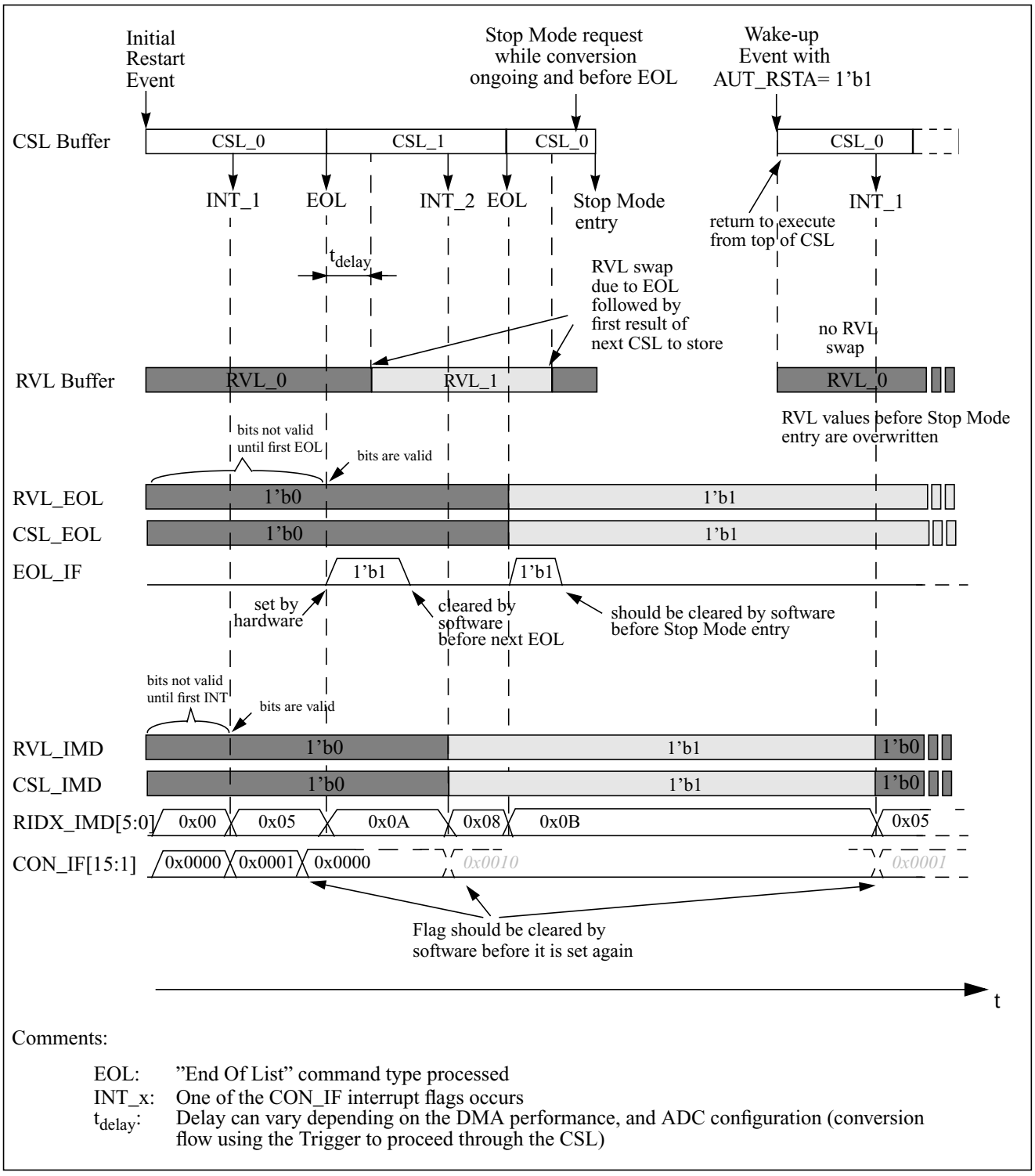



**Figure 10-39. CSL double buffer mode - RVL double buffer mode Diagram**

### 10.8.6 RVL swapping in RVL double buffer mode and related registers ADCIMDRI and ADCEOLRI

When using the RVL in double buffer mode the registers ADCIMDRI and ADCEOLRI can be used by the application software to identify which RVL holds relevant and latest data and which CSL is related to this data. These registers are updated at the setting of one of the CON_IF[15:1] or the EOL_IF interrupt flags. As described in the register description Section 10.4.2.13, “ADC Intermediate Result Information Register (ADCIMDRI) and Section 10.4.2.14, “ADC End Of List Result Information Register (ADCEOLRI) the register ADCIMDRI for instance is always updated at the occurrence of a CON_IF[15:1] interrupt flag amongst other cases. Also each time when the last conversion command of a CSL is finished and the corresponding result is stored the related EOL_IF flag is set and register ADCEOLRI is updated. Hence application software can pick-up conversion results or groups of results or an entire result list fully interrupt driven. A use case example diagram is shown in Figure 10-40.




Comments:

EOL: "End Of List" command type processed
INT_x: One of the CON_IF interrupt flags occurs
$t_{delay}$: Delay can vary depending on the DMA performance, and ADC configuration (conversion flow using the Trigger to proceed through the CSL)

**Figure 10-40. RVL swapping - use case diagram**

## 10.8.7 Conversion flow control application information

The ADC_12B_LBA provides various different conversion control scenarios to the user accomplished by the following features.

The ADC conversion flow control can be realized via the Data Bus only or the Internal Interface only or by both access methods. Which method to be used is software configurable via bits ACC_CFG[1:0].

The conversion flow is controlled via the four conversion flow control bits: SEQA, TRIG, RSTA, LDOK

Two different conversion flow control Modes can be configured: Trigger Mode or Restart Mode

Single or double buffer configuration of CSL and RVL.

### 10.8.7.1 Initial Start of a Command Sequence List

At the initial start of a Command Sequence List after device reset all entries for at least one of the two CSL must have been completed and data must be valid. Depending on if the CSL_0 or the CSL_1 should be executed at the initial start of a Command Sequence List the following conversion control sequence must be applied:

If CSL_0 should be executed at the initial conversion start after device reset:
A Restart Event and a Trigger Event must occur (depending to the selected conversion flow control mode the events must occur one after the other or simultaneously) which causes the ADC to start conversion with commands loaded from CSL_0.

If CSL_1 should be executed at the initial conversion start after device reset:
Bit LDOK must be set simultaneously with the Restart Event followed by a Trigger Event (depending on the selected conversion flow control mode the Trigger events must occur simultaneously or after the Restart event is finished). As soon as the Trigger Event gets executed the ADC starts conversion with commands loaded from CSL_1.

As soon as a new valid Restart Event occurs the flow for ADC register load at conversion sequence start as described in Section 10.5.3.3, “ADC List Usage and Conversion/Conversion Sequence Flow Description applies.

### 10.8.7.2 Restart CSL execution with currently active CSL

To restart a Command Sequence List execution it is mandatory that the ADC is idle (no conversion or conversion sequence is ongoing).

If necessary a possible ongoing conversion sequence can be aborted by the Sequence Abort Event (setting bit SEQA). As soon as bit SEQA got cleared by the ADC the current conversion sequence has been aborted and the ADC is idle (no conversion sequence or conversion ongoing)

After a conversion sequence abort is executed it is mandatory to request a Restart Event (bit RSTA set). After the Restart Event is finished (bit RSTA is cleared) the ADC accepts a new Trigger Event (bit TRIG can be set) and begins conversion from top of the currently active CSL. In conversion flow control mode “Trigger Mode” only a Restart Event is necessary if ADC is idle to restart Conversion Sequence List execution (the Trigger Event occurs automatically).

It is allowed to set bit RSTA and SEQA simultaneously which cause first the Sequence Abort Event to happen followed by a Restart Event.
If both flow control bits are set simultaneously in conversion flow control mode “Restart Mode” it prevents the error flags RSTA_EIF and LDOK_EIF to occur.
If both flow control bits are set simultaneously in conversion flow control mode “Trigger Mode” it prevents the error flag RSTA_EIF to occur.

If only a Restart Event occurs while ADC is not idle and bit SEQA is not set already (Sequence Abort Event in progress) a Sequence Abort Event is issued automatically and bit RSTAR_EIF is set.

Please see also the detailed conversion flow control bit Mandatory Requirements and execution information for bit RSTA and SEQA described in Section 10.5.3.2.5, “The four ADC conversion flow control bits.

#### 10.8.7.3 Restart CSL execution with new/other CSL (alternative CSL becomes active CSL) - CSL swapping

After all alternative conversion command list entries are finished the bit LDOK can be set simultaneously with the next Restart Event to swap command buffers.

To start conversion command list execution it is mandatory that the ADC is idle (no conversion or conversion sequence is ongoing).

If necessary a possible ongoing conversion sequence can be aborted by the Sequence Abort Event (setting bit SEQA). As soon as bit SEQA got cleared by the ADC the current conversion sequence has been aborted and the ADC is idle (no conversion sequence or conversion ongoing)

After a conversion sequence abort is executed it is mandatory to request a Restart Event (bit RSTA set) and simultaneously set bit LDOK to swap the CSL buffer. After the Restart Event is finished (bit RSTA and LDOK are cleared) the ADC accepts a new Trigger Event (bit TRIG can be set) and begins conversion from top of the newly selected CSL buffer. In conversion flow control mode “Trigger Mode” only a Restart Event (simultaneously with bit LDOK being set) is necessary to restart conversion command list execution with the newly selected CSL buffer (the Trigger Event occurs automatically).

It is allowed to set bits RSTA, LDOK and SEQA simultaneously which causes first the Sequence Abort Event to happen followed by a Restart Event.
If these bits are set simultaneously in “Restart Mode” it prevents the error flags RSTA_EIF and LDOK_EIF being set.
If these bits are set simultaneously in “Trigger Mode” it prevents the error flag RSTA_EIF being set.

If only a Restart Event occurs while ADC is not idle and bit SEQA is not set already (Sequence Abort Event in progress) a Sequence Abort Event is issued automatically and bit RSTAR_EIF is set.

Please see also the detailed conversion flow control bit Mandatory Requirements and execution information for bit RSTA and SEQA described in Section 10.5.3.2.5, “The four ADC conversion flow control bits.

## 10.8.8 Continuous Conversion

Applications which just require to continuously convert a list of channels without the need of timing control or performing different sequences of conversions (grouped number of different channels to convert) can make use of the following simple setup:

- “Trigger Mode” configuration

- Single buffer CSL

- Depending on data transfer rate either use single or double buffer RVL configuration

- Define a list of conversion commands which only contains the “End Of List” command with automatic wrap to top of CSL

After finishing the configuration and enabling the ADC an initial Restart Event is sufficient to launch the continuous conversion until next device reset or low power mode.

In case a Low Power Mode is used:
If bit AUT_RSTA is set before Low Power Mode is entered the conversion continues automatically as soon as a low power mode (Stop Mode or Wait Mode with bit SWAI set) is exited.

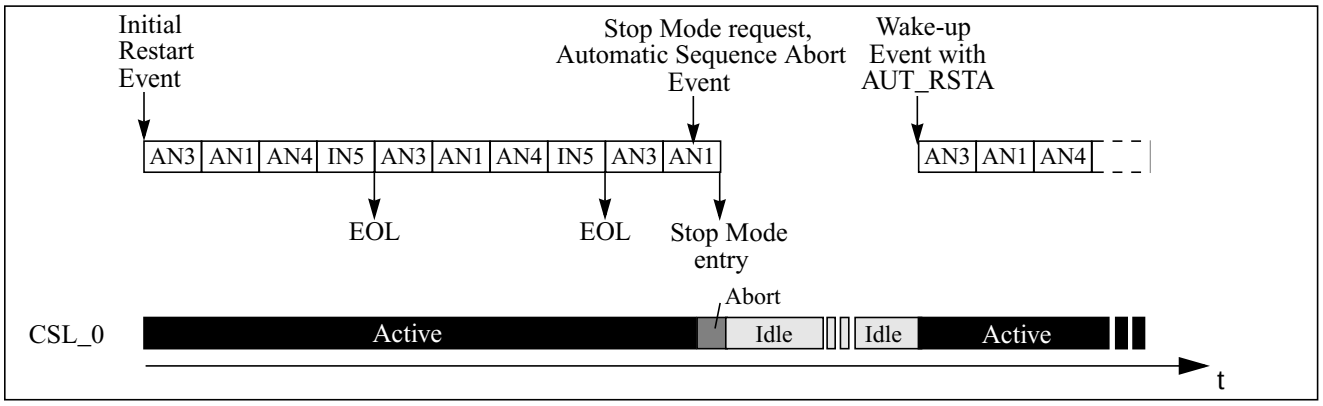



**Figure 10-41. Conversion Flow Control Diagram - Continuous Conversion (with Stop Mode)**

## 10.8.9 Triggered Conversion - Single CSL

Applications which require to convert one or more groups of different channels in a periodic and timed manner can make use of a configuration in “Trigger Mode” with a single CSL containing a list of sequences. This means the CSL consists of several sequences each separated by an “End of Sequence” command. The last command of the CSL uses the “End Of List” command with wrap to top of CSL and waiting for a Trigger (CMD_SEL[1:0] =2’b11). Hence after the initial Restart Event each sequence can be launched via a Trigger Event and repetition of the CSL can be launched via a Trigger after execution of the “End Of List” command.

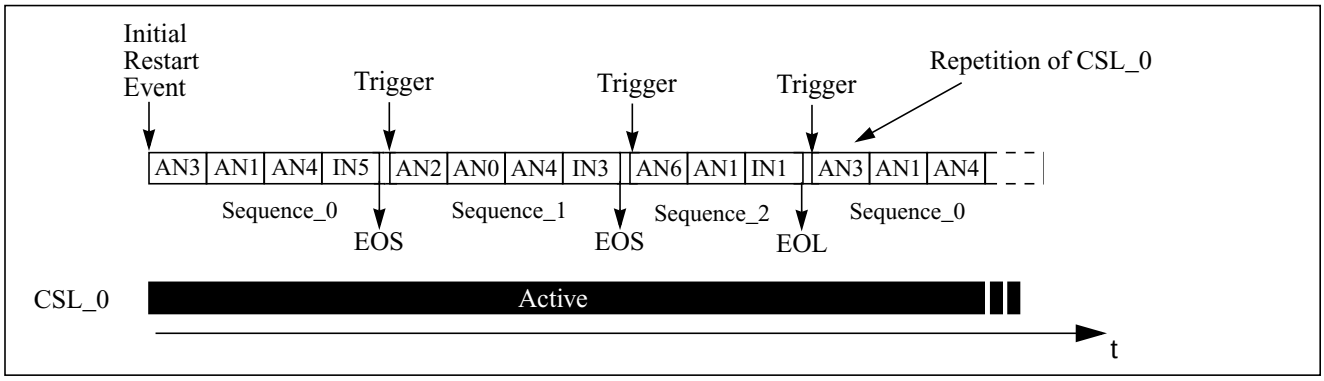



**Figure 10-42. Conversion Flow Control Diagram - Triggered Conversion (CSL Repetition)**

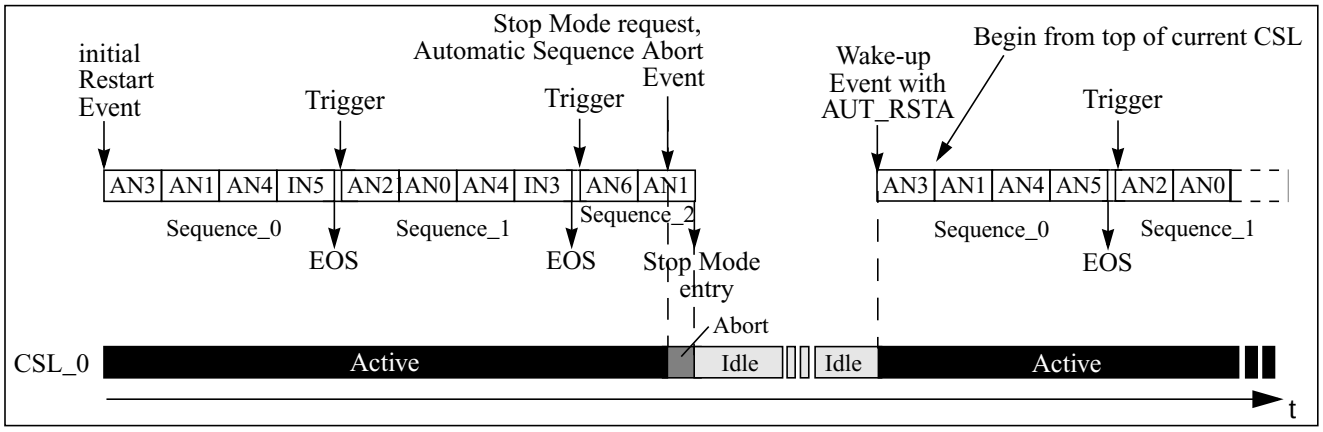



**Figure 10-43. Conversion Flow Control Diagram - Triggered Conversion (with Stop Mode)**

In case a Low Power Mode is used:
If bit AUT_RSTA is set before Low Power Mode is entered the conversion continues automatically as soon as a low power mode (Stop Mode or Wait Mode with bit SWAI set) is exited.

## 10.8.10 Fully Timing Controlled Conversion

As described before if using “Trigger Mode” a Restart Event always causes a Trigger automatically. To have full and precise timing control about the beginning of any conversion/sequence the “Restart Mode” is available. In “Restart Mode” a Restart Event does not cause a Trigger automatically instead the Trigger must be issued separately and timing correct, which means the Trigger is not allowed before the Restart Event (conversion command loading) is finished (bit RSTA=1’b0 again). The time required from Trigger until sampling phase starts is given (refer to Section 10.4.2.6, “ADC Conversion Flow Control Register (ADCFLWCTL), Timing considerations) and hence timing is fully controllable by the application. Additionally if a Trigger occurs before a Restart Event is finished, this causes the TRIG_EIF flag being set. This allows detection of false flow control sequences.

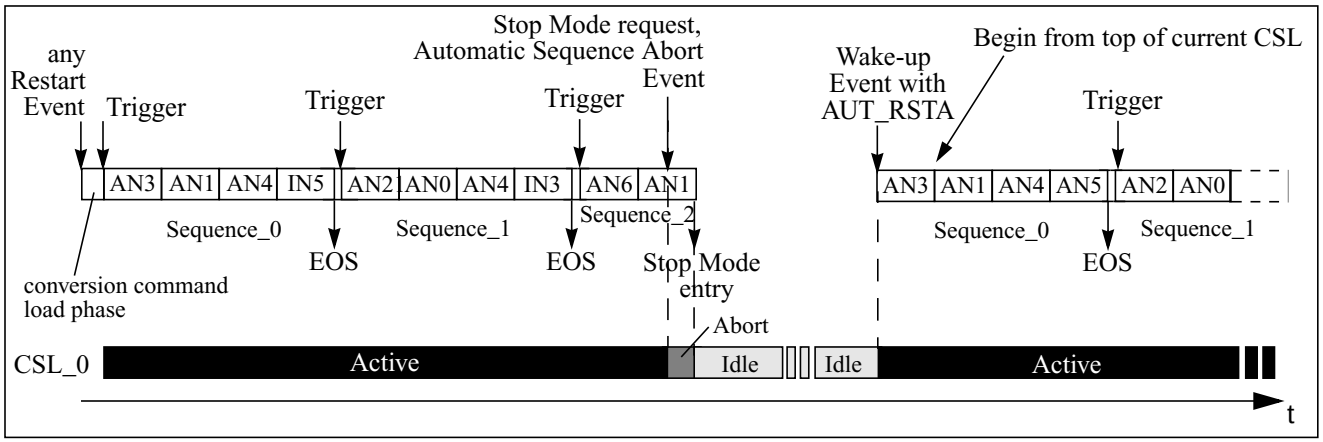



**Figure 10-44. Conversion Flow Control Diagram - Fully Timing Controlled Conversion (with Stop Mode)**

Different to the Stop Mode entry shown in Figure 10-43 and Figure 10-44 it is recommended to issue the Stop Mode at sequence boundaries (when ADC is idle and no conversion/conversion sequence is ongoing).

Any of the Conversion flow control application use cases described above (Continuous, Triggered or Fully Timing Controlled Conversion) can be used with CSL single buffer mode or with CSL double buffer mode. If using the CSL double buffer mode the CSL swapping is performed by issuing a Restart Event with bit LDOK being set.

# Chapter 11
# Freescale's Scalable Controller Area Network (S12MSCANV3)

## Revision History

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V03.14 | 12 Nov 2012 | Table 11-10 | • Corrected RxWRN and TxWRN threshold values |
| V03.15 | 12 Jan 2013 | Table 11-2<br>Table 11-25<br>Figure 11-37<br>11.1/11-415<br>11.3.2.15/11-436 | • Updated TIME bit description<br>• Added register names to buffer map<br>• Updated TSRH and TSRL read conditions<br>• Updated introduction<br>• Updated CANTXERR and CANRXERR register notes |
| V03.16 | 08 Aug 2013 | | • Corrected typos |

## 11.1 Introduction

Freescale's scalable controller area network (S12MSCANV3) definition is based on the MSCAN12 definition, which is the specific implementation of the MSCAN concept targeted for the S12, S12X and S12Z microcontroller families.

The module is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991. For users to fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to familiarize the reader with the terms and concepts contained within this document.

Though not exclusively intended for automotive applications, CAN protocol is designed to meet the specific requirements of a vehicle serial data bus: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN uses an advanced buffer arrangement resulting in predictable real-time behavior and simplified application software.

## 11.1.1 Glossary

**Table 11-1. Terminology**

| | |
|---|---|
| ACK | Acknowledge of CAN message |
| CAN | Controller Area Network |
| CRC | Cyclic Redundancy Code |
| EOF | End of Frame |
| FIFO | First-In-First-Out Memory |
| IFS | Inter-Frame Sequence |
| SOF | Start of Frame |
| CPU bus | CPU related read/write data bus |
| CAN bus | CAN protocol related serial bus |
| oscillator clock | Direct clock from external oscillator |
| bus clock | CPU bus related clock |
| CAN clock | CAN protocol related clock |

## 11.1.2 Block Diagram

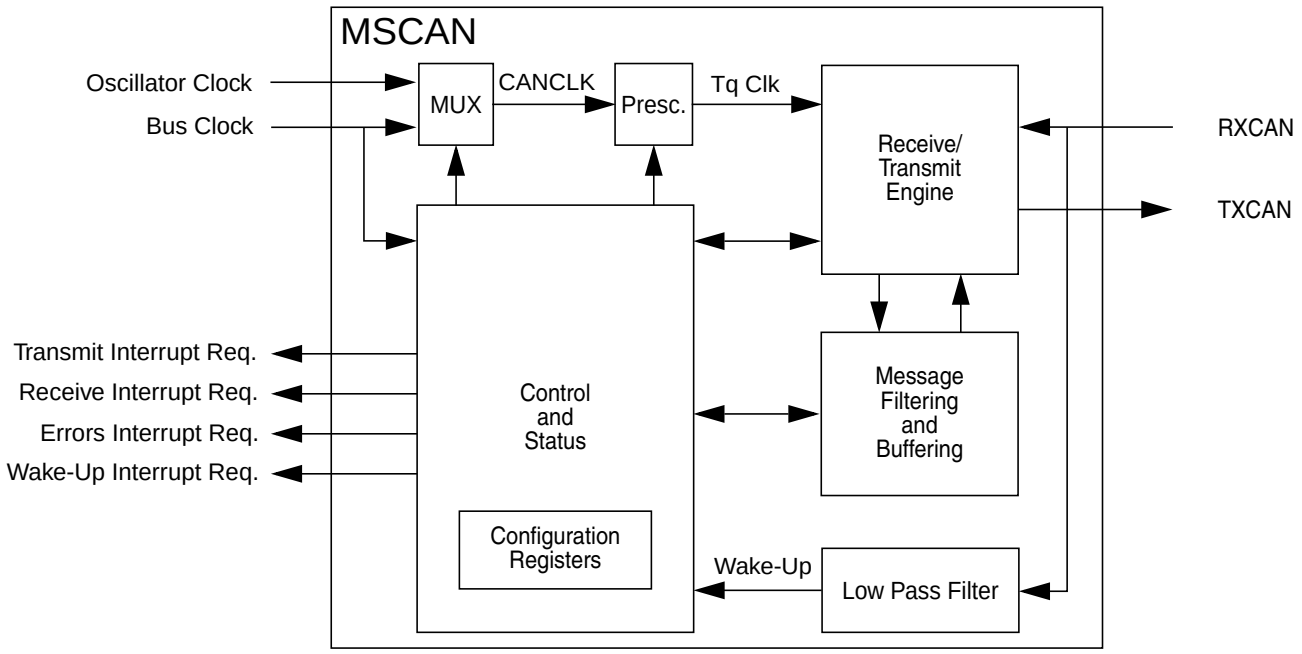



**Figure 11-1. MSCAN Block Diagram**

### 11.1.3 Features

The basic features of the MSCAN are as follows:

- Implementation of the CAN protocol — Version 2.0A/B
  - — Standard and extended data frames
  - — Zero to eight bytes data length
  - — Programmable bit rate up to 1 Mbps[1]
  - — Support for remote frames
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a "local priority" concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or eight 8-bit filters
- Programmable wake-up functionality with integrated low-pass filter
- Programmable loopback mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Programmable bus-off recovery functionality
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)
- Programmable MSCAN clock source either bus clock or oscillator clock
- Internal timer for time-stamping of received and transmitted messages
- Three low-power modes: sleep, power down, and MSCAN enable
- Global initialization of configuration registers

### 11.1.4 Modes of Operation

For a description of the specific MSCAN modes and the module operation related to the system operating modes refer to Section 11.4.4, "Modes of Operation".

1. Depending on the actual bit timing and the clock jitter of the PLL.

## 11.2 External Signal Description

The MSCAN uses two external pins.

**NOTE**

On MCUs with an integrated CAN physical interface (transceiver) the MSCAN interface is connected internally to the transceiver interface. In these cases the external availability of signals TXCAN and RXCAN is optional.

### 11.2.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

### 11.2.2 TXCAN — CAN Transmitter Output Pin

TXCAN is the MSCAN transmitter output pin. The TXCAN output pin represents the logic level on the CAN bus:

0 = Dominant state
1 = Recessive state

### 11.2.3 CAN System

A typical CAN system with MSCAN is shown in Figure 11-2. Each CAN station is connected physically to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defective CAN or defective stations.

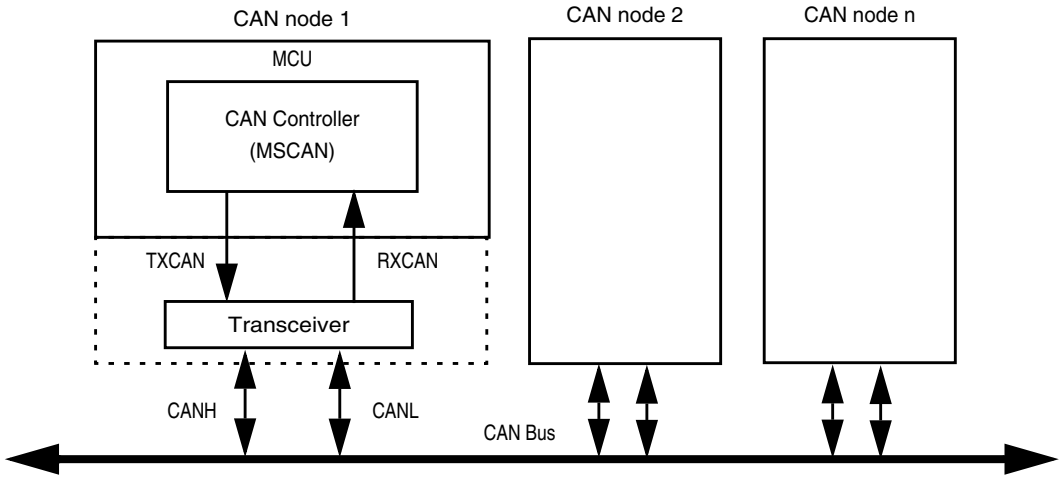



**Figure 11-2. CAN System**

## 11.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the MSCAN.

### 11.3.1 Module Memory Map

Figure 11-3 gives an overview on all registers and their individual bits in the MSCAN memory map. The *register address* results from the addition of *base address* and *address offset*. The *base address* is determined at the MCU level and can be found in the MCU memory map description. The *address offset* is defined at the module level.

The MSCAN occupies 64 bytes in the memory space. The base address of the MSCAN module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

The detailed register descriptions follow in the order they appear in the register map.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 CANCTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | W | | | | | | | | |
| 0x0001 CANCTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | W | | | | | | | | |
| 0x0002 CANBTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | W | | | | | | | | |
| 0x0003 CANBTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | W | | | | | | | | |
| 0x0004 CANRFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | W | | | | | | | | |
| 0x0005 CANRIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | W | | | | | | | | |
| 0x0006 CANTFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | W | | | | | | | | |
| 0x0007 CANTIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | W | | | | | | | | |
| 0x0008 CANTARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | W | | | | | | | | |
| 0x0009 CANTAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | W | | | | | | | | |
| 0x000A CANTBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | W | | | | | | | | |
| 0x000B CANIDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | W | | | | | | | | |
| 0x000C Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000D CANMISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | W | | | | | | | | |
| 0x000E CANRXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 11-3. MSCAN Register Summary**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000F CANTXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | W | | | | | | | | |
| 0x0010–0x0013 CANIDAR0–3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | W | | | | | | | | |
| 0x0014–0x0017 CANIDMRx | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | W | | | | | | | | |
| 0x0018–0x001B CANIDAR4–7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | W | | | | | | | | |
| 0x001C–0x001F CANIDMR4–7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | W | | | | | | | | |
| 0x0020–0x002F CANRXFG | R | See Section 11.3.3, "Programmer's Model of Message Storage" | | | | | | | |
| | W | | | | | | | | |
| 0x0030–0x003F CANTXFG | R | See Section 11.3.3, "Programmer's Model of Message Storage" | | | | | | | |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 11-3. MSCAN Register Summary (continued)**

## 11.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the MSCAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read.

### 11.3.2.1 MSCAN Control Register 0 (CANCTL0)

The CANCTL0 register provides various control bits of the MSCAN module as described below.

Module Base + 0x0000 — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 11-4. MSCAN Control Register 0 (CANCTL0)**

1. Read: Anytime
   Write: Anytime when out of initialization mode; exceptions are read-only RXACT and SYNCH, RXFRM (which is set by the module only), and INITRQ (which is also writable in initialization mode)

**NOTE**

The CANCTL0 register, except WUPE, INITRQ, and SLPRQ, is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

**Table 11-2. CANCTL0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RXFRM | **Received Frame Flag** — This bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. After it is set, it remains set until cleared by software or reset. Clearing is done by writing a 1. Writing a 0 is ignored. This bit is not valid in loopback mode.<br>0 No valid message was received since last clearing this flag<br>1 A valid message was received since last clearing of this flag |
| 6<br>RXACT | **Receiver Active Status** — This read-only flag indicates the MSCAN is receiving a message(1). The flag is controlled by the receiver front end. This bit is not valid in loopback mode.<br>0 MSCAN is transmitting or idle<br>1 MSCAN is receiving a message (including when arbitration is lost) |
| 5<br>CSWAI(2) | **CAN Stops in Wait Mode** — Enabling this bit allows for lower power consumption in wait mode by disabling all the clocks at the CPU bus interface to the MSCAN module.<br>0 The module is not affected during wait mode<br>1 The module ceases to be clocked during wait mode |
| 4<br>SYNCH | **Synchronized Status** — This read-only flag indicates whether the MSCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the MSCAN.<br>0 MSCAN is not synchronized to the CAN bus<br>1 MSCAN is synchronized to the CAN bus |
| 3<br>TIME | **Timer Enable** — This bit activates an internal 16-bit wide free running timer which is clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp will be assigned to each transmitted/received message within the active TX/RX buffer. Right after the EOF of a valid message on the CAN bus, the time stamp is written to the highest bytes (0x000E, 0x000F) in the appropriate buffer (see Section 11.3.3, "Programmer's Model of Message Storage"). In loopback mode no receive timestamp is generated. The internal timer is reset (all bits set to 0) when disabled. This bit is held low in initialization mode.<br>0 Disable internal MSCAN timer<br>1 Enable internal MSCAN timer |
| 2<br>WUPE(3) | **Wake-Up Enable** — This configuration bit allows the MSCAN to restart from sleep mode or from power down mode (entered from sleep) when traffic on CAN is detected (see Section 11.4.5.5, "MSCAN Sleep Mode"). This bit must be configured before sleep mode entry for the selected function to take effect.<br>0 Wake-up disabled — The MSCAN ignores traffic on CAN<br>1 Wake-up enabled — The MSCAN is able to restart |

**Table 11-2. CANCTL0 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SLPRQ[4] | **Sleep Mode Request** — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see Section 11.4.5.5, "MSCAN Sleep Mode"). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPAK = 1 (see Section 11.3.2.2, "MSCAN Control Register 1 (CANCTL1)"). SLPRQ cannot be set while the WUPIF flag is set (see Section 11.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)"). Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself.<br>0 Running — The MSCAN functions normally<br>1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle |
| 0<br>INITRQ[5],[6] | **Initialization Mode Request** — When this bit is set by the CPU, the MSCAN skips to initialization mode (see Section 11.4.4.5, "MSCAN Initialization Mode"). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (Section 11.3.2.2, "MSCAN Control Register 1 (CANCTL1)").<br>The following registers enter their hard reset state and restore their default values: CANCTL0[7], CANRFLG[8], CANRIER[9], CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL.<br>The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode.<br>When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits.<br>Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0.<br>0 Normal operation<br>1 MSCAN in initialization mode |

1. See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.
2. In order to protect from accidentally violating the CAN protocol, TXCAN is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see Section 11.4.5.2, "Operation in Wait Mode" and Section 11.4.5.3, "Operation in Stop Mode").
3. The CPU has to make sure that the WUPE register and the WUPIE wake-up interrupt enable register (see Section 11.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)) is enabled, if the recovery mechanism from stop or wait is required.
4. The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1).
5. The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).
6. In order to protect from accidentally violating the CAN protocol, TXCAN is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before requesting initialization mode.
7. Not including WUPE, INITRQ, and SLPRQ.
8. TSTAT1 and TSTAT0 are not affected by initialization mode.
9. RSTAT1 and RSTAT0 are not affected by initialization mode.

### 11.3.2.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

Module Base + 0x0001 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 11-5. MSCAN Control Register 1 (CANCTL1)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except CANE which is write once in normal and anytime in special system operation modes when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-3. CANCTL1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CANE | **MSCAN Enable**<br>0 MSCAN module is disabled<br>1 MSCAN module is enabled |
| 6<br>CLKSRC | **MSCAN Clock Source** — This bit defines the clock source for the MSCAN module (only for systems with a clock generation module; Section 11.4.3.2, "Clock System," and Section Figure 11-43., "MSCAN Clocking Scheme,").<br>0 MSCAN clock source is the oscillator clock<br>1 MSCAN clock source is the bus clock |
| 5<br>LOOPB | **Loopback Self Test Mode** — When this bit is set, the MSCAN performs an internal loopback which can be used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input is ignored and the TXCAN output goes to the recessive state (logic 1). The MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, the MSCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.<br>0 Loopback self test disabled<br>1 Loopback self test enabled |
| 4<br>LISTEN | **Listen Only Mode** — This bit configures the MSCAN as a CAN bus monitor. When LISTEN is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out (see Section 11.4.4.4, "Listen-Only Mode"). In addition, the error counters are frozen. Listen only mode supports applications which require "hot plugging" or throughput analysis. The MSCAN is unable to transmit any messages when listen only mode is active.<br>0 Normal operation<br>1 Listen only mode activated |
| 3<br>BORM | **Bus-Off Recovery Mode** — This bit configures the bus-off state recovery mode of the MSCAN. Refer to Section 11.5.2, "Bus-Off Recovery," for details.<br>0 Automatic bus-off recovery (see Bosch CAN 2.0A/B protocol specification)<br>1 Bus-off recovery upon user request |
| 2<br>WUPM | **Wake-Up Mode** — If WUPE in CANCTL0 is enabled, this bit defines whether the integrated low-pass filter is applied to protect the MSCAN from spurious wake-up (see Section 11.4.5.5, "MSCAN Sleep Mode").<br>0 MSCAN wakes up on any dominant level on the CAN bus<br>1 MSCAN wakes up only in case of a dominant pulse on the CAN bus that has a length of $T_{wup}$ |

**Table 11-3. CANCTL1 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SLPAK | **Sleep Mode Acknowledge** — This flag indicates whether the MSCAN module has entered sleep mode (see Section 11.4.5.5, "MSCAN Sleep Mode"). It is used as a handshake flag for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ = 1 and SLPAK = 1. Depending on the setting of WUPE, the MSCAN will clear the flag if it detects activity on the CAN bus while in sleep mode.<br>0 Running — The MSCAN operates normally<br>1 Sleep mode active — The MSCAN has entered sleep mode |
| 0<br>INITAK | **Initialization Mode Acknowledge** — This flag indicates whether the MSCAN module is in initialization mode (see Section 11.4.4.5, "MSCAN Initialization Mode"). It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ = 1 and INITAK = 1. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0–CANIDAR7, and CANIDMR0–CANIDMR7 can be written only by the CPU when the MSCAN is in initialization mode.<br>0 Running — The MSCAN operates normally<br>1 Initialization mode active — The MSCAN has entered initialization mode |

### 11.3.2.3 MSCAN Bus Timing Register 0 (CANBTR0)

The CANBTR0 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0002 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-6. MSCAN Bus Timing Register 0 (CANBTR0)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-4. CANBTR0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-6<br>SJW[1:0] | **Synchronization Jump Width** — The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus (see Table 11-5). |
| 5-0<br>BRP[5:0] | **Baud Rate Prescaler** — These bits determine the time quanta (Tq) clock which is used to build up the bit timing (see Table 11-6). |

**Table 11-5. Synchronization Jump Width**

| SJW1 | SJW0 | Synchronization Jump Width |
|---|---|---|
| 0 | 0 | 1 Tq clock cycle |
| 0 | 1 | 2 Tq clock cycles |
| 1 | 0 | 3 Tq clock cycles |
| 1 | 1 | 4 Tq clock cycles |

**Table 11-6. Baud Rate Prescaler**

| BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | Prescaler value (P) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| : | : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 1 | 64 |

### 11.3.2.4 MSCAN Bus Timing Register 1 (CANBTR1)

The CANBTR1 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0003 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-7. MSCAN Bus Timing Register 1 (CANBTR1)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-7. CANBTR1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SAMP | **Sampling** — This bit determines the number of CAN bus samples taken per bit time.<br>0 One sample per bit.<br>1 Three samples per bit(1).<br>If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP = 0). |
| 6-4<br>TSEG2[2:0] | **Time Segment 2** — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 11-44). Time segment 2 (TSEG2) values are programmable as shown in Table 11-8. |
| 3-0<br>TSEG1[3:0] | **Time Segment 1** — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 11-44). Time segment 1 (TSEG1) values are programmable as shown in Table 11-9. |

1. In this case, PHASE_SEG1 must be at least 2 time quanta (Tq).

**Table 11-8. Time Segment 2 Values**

| TSEG22 | TSEG21 | TSEG20 | Time Segment 2 |
|---|---|---|---|
| 0 | 0 | 0 | 1 Tq clock cycle(1) |
| 0 | 0 | 1 | 2 Tq clock cycles |
| : | : | : | : |
| 1 | 1 | 0 | 7 Tq clock cycles |
| 1 | 1 | 1 | 8 Tq clock cycles |

1. This setting is not valid. Please refer to Table 11-36 for valid settings.

**Table 11-9. Time Segment 1 Values**

| TSEG13 | TSEG12 | TSEG11 | TSEG10 | Time segment 1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 Tq clock cycle(1) |
| 0 | 0 | 0 | 1 | 2 Tq clock cycles1 |
| 0 | 0 | 1 | 0 | 3 Tq clock cycles1 |
| 0 | 0 | 1 | 1 | 4 Tq clock cycles |
| : | : | : | : | : |
| 1 | 1 | 1 | 0 | 15 Tq clock cycles |
| 1 | 1 | 1 | 1 | 16 Tq clock cycles |

1. This setting is not valid. Please refer to Table 11-36 for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in Table 11-8 and Table 11-9).

***Eqn. 11-1***

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{CANCLK}} \bullet (1 + \text{TimeSegment1} + \text{TimeSegment2})$$

### 11.3.2.5 MSCAN Receiver Flag Register (CANRFLG)

A flag can be cleared only by software (writing a 1 to the corresponding bit position) when the condition which caused the setting is no longer valid. Every flag has an associated interrupt enable bit in the CANRIER register.

Module Base + 0x0004 — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-8. MSCAN Receiver Flag Register (CANRFLG)**

1. Read: Anytime
   Write: Anytime when not in initialization mode, except RSTAT[1:0] and TSTAT[1:0] flags which are read-only; write of 1 clears flag; write of 0 is ignored

**NOTE**

The CANRFLG register is held in the reset state[1] when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

**Table 11-10. CANRFLG Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WUPIF | **Wake-Up Interrupt Flag** — If the MSCAN detects CAN bus activity while in sleep mode (see Section 11.4.5.5, "MSCAN Sleep Mode,") and WUPE = 1 in CANTCTL0 (see Section 11.3.2.1, "MSCAN Control Register 0 (CANCTL0)"), the module will set WUPIF. If not masked, a wake-up interrupt is pending while this flag is set.<br>0 No wake-up activity observed while in sleep mode<br>1 MSCAN detected activity on the CAN bus and requested wake-up |
| 6<br>CSCIF | **CAN Status Change Interrupt Flag** — This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status (see Section 11.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)"). If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again.<br>0 No change in CAN bus status occurred since last interrupt<br>1 MSCAN changed current CAN bus status |
| 5-4<br>RSTAT[1:0] | **Receiver Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is:<br>00 RxOK: 0 ≤ receive error counter < 96<br>01 RxWRN: 96 ≤ receive error counter < 128<br>10 RxERR: 128 ≤ receive error counter<br>*11 Bus-off(1): 256 ≤ transmit error counter* |
| 3-2<br>TSTAT[1:0] | **Transmitter Status Bits** — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the MSCAN. The coding for the bits TSTAT1, TSTAT0 is:<br>00 TxOK: 0 ≤ transmit error counter < 96<br>01 TxWRN: 96 ≤ transmit error counter < 128<br>10 TxERR: 128 ≤ transmit error counter < 256<br>11 Bus-Off: 256 ≤ transmit error counter |

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

**Table 11-10. CANRFLG Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>OVRIF | **Overrun Interrupt Flag** — This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.<br>0 No data overrun condition<br>1 A data overrun detected |
| 0<br>RXF(2) | **Receive Buffer Full Flag** — RXF is set by the MSCAN when a new message is shifted in the receiver FIFO. This flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After the CPU has read that message from the RxFG buffer in the receiver FIFO, the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the shifting of the next FIFO entry into the foreground buffer (RxFG). If not masked, a receive interrupt is pending while this flag is set.<br>0 No new message available within the RxFG<br>1 The receiver FIFO is not empty. A new message is available in the RxFG |

1. Redundant information for the most critical CAN bus status which is "bus-off". This only occurs if the Tx error counter exceeds a number of 255 errors. Bus-off affects the receiver state. As soon as the transmitter leaves its bus-off state the receiver state skips to RxOK too. Refer also to TSTAT[1:0] coding in this register.
2. To ensure data integrity, do not read the receive buffer registers while the RXF flag is cleared. For MCUs with dual CPUs, reading the receive buffer registers while the RXF flag is cleared may result in a CPU fault condition.

### 11.3.2.6 MSCAN Receiver Interrupt Enable Register (CANRIER)

This register contains the interrupt enable bits for the interrupt flags described in the CANRFLG register.

Module Base + 0x0005 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-9. MSCAN Receiver Interrupt Enable Register (CANRIER)**

1. Read: Anytime
   Write: Anytime when not in initialization mode

**NOTE**

The CANRIER register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).

The RSTATE[1:0], TSTATE[1:0] bits are not affected by initialization mode.

**Table 11-11. CANRIER Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>WUPIE[1] | **Wake-Up Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A wake-up event causes a Wake-Up interrupt request. |
| 6<br>CSCIE | **CAN Status Change Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A CAN Status Change event causes an error interrupt request. |
| 5-4<br>RSTATE[1:0] | **Receiver Status Change Enable** — These RSTAT enable bits control the sensitivity level in which receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTAT flags continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending.<br>00 Do not generate any CSCIF interrupt caused by receiver state changes.<br>01 Generate CSCIF interrupt only if the receiver enters or leaves "bus-off" state. Discard other receiver state changes for generating CSCIF interrupt.<br>10 Generate CSCIF interrupt only if the receiver enters or leaves "RxErr" or "bus-off"[2] state. Discard other receiver state changes for generating CSCIF interrupt.<br>11 Generate CSCIF interrupt on all state changes. |
| 3-2<br>TSTATE[1:0] | **Transmitter Status Change Enable** — These TSTAT enable bits control the sensitivity level in which transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTAT flags continue to indicate the actual transmitter state and are only updated if no CSCIF interrupt is pending.<br>00 Do not generate any CSCIF interrupt caused by transmitter state changes.<br>01 Generate CSCIF interrupt only if the transmitter enters or leaves "bus-off" state. Discard other transmitter state changes for generating CSCIF interrupt.<br>10 Generate CSCIF interrupt only if the transmitter enters or leaves "TxErr" or "bus-off" state. Discard other transmitter state changes for generating CSCIF interrupt.<br>11 Generate CSCIF interrupt on all state changes. |
| 1<br>OVRIE | **Overrun Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 An overrun event causes an error interrupt request. |
| 0<br>RXFIE | **Receiver Full Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A receive buffer full (successful message reception) event causes a receiver interrupt request. |

1. WUPIE and WUPE (see Section 11.3.2.1, "MSCAN Control Register 0 (CANCTL0)") must both be enabled if the recovery mechanism from stop or wait is required.
2. Bus-off state is only defined for transmitters by the CAN standard (see Bosch CAN 2.0A/B protocol specification). Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RXSTAT[1:0] flags define an additional bus-off state for the receiver (see Section 11.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)").

### 11.3.2.7 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

Module Base + 0x0006 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

= Unimplemented

**Figure 11-10. MSCAN Transmitter Flag Register (CANTFLG)**

1. Read: Anytime
   Write: Anytime when not in initialization mode; write of 1 clears flag, write of 0 is ignored

**NOTE**

The CANTFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 11-12. CANTFLG Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>TXE[2:0] | **Transmitter Buffer Empty** — This flag indicates that the associated transmit message buffer is empty, and thus not scheduled for transmission. The CPU must clear the flag after a message is set up in the transmit buffer and is due for transmission. The MSCAN sets the flag after the message is sent successfully. The flag is also set by the MSCAN when the transmission request is successfully aborted due to a pending abort request (see Section 11.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)"). If not masked, a transmit interrupt is pending while this flag is set.<br>Clearing a TXEx flag also clears the corresponding ABTAKx (see Section 11.3.2.10, "MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)"). When a TXEx flag is set, the corresponding ABTRQx bit is cleared (see Section 11.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)").<br>When listen-mode is active (see Section 11.3.2.2, "MSCAN Control Register 1 (CANCTL1)") the TXEx flags cannot be cleared and no transmission is started.<br>Read and write accesses to the transmit buffer will be blocked, if the corresponding TXEx bit is cleared (TXEx = 0) and the buffer is scheduled for transmission.<br>0 The associated message buffer is full (loaded with a message due for transmission)<br>1 The associated message buffer is empty (not scheduled) |

## 11.3.2.8 MSCAN Transmitter Interrupt Enable Register (CANTIER)

This register contains the interrupt enable bits for the transmit buffer empty interrupt flags.

Module Base + 0x0007 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-11. MSCAN Transmitter Interrupt Enable Register (CANTIER)**

1. Read: Anytime
   Write: Anytime when not in initialization mode

**NOTE**

The CANTIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 11-13. CANTIER Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>TXEIE[2:0] | **Transmitter Empty Interrupt Enable**<br>0 No interrupt request is generated from this event.<br>1 A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request. |

### 11.3.2.9 MSCAN Transmitter Message Abort Request Register (CANTARQ)

The CANTARQ register allows abort request of queued messages as described below.

Module Base + 0x0008 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-12. MSCAN Transmitter Message Abort Request Register (CANTARQ)**

1. Read: Anytime
   Write: Anytime when not in initialization mode

**NOTE**

The CANTARQ register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 11-14. CANTARQ Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>ABTRQ[2:0] | **Abort Request** — The CPU sets the ABTRQx bit to request that a scheduled message buffer (TXEx = 0) be aborted. The MSCAN grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and abort acknowledge flags (ABTAK, see Section 11.3.2.10, "MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)") are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQx. ABTRQx is reset whenever the associated TXE flag is set.<br>0 No abort request<br>1 Abort request pending |

### 11.3.2.10 MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)

The CANTAAK register indicates the successful abort of a queued message, if requested by the appropriate bits in the CANTARQ register.

Module Base + 0x0009 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-13. MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)**

1. Read: Anytime
   Write: Unimplemented

**NOTE**

The CANTAAK register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

**Table 11-15. CANTAAK Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0 ABTAK[2:0] | **Abort Acknowledge** — This flag acknowledges that a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this flag can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAKx flag is cleared whenever the corresponding TXE flag is cleared.<br>0 The message was not aborted.<br>1 The message was aborted. |

### 11.3.2.11 MSCAN Transmit Buffer Selection Register (CANTBSEL)

The CANTBSEL register allows the selection of the actual transmit message buffer, which then will be accessible in the CANTXFG register space.

Module Base + 0x000A Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-14. MSCAN Transmit Buffer Selection Register (CANTBSEL)**

1. Read: Find the lowest ordered bit set to 1, all other bits will be read as 0
   Write: Anytime when not in initialization mode

**NOTE**

The CANTBSEL register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK=1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

**Table 11-16. CANTBSEL Register Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>TX[2:0] | **Transmit Buffer Select** — The lowest numbered bit places the respective transmit buffer in the CANTXFG register space (e.g., TX1 = 1 and TX0 = 1 selects transmit buffer TX0; TX1 = 1 and TX0 = 0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer will be blocked, if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)").<br>0 The associated message buffer is deselected<br>1 The associated message buffer is selected, if lowest numbered bit |

The following gives a short programming example of the usage of the CANTBSEL register:

To get the next available transmit buffer, application software must read the CANTFLG register and write this value back into the CANTBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CANTFLG is therefore 0b0000_0110. When writing this value back to CANTBSEL, the Tx buffer TX1 is selected in the CANTXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CANTBSEL results in 0b0000_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software's selection of the next available Tx buffer.

- LDAA CANTFLG; value read is 0b0000_0110
- STAA CANTBSEL; value written is 0b0000_0110
- LDAA CANTBSEL; value read is 0b0000_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG registers.

### 11.3.2.12 MSCAN Identifier Acceptance Control Register (CANIDAC)

The CANIDAC register is used for identifier acceptance control as described below.

Module Base + 0x000B — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-15. MSCAN Identifier Acceptance Control Register (CANIDAC)**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHITx, which are read-only

**Table 11-17. CANIDAC Register Field Descriptions**

| Field | Description |
|---|---|
| 5-4<br>IDAM[1:0] | **Identifier Acceptance Mode** — The CPU sets these flags to define the identifier acceptance filter organization (see Section 11.4.3, "Identifier Acceptance Filter"). Table 11-18 summarizes the different settings. In filter closed mode, no message is accepted such that the foreground buffer is never reloaded. |
| 2-0<br>IDHIT[2:0] | Identifier Acceptance Hit Indicator — The MSCAN sets these flags to indicate an identifier acceptance hit (see Section 11.4.3, "Identifier Acceptance Filter"). Table 11-19 summarizes the different settings. |

**Table 11-18. Identifier Acceptance Mode Settings**

| IDAM1 | IDAM0 | Identifier Acceptance Mode |
|---|---|---|
| 0 | 0 | Two 32-bit acceptance filters |
| 0 | 1 | Four 16-bit acceptance filters |
| 1 | 0 | Eight 8-bit acceptance filters |
| 1 | 1 | Filter closed |

**Table 11-19. Identifier Acceptance Hit Indication**

| IDHIT2 | IDHIT1 | IDHIT0 | Identifier Acceptance Hit |
|---|---|---|---|
| 0 | 0 | 0 | Filter 0 hit |
| 0 | 0 | 1 | Filter 1 hit |
| 0 | 1 | 0 | Filter 2 hit |
| 0 | 1 | 1 | Filter 3 hit |
| 1 | 0 | 0 | Filter 4 hit |
| 1 | 0 | 1 | Filter 5 hit |
| 1 | 1 | 0 | Filter 6 hit |
| 1 | 1 | 1 | Filter 7 hit |

The IDHITx indicators are always related to the message in the foreground buffer (RxFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well.

### 11.3.2.13 MSCAN Reserved Register

This register is reserved for factory testing of the MSCAN module and is not available in normal system operating modes.

Module Base + 0x000C — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-16. MSCAN Reserved Register**

1. Read: Always reads zero in normal system operation modes
   Write: Unimplemented in normal system operation modes

**NOTE**

Writing to this register when in special system operating modes can alter the MSCAN functionality.

### 11.3.2.14 MSCAN Miscellaneous Register (CANMISC)

This register provides additional features.

Module Base + 0x000D — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-17. MSCAN Miscellaneous Register (CANMISC)**

1. Read: Anytime
   Write: Anytime; write of '1' clears flag; write of '0' ignored

**Table 11-20. CANMISC Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>BOHOLD | **Bus-off State Hold Until User Request** — If BORM is set in MSCAN Control Register 1 (CANCTL1), this bit indicates whether the module has entered the bus-off state. Clearing this bit requests the recovery from bus-off. Refer to Section 11.5.2, "Bus-Off Recovery," for details.<br>0 Module is not bus-off or recovery has been requested by user in bus-off state<br>1 Module is bus-off and holds this state until user request |

### 11.3.2.15 MSCAN Receive Error Counter (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

Module Base + 0x000E — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-18. MSCAN Receive Error Counter (CANRXERR)**

1. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)
   Write: Unimplemented

**NOTE**

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

### 11.3.2.16 MSCAN Transmit Error Counter (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

Module Base + 0x000F — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-19. MSCAN Transmit Error Counter (CANTXERR)**

1. Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)
   Write: Unimplemented

**NOTE**

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

### 11.3.2.17 MSCAN Identifier Acceptance Registers (CANIDAR0-7)

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the IDR0–IDR3 registers (see Section 11.3.3.1, "Identifier Registers (IDR0–IDR3)") of incoming messages in a bit by bit manner (see Section 11.4.3, "Identifier Acceptance Filter").

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CANIDAR0/1, CANIDMR0/1) are applied.

Module Base + 0x0010 to Module Base + 0x0013 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-20. MSCAN Identifier Acceptance Registers (First Bank) — CANIDAR0–CANIDAR3**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-21. CANIDAR0–CANIDAR3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AC[7:0] | **Acceptance Code Bits** — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register. |

Module Base + 0x0018 to Module Base + 0x001B Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-21. MSCAN Identifier Acceptance Registers (Second Bank) — CANIDAR4–CANIDAR7**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-22. CANIDAR4–CANIDAR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AC[7:0] | **Acceptance Code Bits** — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register. |

### 11.3.2.18 MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1 and CANIDMR5 to "don't care." To receive standard identifiers in 16 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1, CANIDMR3, CANIDMR5, and CANIDMR7 to "don't care."

Module Base + 0x0014 to Module Base + 0x0017 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-22. MSCAN Identifier Mask Registers (First Bank) — CANIDMR0–CANIDMR3**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-23. CANIDMR0–CANIDMR3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AM[7:0] | **Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.<br>0 Match corresponding acceptance code register and identifier bits<br>1 Ignore corresponding acceptance code register bit |

Module Base + 0x001C to Module Base + 0x001F Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-23. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 11-24. CANIDMR4–CANIDMR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AM[7:0] | **Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.<br>0 Match corresponding acceptance code register and identifier bits<br>1 Ignore corresponding acceptance code register bit |

## 11.3.3 Programmer's Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13 byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (see Section 11.3.2.1, "MSCAN Control Register 0 (CANCTL0)").

The time stamp register is written by the MSCAN. The CPU can only read these registers.

**Table 11-25. Message Buffer Organization**

| Offset Address | Register | Access |
|---|---|---|
| 0x00X0 | IDR0 — Identifier Register 0 | R/W |
| 0x00X1 | IDR1 — Identifier Register 1 | R/W |
| 0x00X2 | IDR2 — Identifier Register 2 | R/W |
| 0x00X3 | IDR3 — Identifier Register 3 | R/W |
| 0x00X4 | DSR0 — Data Segment Register 0 | R/W |
| 0x00X5 | DSR1 — Data Segment Register 1 | R/W |
| 0x00X6 | DSR2 — Data Segment Register 2 | R/W |
| 0x00X7 | DSR3 — Data Segment Register 3 | R/W |
| 0x00X8 | DSR4 — Data Segment Register 4 | R/W |
| 0x00X9 | DSR5 — Data Segment Register 5 | R/W |
| 0x00XA | DSR6 — Data Segment Register 6 | R/W |
| 0x00XB | DSR7 — Data Segment Register 7 | R/W |
| 0x00XC | DLR — Data Length Register | R/W |
| 0x00XD | TBPR — Transmit Buffer Priority Register(1) | R/W |
| 0x00XE | TSRH — Time Stamp Register (High Byte) | R |
| 0x00XF | TSRL — Time Stamp Register (Low Byte) | R |

1. Not applicable for receive buffers

Figure 11-24 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 11-25.

All bits of the receive and transmit buffers are 'x' out of reset because of RAM-based implementation[1]. All reserved or unused bits of the receive and transmit buffers always read 'x'.

1. Exception: The transmit buffer priority registers are 0 out of reset.

**Figure 11-24. Receive/Transmit Message Buffer — Extended Identifier Mapping**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x00X0 IDR0 | R | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| | W | | | | | | | | |
| 0x00X1 IDR1 | R | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| | W | | | | | | | | |
| 0x00X2 IDR2 | R | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| | W | | | | | | | | |
| 0x00X3 IDR3 | R | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| | W | | | | | | | | |
| 0x00X4 DSR0 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X5 DSR1 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X6 DSR2 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X7 DSR3 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X8 DSR4 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00X9 DSR5 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00XA DSR6 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00XB DSR7 | R | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| | W | | | | | | | | |
| 0x00XC DLR | R | | | | | DLC3 | DLC2 | DLC1 | DLC0 |
| | W | | | | | | | | |

**Figure 11-24. Receive/Transmit Message Buffer — Extended Identifier Mapping (continued)**

| Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit0 |
|---|---|---|---|---|---|---|---|---|

= Unused, always read 'x'

Read:

- For transmit buffers, anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").
- For receive buffers, only when RXF flag is set (see Section 11.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)").

Write:

- For transmit buffers, anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").
- Unimplemented for receive buffers.

Reset: Undefined because of RAM-based implementation

**Figure 11-25. Receive/Transmit Message Buffer — Standard Identifier Mapping**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| IDR0 0x00X0 | R | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| | W | | | | | | | | |
| IDR1 0x00X1 | R | ID2 | ID1 | ID0 | RTR | IDE (=0) | | | |
| | W | | | | | | | | |
| IDR2 0x00X2 | R | | | | | | | | |
| | W | | | | | | | | |
| IDR3 0x00X3 | R | | | | | | | | |
| | W | | | | | | | | |

= Unused, always read 'x'

### 11.3.3.1 Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consist of a total of 32 bits: ID[28:0], SRR, IDE, and RTR. The identifier registers for a standard format identifier consist of a total of 13 bits: ID[10:0], RTR, and IDE.

#### 11.3.3.1.1 IDR0–IDR3 for Extended Identifier Mapping

Module Base + 0x00X0

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-26. Identifier Register 0 (IDR0) — Extended Identifier Mapping**

**Table 11-26. IDR0 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-0<br>ID[28:21] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X1

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-27. Identifier Register 1 (IDR1) — Extended Identifier Mapping**

**Table 11-27. IDR1 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-5<br>ID[20:18] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |
| 4<br>SRR | **Substitute Remote Request** — This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers. |
| 3<br>IDE | **ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send.<br>0 Standard format (11 bit)<br>1 Extended format (29 bit) |
| 2-0<br>ID[17:15] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X2

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-28. Identifier Register 2 (IDR2) — Extended Identifier Mapping**

**Table 11-28. IDR2 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-0<br>ID[14:7] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |

Module Base + 0x00X3

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-29. Identifier Register 3 (IDR3) — Extended Identifier Mapping**

**Table 11-29. IDR3 Register Field Descriptions — Extended**

| Field | Description |
|---|---|
| 7-1<br>ID[6:0] | **Extended Format Identifier —** The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. |
| 0<br>RTR | **Remote Transmission Request** — This flag reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.<br>0 Data frame<br>1 Remote frame |

### 11.3.3.1.2 IDR0–IDR3 for Standard Identifier Mapping

Module Base + 0x00X0

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-30. Identifier Register 0 — Standard Mapping**

**Table 11-30. IDR0 Register Field Descriptions — Standard**

| Field | Description |
|---|---|
| 7-0<br>ID[10:3] | **Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 11-31. |

Module Base + 0x00X1

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID2 | ID1 | ID0 | RTR | IDE (=0) | | | |
| Reset: | x | x | x | x | x | x | x | x |

= Unused; always read 'x'

**Figure 11-31. Identifier Register 1 — Standard Mapping**

**Table 11-31. IDR1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-5<br>ID[2:0] | **Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 11-30. |
| 4<br>RTR | **Remote Transmission Request** — This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.<br>0 Data frame<br>1 Remote frame |
| 3<br>IDE | **ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send.<br>0 Standard format (11 bit)<br>1 Extended format (29 bit) |

Module Base + 0x00X2

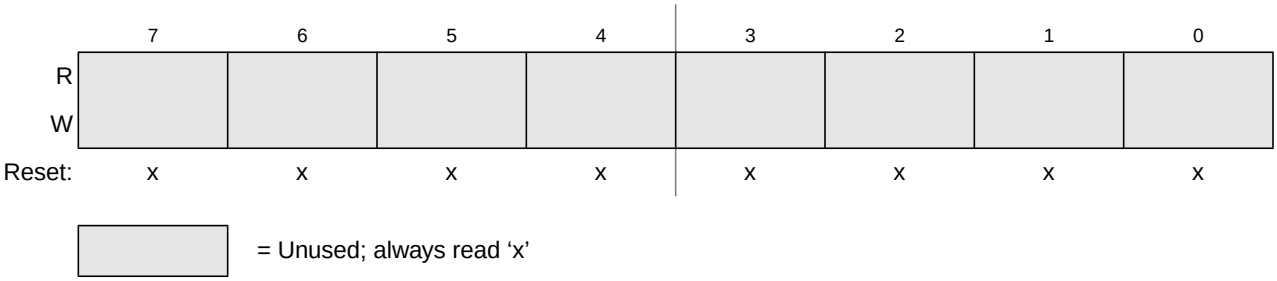



**Figure 11-32. Identifier Register 2 — Standard Mapping**

Module Base + 0x00X3

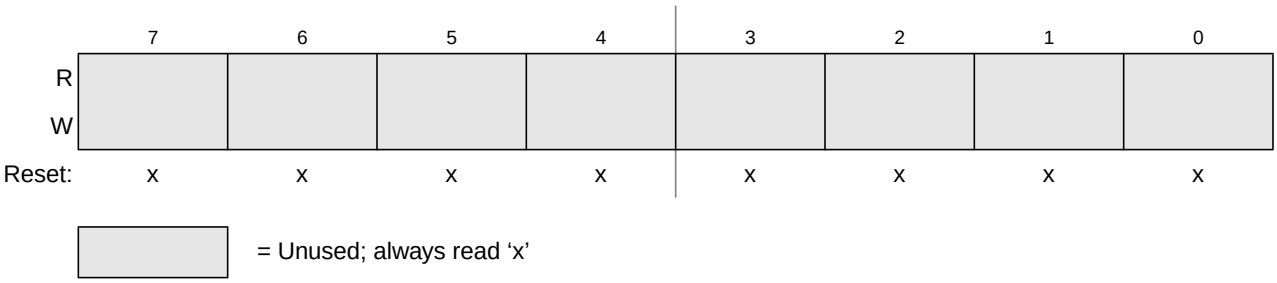



**Figure 11-33. Identifier Register 3 — Standard Mapping**

### 11.3.3.2 Data Segment Registers (DSR0-7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

Module Base + 0x00X4 to Module Base + 0x00XB

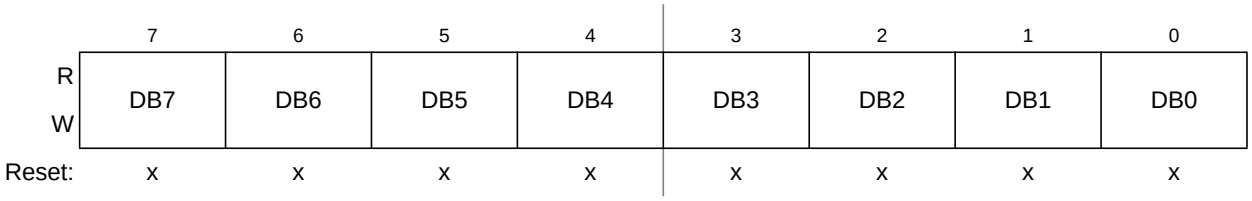



**Figure 11-34. Data Segment Registers (DSR0–DSR7) — Extended Identifier Mapping**

**Table 11-32. DSR0–DSR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>DB[7:0] | **Data bits 7-0** |

### 11.3.3.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

Module Base + 0x00XC

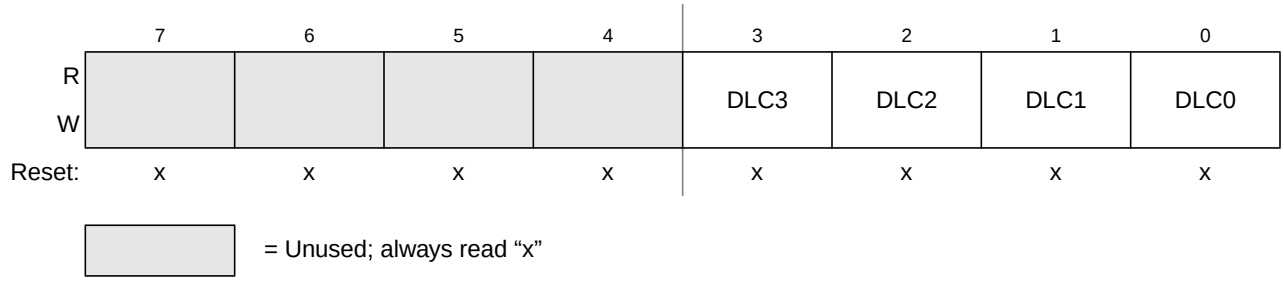



**Figure 11-35. Data Length Register (DLR) — Extended Identifier Mapping**

**Table 11-33. DLR Register Field Descriptions**

| Field | Description |
|---|---|
| 3-0<br>DLC[3:0] | **Data Length Code Bits** — The data length code contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 11-34 shows the effect of setting the DLC bits. |

**Table 11-34. Data Length Codes**

| Data Length Code | | | | Data Byte Count |
|---|---|---|---|---|
| DLC3 | DLC2 | DLC1 | DLC0 | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |

### 11.3.3.4 Transmit Buffer Priority Register (TBPR)

This register defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN and is defined to be highest for the smallest binary number. The MSCAN implements the following internal prioritization mechanisms:

- All transmission buffers with a cleared TXEx flag participate in the prioritization immediately before the SOF (start of frame) is sent.

- The transmission buffer with the lowest local priority field wins the prioritization.

In cases of more than one buffer having the same lowest priority, the message buffer with the lower index number wins.

Module Base + 0x00XD — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO7 | PRIO6 | PRIO5 | PRIO4 | PRIO3 | PRIO2 | PRIO1 | PRIO0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-36. Transmit Buffer Priority Register (TBPR)**

1. Read: Anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)")
   Write: Anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)")

### 11.3.3.5 Time Stamp Register (TSRH–TSRL)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus (see Section 11.3.2.1, "MSCAN Control Register 0 (CANCTL0)"). In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Module Base + 0x00XE — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TSR15 | TSR14 | TSR13 | TSR12 | TSR11 | TSR10 | TSR9 | TSR8 |
| W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-37. Time Stamp Register — High Byte (TSRH)**

1. Read: For transmit buffers: Anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). For receive buffers: Anytime when RXF is set.
   Write: Unimplemented

Module Base + 0x00XF

Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TSR7 | TSR6 | TSR5 | TSR4 | TSR3 | TSR2 | TSR1 | TSR0 |
| W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 11-38. Time Stamp Register — Low Byte (TSRL)**

1. Read: or transmit buffers: Anytime when TXEx flag is set (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). For receive buffers: Anytime when RXF is set.
   Write: Unimplemented

## 11.4 Functional Description

### 11.4.1 General

This section provides a complete functional description of the MSCAN.

### 11.4.2 Message Storage

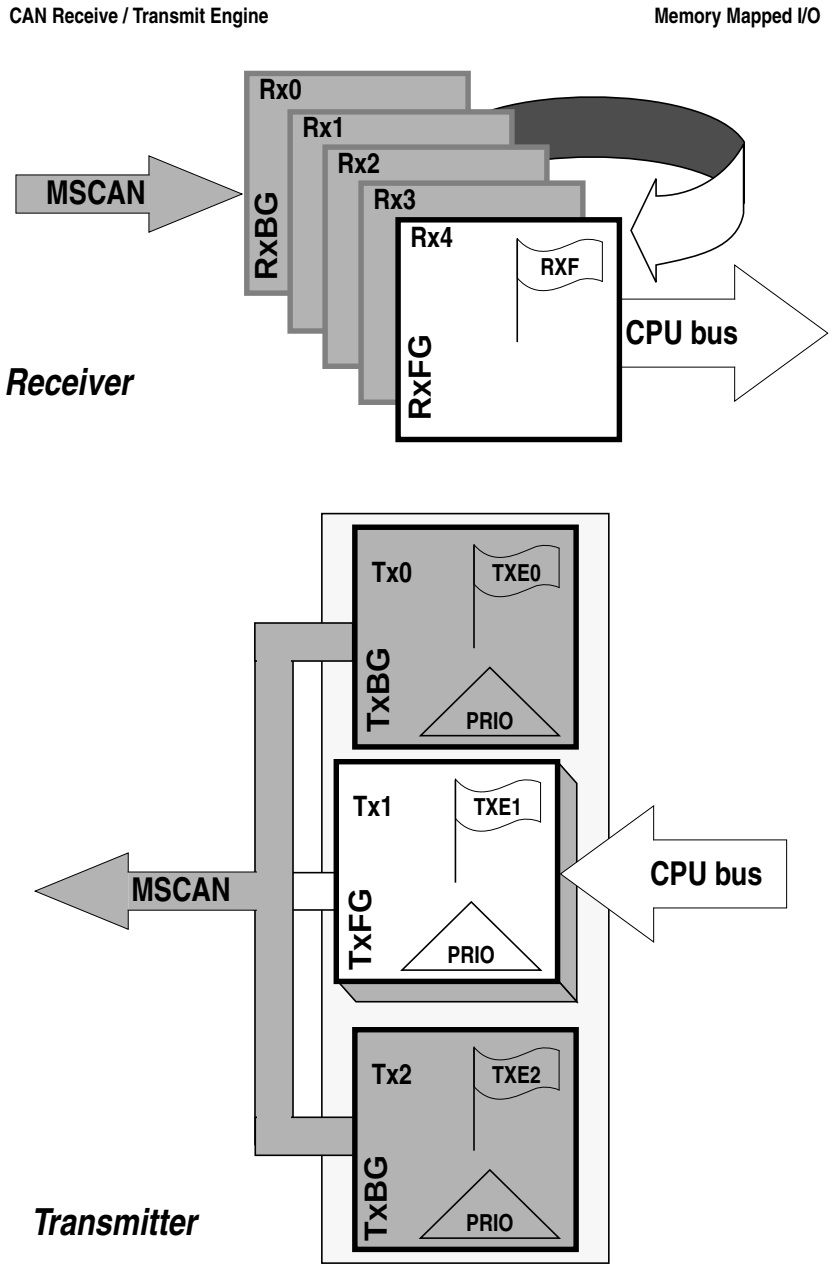



**Figure 11-39. User Model for Message Buffer Organization**

The MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

### 11.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and only release the CAN bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactiveness requirements of the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the "local priority" concept described in Section 11.4.2.2, "Transmit Structures."

### 11.4.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance. The three buffers are arranged as shown in Figure 11-39.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 11.3.3, "Programmer's Model of Message Storage"). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO) (see Section 11.3.3.4, "Transmit Buffer Priority Register (TBPR)"). The remaining two bytes are used for time stamping of a message, if required (see Section 11.3.3.5, "Time Stamp Register (TSRH–TSRL)").

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see Section 11.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)"). If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CANTBSEL register (see Section 11.3.2.11, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). This makes the respective buffer accessible within the CANTXFG address space (see Section 11.3.3, "Programmer's Model of Message Storage"). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler

software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 11.4.7.2, "Transmit Interrupt") is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 11.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)".) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

### 11.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see Figure 11-39). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 11-39). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see Section 11.3.3, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 11.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)") signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see Section 11.4.3, "Identifier Acceptance Filter") and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO, sets the RXF flag, and

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.

generates a receive interrupt[1] (see Section 11.4.7.3, "Receive Interrupt") to the CPU. The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode (see Section 11.3.2.2, "MSCAN Control Register 1 (CANCTL1)") where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled (see Section 11.4.7.5, "Error Interrupt"). The MSCAN remains able to transmit messages while the receiver FIFO is being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

### 11.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see Section 11.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)") define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked 'don't care' in the MSCAN identifier mask registers (see Section 11.3.2.18, "MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)").

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CANIDAC register (see Section 11.3.2.12, "MSCAN Identifier Acceptance Control Register (CANIDAC)"). These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters.

1. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

Figure 11-40 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

- Four identifier acceptance filters, each to be applied to:
  - The 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages.
  - The 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 11-41 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier.
  Figure 11-42 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

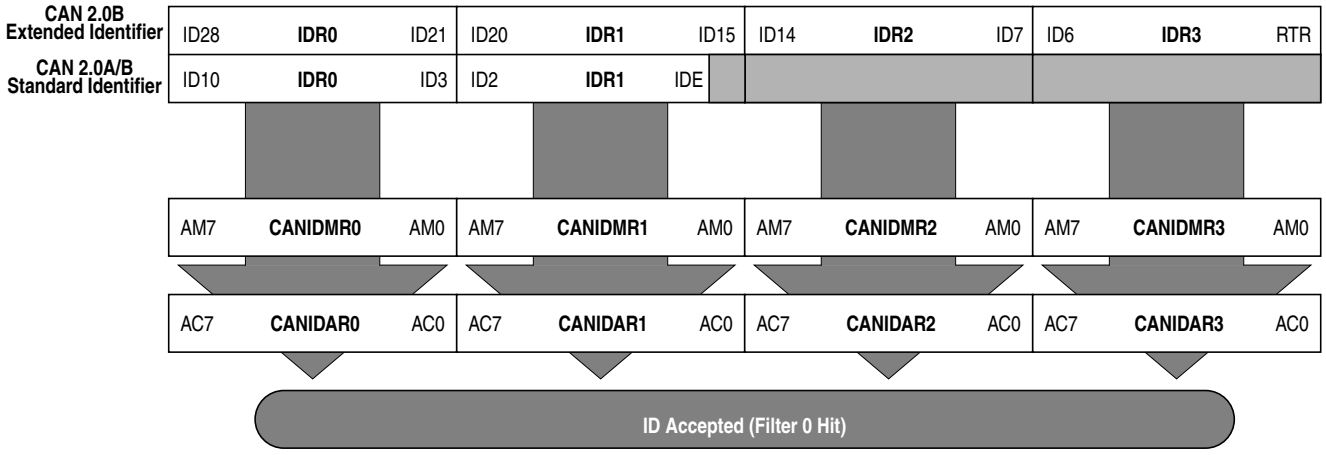



**Figure 11-40. 32-bit Maskable Identifier Acceptance Filter**

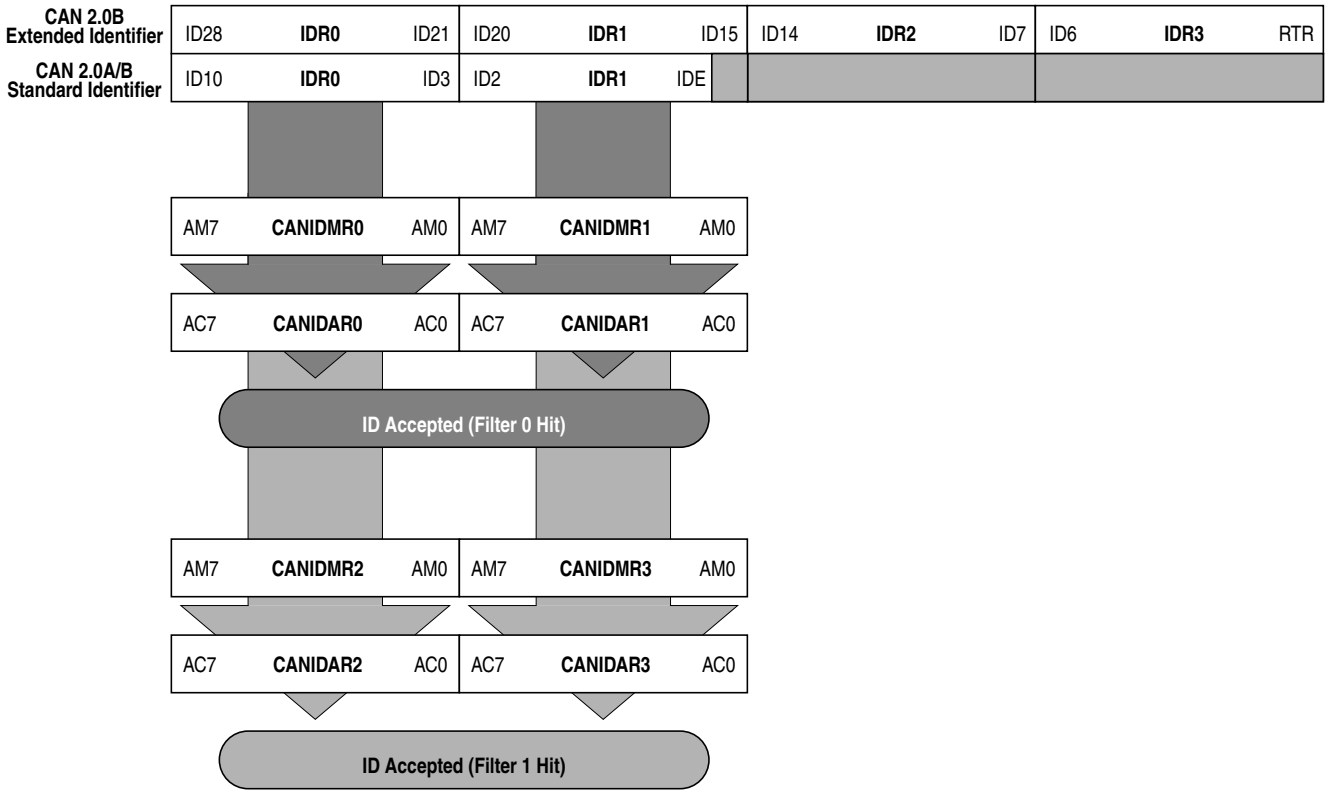



**Figure 11-41. 16-bit Maskable Identifier Acceptance Filters**

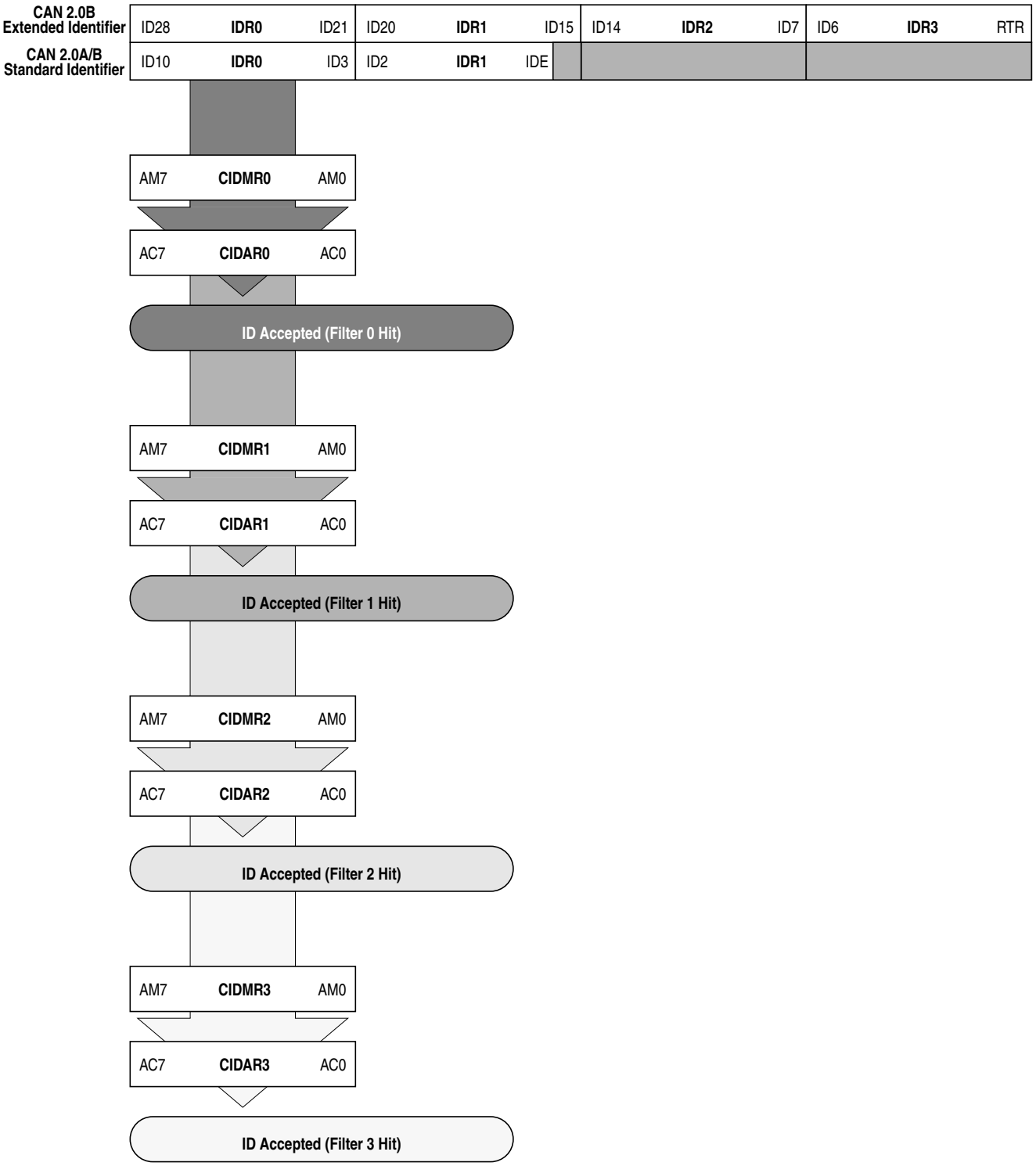



**Figure 11-42. 8-bit Maskable Identifier Acceptance Filters**

### 11.4.3.1 Protocol Violation Protection

The MSCAN protects the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN has to be in Initialization Mode. The corresponding INITRQ/INITAK handshake bits in the CANCTL0/CANCTL1 registers (see Section 11.3.2.1, "MSCAN Control Register 0 (CANCTL0)") serve as a lock to protect the following registers:
  - MSCAN control 1 register (CANCTL1)
  - MSCAN bus timing registers 0 and 1 (CANBTR0, CANBTR1)
  - MSCAN identifier acceptance control register (CANIDAC)
  - MSCAN identifier acceptance registers (CANIDAR0–CANIDAR7)
  - MSCAN identifier mask registers (CANIDMR0–CANIDMR7)
- The TXCAN is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode (see Section 11.4.5.6, "MSCAN Power Down Mode," and Section 11.4.4.5, "MSCAN Initialization Mode").
- The MSCAN enable bit (CANE) is writable only once in normal system operation modes, which provides further protection against inadvertently disabling the MSCAN.

### 11.4.3.2 Clock System

Figure 11-43 shows the structure of the MSCAN clock generation circuitry.

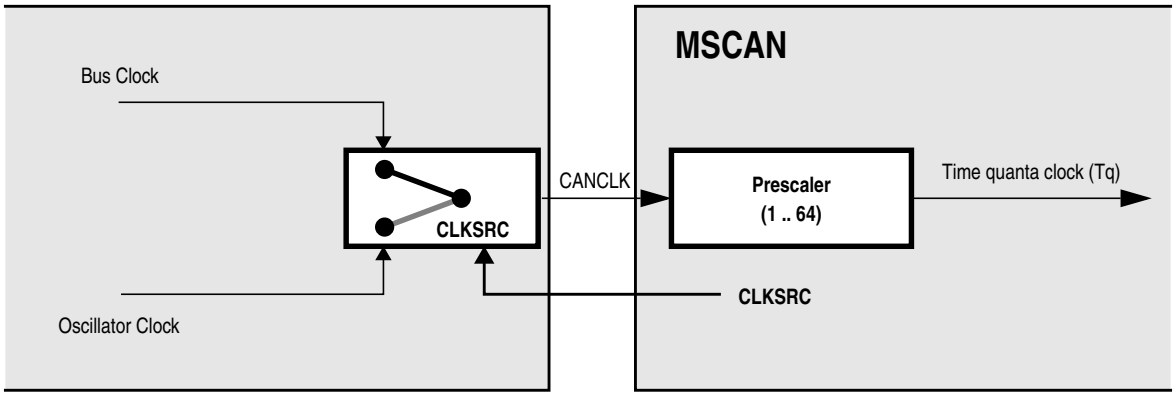



**Figure 11-43. MSCAN Clocking Scheme**

The clock source bit (CLKSRC) in the CANCTL1 register (11.3.2.2/11-423) defines whether the internal CANCLK is connected to the output of a crystal oscillator (oscillator clock) or to the bus clock.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 45% to 55% duty cycle of the clock is required.

If the bus clock is generated from a PLL, it is recommended to select the oscillator clock rather than the bus clock due to jitter considerations, especially at the faster CAN bus rates.

For microcontrollers without a clock and reset generator (CRG), CANCLK is driven from the crystal oscillator (oscillator clock).

A programmable prescaler generates the time quanta (Tq) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

***Eqn. 11-2***

$$Tq = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments as described in the Bosch CAN 2.0A/B specification. (see Figure 11-44):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

***Eqn. 11-3***

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

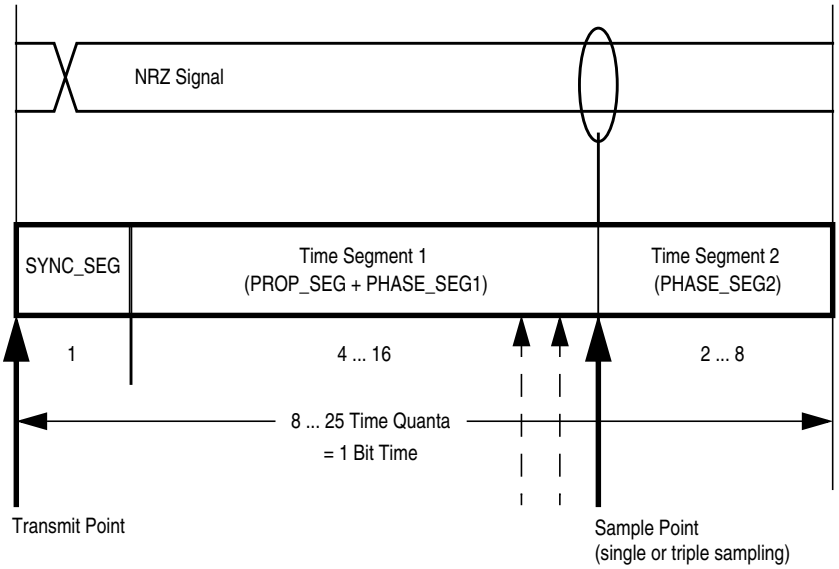



**Figure 11-44. Segments within the Bit Time**

**Table 11-35. Time Segment Syntax**

| Syntax | Description |
|---|---|
| SYNC_SEG | System expects transitions to occur on the CAN bus during this period. |
| Transmit Point | A node in transmit mode transfers a new value to the CAN bus at this point. |
| Sample Point | A node in receive mode samples the CAN bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample. |

The synchronization jump width (see the Bosch CAN 2.0A/B specification for details) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The SYNC_SEG, TSEG1, TSEG2, and SJW parameters are set by programming the MSCAN bus timing registers (CANBTR0, CANBTR1) (see Section 11.3.2.3, "MSCAN Bus Timing Register 0 (CANBTR0)" and Section 11.3.2.4, "MSCAN Bus Timing Register 1 (CANBTR1)").

Table 11-36 gives an overview of the Bosch CAN 2.0A/B specification compliant segment settings and the related parameter values.

**NOTE**

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard.

**Table 11-36. Bosch CAN 2.0A/B Compliant Bit Time Segment Settings**

| Time Segment 1 | TSEG1 | Time Segment 2 | TSEG2 | Synchronization Jump Width | SJW |
|---|---|---|---|---|---|
| 5 .. 10 | 4 .. 9 | 2 | 1 | 1 .. 2 | 0 .. 1 |
| 4 .. 11 | 3 .. 10 | 3 | 2 | 1 .. 3 | 0 .. 2 |
| 5 .. 12 | 4 .. 11 | 4 | 3 | 1 .. 4 | 0 .. 3 |
| 6 .. 13 | 5 .. 12 | 5 | 4 | 1 .. 4 | 0 .. 3 |
| 7 .. 14 | 6 .. 13 | 6 | 5 | 1 .. 4 | 0 .. 3 |
| 8 .. 15 | 7 .. 14 | 7 | 6 | 1 .. 4 | 0 .. 3 |
| 9 .. 16 | 8 .. 15 | 8 | 7 | 1 .. 4 | 0 .. 3 |

## 11.4.4 Modes of Operation

### 11.4.4.1 Normal System Operating Modes

The MSCAN module behaves as described within this specification in all normal system operating modes. Write restrictions exist for some registers.

#### 11.4.4.2 Special System Operating Modes

The MSCAN module behaves as described within this specification in all special system operating modes. Write restrictions which exist on specific registers in normal modes are lifted for test purposes in special modes.

#### 11.4.4.3 Emulation Modes

In all emulation modes, the MSCAN module behaves just like in normal system operating modes as described within this specification.

#### 11.4.4.4 Listen-Only Mode

In an optional CAN bus monitoring mode (listen-only), the CAN node is able to receive valid data frames and valid remote frames, but it sends only "recessive" bits on the CAN bus. In addition, it cannot start a transmission.

If the MAC sub-layer is required to send a "dominant" bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the MAC sub-layer monitors this "dominant" bit, although the CAN bus may remain in recessive state externally.

#### 11.4.4.5 MSCAN Initialization Mode

The MSCAN enters initialization mode when it is enabled (CANE=1).

When entering initialization mode during operation, any on-going transmission or reception is immediately aborted and synchronization to the CAN bus is lost, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives TXCAN into a recessive state.

**NOTE**

The user is responsible for ensuring that the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPAK = 1) before setting the INITRQ bit in the CANCTL0 register. Otherwise, the abort of an on-going message can cause an error condition and can impact other CAN bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers remain accessible. This mode is used to reset the CANCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAAK, and CANTBSEL registers to their default values. In addition, the MSCAN enables the configuration of the CANBTR0, CANBTR1 bit timing registers; CANIDAC; and the CANIDAR, CANIDMR message filters. See Section 11.3.2.1, "MSCAN Control Register 0 (CANCTL0)," for a detailed description of the initialization mode.

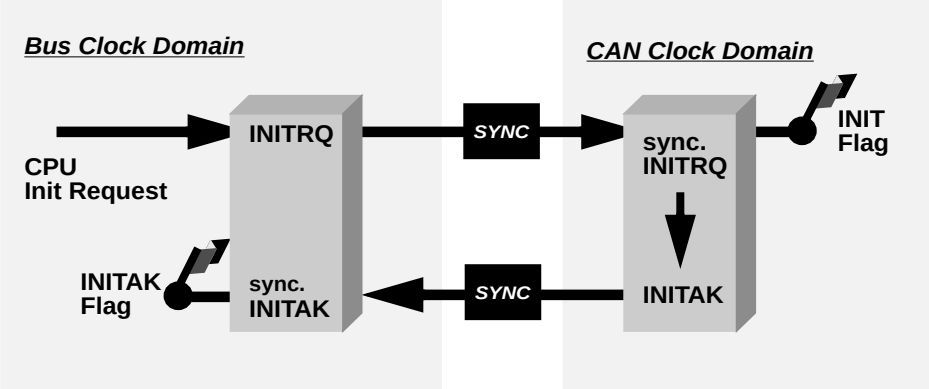



**Figure 11-45. Initialization Request/Acknowledge Cycle**

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see Figure 11-45).

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

**NOTE**

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

## 11.4.5 Low-Power Options

If the MSCAN is disabled (CANE = 0), the MSCAN clocks are stopped for power saving.

If the MSCAN is enabled (CANE = 1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In power down mode, all clocks are stopped and no power is consumed.

Table 11-37 summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

**Table 11-37. CPU vs. MSCAN Operating Modes**

| CPU Mode | MSCAN Mode | | | |
|---|---|---|---|---|
| | Normal | Reduced Power Consumption | | |
| | | Sleep | Power Down | Disabled (CANE=0) |
| **RUN** | CSWAI = X[1]<br>SLPRQ = 0<br>SLPAK = 0 | CSWAI = X<br>SLPRQ = 1<br>SLPAK = 1 | | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |
| **WAIT** | CSWAI = 0<br>SLPRQ = 0<br>SLPAK = 0 | CSWAI = 0<br>SLPRQ = 1<br>SLPAK = 1 | CSWAI = 1<br>SLPRQ = X<br>SLPAK = X | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |
| **STOP** | | | CSWAI = X<br>SLPRQ = X<br>SLPAK = X | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |

1. 'X' means don't care.

### 11.4.5.1 Operation in Run Mode

As shown in Table 11-37, only MSCAN sleep mode is available as low power option when the CPU is in run mode.

### 11.4.5.2 Operation in Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode. If the CSWAI bit is set, additional power can be saved in power down mode because the CPU clocks are stopped. After leaving this power down mode, the MSCAN restarts and enters normal mode again.

While the CPU is in wait mode, the MSCAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode).

### 11.4.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop mode, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits (Table 11-37).

### 11.4.5.4 MSCAN Normal Mode

This is a non-power-saving mode. Enabling the MSCAN puts the module from disabled mode into normal mode. In this mode the module can either be in initialization mode or out of initialization mode. See Section 11.4.4.5, "MSCAN Initialization Mode".

### 11.4.5.5 MSCAN Sleep Mode

The CPU can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CANCTL0 register. The time when the MSCAN enters sleep mode depends on a fixed synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXEx = 0), the MSCAN will continue to transmit until all transmit message buffers are empty (TXEx = 1, transmitted successfully or aborted) and then goes into sleep mode.
- If the MSCAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle.
- If the MSCAN is neither transmitting nor receiving, it immediately goes into sleep mode.

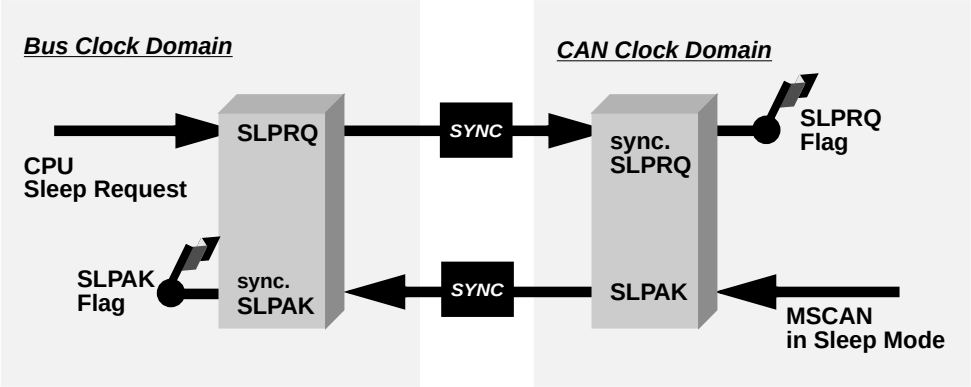



**Figure 11-46. Sleep Request / Acknowledge Cycle**

**NOTE**

The application software must avoid setting up a transmission (by clearing one or more TXEx flag(s)) and immediately request sleep mode (by setting SLPRQ). Whether the MSCAN starts transmitting or goes into sleep mode directly depends on the exact sequence of operations.

If sleep mode is active, the SLPRQ and SLPAK bits are set (Figure 11-46). The application software must use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode.

When in sleep mode (SLPRQ = 1 and SLPAK = 1), the MSCAN stops its internal clocks. However, clocks that allow register accesses from the CPU side continue to run.

If the MSCAN is in bus-off state, it stops counting the 128 occurrences of 11 consecutive recessive bits due to the stopped clocks. TXCAN remains in a recessive state. If RXF = 1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in sleep mode.

It is possible to access the transmit buffers and to clear the associated TXE flags. No message abort takes place while in sleep mode.

If the WUPE bit in CANCTL0 is not asserted, the MSCAN will mask any activity it detects on CAN. RXCAN is therefore held internally in a recessive state. This locks the MSCAN in sleep mode. WUPE must be set before entering sleep mode to take effect.

The MSCAN is able to leave sleep mode (wake up) only when:

- CAN bus activity occurs and WUPE = 1

  or
- the CPU clears the SLPRQ bit

**NOTE**

The CPU cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPAK = 1) is active.

After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received.

The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions will be executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

#### 11.4.5.6 MSCAN Power Down Mode

The MSCAN is in power down mode (Table 11-37) when

- CPU is in stop mode

  or
- CPU is in wait mode and the CSWAI bit is set

When entering the power down mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives TXCAN into a recessive state.

**NOTE**

The user is responsible for ensuring that the MSCAN is not active when power down mode is entered. The recommended procedure is to bring the MSCAN into Sleep mode before the STOP or WAI instruction (if CSWAI is set) is executed. Otherwise, the abort of an ongoing message can cause an error condition and impact other CAN bus devices.

In power down mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in sleep mode before power down mode became active, the module performs an internal recovery cycle after powering up. This causes some fixed delay before the module enters normal mode again.

#### 11.4.5.7 Disabled Mode

The MSCAN is in disabled mode out of reset (CANE=0). All module clocks are stopped for power saving, however the register map can still be accessed as specified.

#### 11.4.5.8 Programmable Wake-Up Function

The MSCAN can be programmed to wake up from sleep or power down mode as soon as CAN bus activity is detected (see control bit WUPE in MSCAN Control Register 0 (CANCTL0). The sensitivity to existing CAN bus action can be modified by applying a low-pass filter function to the RXCAN input line (see control bit WUPM in Section 11.3.2.2, "MSCAN Control Register 1 (CANCTL1)").

This feature can be used to protect the MSCAN from wake-up due to short glitches on the CAN bus lines. Such glitches can result from—for example—electromagnetic interference within noisy environments.

### 11.4.6 Reset Initialization

The reset state of each individual bit is listed in Section 11.3.2, "Register Descriptions," which details all the registers and their bit-fields.

### 11.4.7 Interrupts

This section describes all interrupts originated by the MSCAN. It documents the enable bits and generated flags. Each interrupt is listed and described separately.

#### 11.4.7.1 Description of Interrupt Operation

The MSCAN supports four interrupt vectors (see Table 11-38), any of which can be individually masked (for details see Section 11.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)" to Section 11.3.2.8, "MSCAN Transmitter Interrupt Enable Register (CANTIER)").

Refer to the device overview section to determine the dedicated interrupt vector addresses.

**Table 11-38. Interrupt Vectors**

| Interrupt Source | CCR Mask | Local Enable |
|---|---|---|
| Wake-Up Interrupt (WUPIF) | I bit | CANRIER (WUPIE) |
| Error Interrupts Interrupt (CSCIF, OVRIF) | I bit | CANRIER (CSCIE, OVRIE) |
| Receive Interrupt (RXF) | I bit | CANRIER (RXFIE) |
| Transmit Interrupts (TXE[2:0]) | I bit | CANTIER (TXEIE[2:0]) |

#### 11.4.7.2 Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXEx flag of the empty message buffer is set.

#### 11.4.7.3 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

#### 11.4.7.4 Wake-Up Interrupt

A wake-up interrupt is generated if activity on the CAN bus occurs during MSCAN sleep or power-down mode.

**NOTE**

This interrupt can only occur if the MSCAN was in sleep mode (SLPRQ = 1 and SLPAK = 1) before entering power down mode, the wake-up option is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

#### 11.4.7.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurs. MSCAN Receiver Flag Register (CANRFLG) indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in Section 11.4.2.3, "Receive Structures," occurred.
- **CAN Status Change** — The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags (see Section 11.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)" and Section 11.3.2.6, "MSCAN Receiver Interrupt Enable Register (CANRIER)").

#### 11.4.7.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN Receiver Flag Register (CANRFLG) or the MSCAN Transmitter Flag Register (CANTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in CANRFLG and CANTFLG must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition prevails.

**NOTE**

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

# 11.5 Initialization/Application Information

## 11.5.1 MSCAN initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE
2. Write to the configuration registers in initialization mode
3. Clear INITRQ to leave initialization mode

If the configuration of registers which are only writable in initialization mode shall be changed:

1. Bring the module into sleep mode by setting SLPRQ and awaiting SLPAK to assert after the CAN bus becomes idle.
2. Enter initialization mode: assert INITRQ and await INITAK
3. Write to the configuration registers in initialization mode
4. Clear INITRQ to leave initialization mode and continue

## 11.5.2 Bus-Off Recovery

The bus-off recovery is user configurable. The bus-off state can either be left automatically or on user request.

For reasons of backwards compatibility, the MSCAN defaults to automatic recovery after reset. In this case, the MSCAN will become error active again after counting 128 occurrences of 11 consecutive recessive bits on the CAN bus (see the Bosch CAN 2.0 A/B specification for details).

If the MSCAN is configured for user request (BORM set in MSCAN Control Register 1 (CANCTL1)), the recovery from bus-off starts after both independent events have become true:

- 128 occurrences of 11 consecutive recessive bits on the CAN bus have been monitored
- BOHOLD in MSCAN Miscellaneous Register (CANMISC) has been cleared by the user

These two events may occur in any order.

# Chapter 12
# Serial Communication Interface (S12SCIV6)

**Table 12-1. Revision History**

| Version Number | Revision Date | Effective Date | Author | Description of Changes |
|---|---|---|---|---|
| 05.03 | 12/25/2008 | | | remove redundancy comments in Figure1-2 |
| 05.04 | 08/05/2009 | | | fix typo, SCIBDL reset value be 0x04, not 0x00 |
| 05.05 | 06/03/2010 | | | fix typo, Table 12-3,SCICR1 Even parity should be PT=0<br>fix typo, on page 12-491,should be BKDIF,not BLDIF |
| 06.01 | 05/29/2012 | | | update register map, change BD,move IREN to SCIACR2 |
| 06.02 | 10/17/2012 | | | fix typo on page 12-474 and on page 12-474;fix typo of version V6<br>update fast data tolerance calculation and add notes. |
| 06.03 | 10/25/2012 | | | fix typo Table 12-2, SBR[15:4],not SBR[15:0] |
| 06.04 | 12/19/2012 | | | fix typo Table 12-6,12.4.1/12-486 |
| 06.05 | 02/22/2013 | | | fix typo Figure 12-1./12-471 Figure 12-4./12-474<br>update Table 12-2./12-474 12.4.4/12-488 12.4.6.3/12-495 |
| 06.06 | 03/11/2013 | | | fix typo of BDL reset value,Figure 12-4<br>fix typo of Table 12-2,Table 12-16,reword 12.4.4/12-488 |
| 06.07 | 09/03/2013 | | | update Figure 12-14./12-485 Figure 12-16./12-489<br>Figure 12-20./12-494<br>update 12.4.4/12-488,more detail for two baud<br>add note for Table 12-16./12-488<br>update Figure 12-2./12-473,Figure 12-12./12-484 |

## 12.1 Introduction

This block guide provides an overview of the serial communication interface (SCI) module.

The SCI allows asynchronous serial communications with peripheral devices and other CPUs.

### 12.1.1 Glossary

IR: InfraRed

IrDA: Infrared Design Associate

IRQ: Interrupt Request

LIN: Local Interconnect Network

LSB: Least Significant Bit

MSB: Most Significant Bit

NRZ: Non-Return-to-Zero

RZI: Return-to-Zero-Inverted

RXD: Receive Pin

SCI : Serial Communication Interface

TXD: Transmit Pin

### 12.1.2 Features

The SCI includes these distinctive features:

- Full-duplex or single-wire operation
- Standard mark/space non-return-to-zero (NRZ) format
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse widths
- 16-bit baud rate selection
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable polarity for transmitter and receiver
- Programmable transmitter output parity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Interrupt-driven operation with eight flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
  - Receive wakeup on active edge
  - Transmit collision detect supporting LIN
  - Break Detect supporting LIN
- Receiver framing error detection
- Hardware parity checking

- 1/16 bit-time noise detection

### 12.1.3 Modes of Operation

The SCI functions the same in normal, special, and emulation modes. It has two low power modes, wait and stop modes.

- Run mode
- Wait mode
- Stop mode

### 12.1.4 Block Diagram

Figure 12-1 is a high level block diagram of the SCI module, showing the interaction of various function blocks.

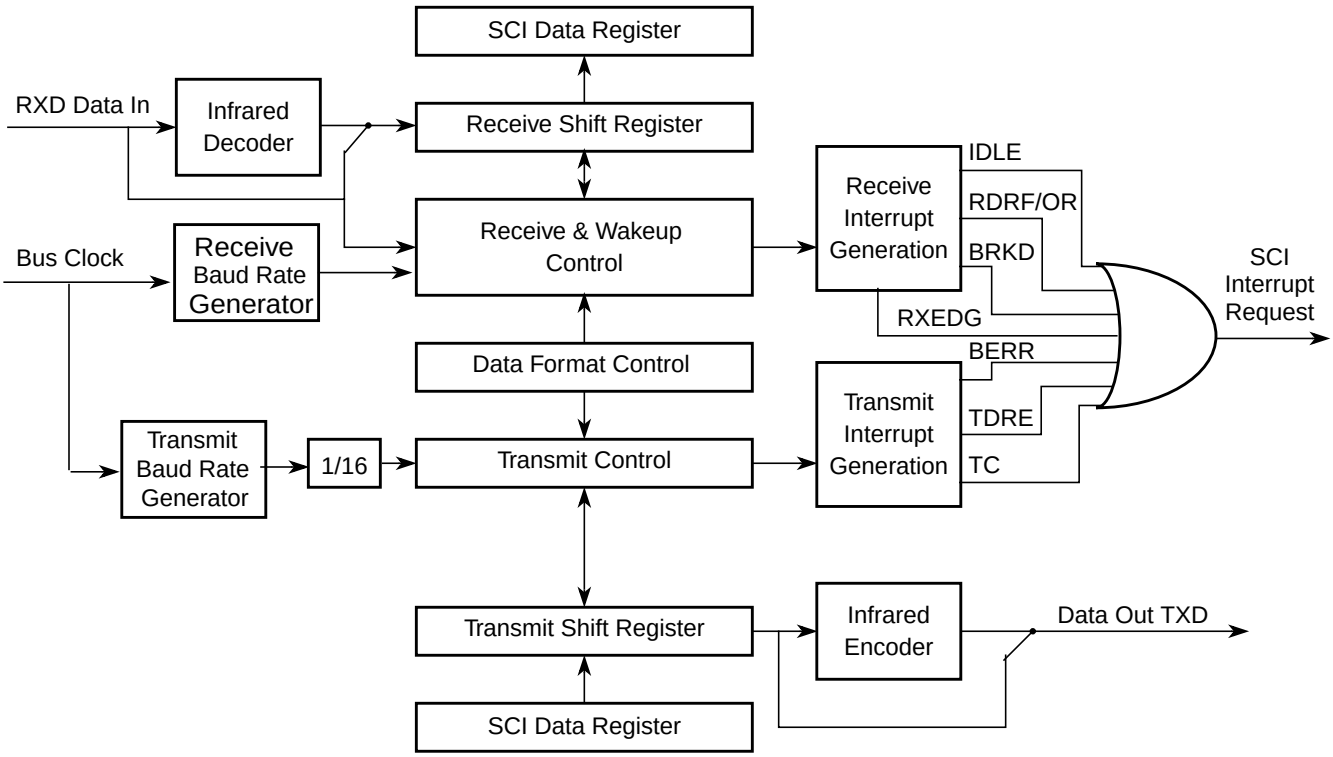



**Figure 12-1. SCI Block Diagram**

## 12.2 External Signal Description

The SCI module has a total of two external pins.

### 12.2.1 TXD — Transmit Pin

The TXD pin transmits SCI (standard or infrared) data. It will idle high in either mode and is high impedance anytime the transmitter is disabled.

### 12.2.2 RXD — Receive Pin

The RXD pin receives SCI (standard or infrared) data. An idle line is detected as a line high. This input is ignored when the receiver is disabled and should be terminated to a known voltage.

## 12.3 Memory Map and Register Definition

This section provides a detailed description of all the SCI registers.

### 12.3.1 Module Memory Map and Register Definition

The memory map for the SCI module is given below in Figure 12-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the SCI module and the address offset for each register.

## 12.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Writes to a reserved register locations do not have any effect and reads of these locations return a zero. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 SCIBDH[1] | R | SBR15 | SBR14 | SBR13 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | W | | | | | | | | |
| 0x0001 SCIBDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | W | | | | | | | | |
| 0x0002 SCICR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | W | | | | | | | | |
| 0x0000 SCIASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | W | | | | | | | | |
| 0x0001 SCIACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | W | | | | | | | | |
| 0x0002 SCIACR2[2] | R | IREN | TNP1 | TNP0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | W | | | | | | | | |
| 0x0003 SCICR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | W | | | | | | | | |
| 0x0004 SCISR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | W | | | | | | | | |
| 0x0005 SCISR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | W | | | | | | | | |
| 0x0006 SCIDRH | R | R8 | T8 | 0 | 0 | 0 | Reserved | Reserved | Reserved |
| | W | | | | | | | | |
| 0x0007 SCIDRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1.These registers are accessible if the AMAP bit in the SCISR2 register is set to zero.

2,These registers are accessible if the AMAP bit in the SCISR2 register is set to one.

= Unimplemented or Reserved

**Figure 12-2. SCI Register Summary**

### 12.3.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SBR15 | SBR14 | SBR13 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-3. SCI Baud Rate Register (SCIBDH)**

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-4. SCI Baud Rate Register (SCIBDL)**

Read: Anytime, if AMAP = 0.

Write: Anytime, if AMAP = 0.

**NOTE**

Those two registers are only visible in the memory map if AMAP = 0 (reset condition).

The SCI baud rate register is used by to determine the baud rate of the SCI, and to control the infrared modulation/demodulation submodule.

**Table 12-2. SCIBDH and SCIBDL Field Descriptions**

| Field | Description |
|---|---|
| SBR[15:0] | **SCI Baud Rate Bits** — The baud rate for the SCI is determined by the bits in this register. The baud rate is calculated two different ways depending on the state of the IREN bit.<br>The formulas for calculating the baud rate are:<br>When IREN = 0 then,<br>SCI baud rate = SCI bus clock / (SBR[15:0])<br>When IREN = 1 then,<br>SCI baud rate = SCI bus clock / (2 x SBR[15:1])<br>**Note:** The baud rate generator is disabled after reset and not started until the TE bit or the RE bit is set for the first time. The baud rate generator is disabled when (SBR[15:4] = 0 and IREN = 0) or (SBR[15:5] = 0 and IREN = 1).<br>**Note:** . User should write SCIBD by word access. The updated SCIBD may take effect until next RT clock start, write SCIBDH or SCIBDL separately may cause baud generator load wrong data at that time,if second write later then RT clock. |

### 12.3.2.2 SCI Control Register 1 (SCICR1)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-5. SCI Control Register 1 (SCICR1)**

Read: Anytime, if AMAP = 0.

Write: Anytime, if AMAP = 0.

**NOTE**

This register is only visible in the memory map if AMAP = 0 (reset condition).

**Table 12-3. SCICR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LOOPS | **Loop Select Bit** — LOOPS enables loop operation. In loop operation, the RXD pin is disconnected from the SCI and the transmitter output is internally connected to the receiver input. Both the transmitter and the receiver must be enabled to use the loop function.<br>0 Normal operation enabled<br>1 Loop operation enabled<br>The receiver input is determined by the RSRC bit. |
| 6<br>SCISWAI | **SCI Stop in Wait Mode Bit** — SCISWAI disables the SCI in wait mode.<br>0 SCI enabled in wait mode<br>1 SCI disabled in wait mode |
| 5<br>RSRC | **Receiver Source Bit** — When LOOPS = 1, the RSRC bit determines the source for the receiver shift register input. See Table 12-4.<br>0 Receiver input internally connected to transmitter output<br>1 Receiver input connected externally to transmitter |
| 4<br>M | **Data Format Mode Bit** — MODE determines whether data characters are eight or nine bits long.<br>0 One start bit, eight data bits, one stop bit<br>1 One start bit, nine data bits, one stop bit |
| 3<br>WAKE | **Wakeup Condition Bit** — WAKE determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on the RXD pin.<br>0 Idle line wakeup<br>1 Address mark wakeup |
| 2<br>ILT | **Idle Line Type Bit** — ILT determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>0 Idle character bit count begins after start bit<br>1 Idle character bit count begins after stop bit |

**Table 12-3. SCICR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>PE | **Parity Enable Bit** — PE enables the parity function. When enabled, the parity function inserts a parity bit in the most significant bit position.<br>0 Parity function disabled<br>1 Parity function enabled |
| 0<br>PT | **Parity Type Bit** — PT determines whether the SCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit.<br>0 Even parity<br>1 Odd parity |

**Table 12-4. Loop Functions**

| LOOPS | RSRC | Function |
|---|---|---|
| 0 | x | Normal operation |
| 1 | 0 | Loop mode with transmitter output internally connected to receiver input |
| 1 | 1 | Single-wire mode with TXD pin connected to receiver input |

### 12.3.2.3 SCI Alternative Status Register 1 (SCIASR1)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-6. SCI Alternative Status Register 1 (SCIASR1)**

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

**Table 12-5. SCIASR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RXEDGIF | **Receive Input Active Edge Interrupt Flag** — RXEDGIF is asserted, if an active edge (falling if RXPOL = 0, rising if RXPOL = 1) on the RXD input occurs. RXEDGIF bit is cleared by writing a “1” to it.<br>0 No active receive on the receive input has occurred<br>1 An active edge on the receive input has occurred |
| 2<br>BERRV | **Bit Error Value** — BERRV reflects the state of the RXD input when the bit error detect circuitry is enabled and a mismatch to the expected value happened. The value is only meaningful, if BERRIF = 1.<br>0 A low input was sampled, when a high was expected<br>1 A high input reassembled, when a low was expected |
| 1<br>BERRIF | **Bit Error Interrupt Flag** — BERRIF is asserted, when the bit error detect circuitry is enabled and if the value sampled at the RXD input does not match the transmitted value. If the BERRIE interrupt enable bit is set an interrupt will be generated. The BERRIF bit is cleared by writing a “1” to it.<br>0 No mismatch detected<br>1 A mismatch has occurred |
| 0<br>BKDIF | **Break Detect Interrupt Flag** — BKDIF is asserted, if the break detect circuitry is enabled and a break signal is received. If the BKDIE interrupt enable bit is set an interrupt will be generated. The BKDIF bit is cleared by writing a “1” to it.<br>0 No break signal was received<br>1 A break signal was received |

### 12.3.2.4 SCI Alternative Control Register 1 (SCIACR1)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-7. SCI Alternative Control Register 1 (SCIACR1)**

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

**Table 12-6. SCIACR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RXEDGIE | **Receive Input Active Edge Interrupt Enable** — RXEDGIE enables the receive input active edge interrupt flag, RXEDGIF, to generate interrupt requests.<br>0 RXEDGIF interrupt requests disabled<br>1 RXEDGIF interrupt requests enabled |
| 1<br>BERRIE | **Bit Error Interrupt Enable** — BERRIE enables the bit error interrupt flag, BERRIF, to generate interrupt requests.<br>0 BERRIF interrupt requests disabled<br>1 BERRIF interrupt requests enabled |
| 0<br>BKDIE | **Break Detect Interrupt Enable** — BKDIE enables the break detect interrupt flag, BKDIF, to generate interrupt requests.<br>0 BKDIF interrupt requests disabled<br>1 BKDIF interrupt requests enabled |

### 12.3.2.5 SCI Alternative Control Register 2 (SCIACR2)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IREN | TNP1 | TNP0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-8. SCI Alternative Control Register 2 (SCIACR2)**

Read: Anytime, if AMAP = 1

Write: Anytime, if AMAP = 1

**Table 12-7. SCIACR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IREN | **Infrared Enable Bit** — This bit enables/disables the infrared modulation/demodulation submodule.<br>0 IR disabled<br>1 IR enabled |
| 6:5<br>TNP[1:0] | **Transmitter Narrow Pulse Bits** — These bits enable whether the SCI transmits a 1/16, 3/16, 1/32 or 1/4 narrow pulse. See Table 12-8. |
| 2:1<br>BERRM[1:0] | **Bit Error Mode** — Those two bits determines the functionality of the bit error detect feature. See Table 12-9. |
| 0<br>BKDFE | **Break Detect Feature Enable** — BKDFE enables the break detect circuitry.<br>0 Break detect circuit disabled<br>1 Break detect circuit enabled |

**Table 12-8. IRSCI Transmit Pulse Width**

| TNP[1:0] | Narrow Pulse Width |
|---|---|
| 11 | 1/4 |
| 10 | 1/32 |
| 01 | 1/16 |
| 00 | 3/16 |

**Table 12-9. Bit Error Mode Coding**

| BERRM1 | BERRM0 | Function |
|---|---|---|
| 0 | 0 | Bit error detect circuit is disabled |
| 0 | 1 | Receive input sampling occurs during the 9th time tick of a transmitted bit (refer to Figure 12-19) |
| 1 | 0 | Receive input sampling occurs during the 13th time tick of a transmitted bit (refer to Figure 12-19) |
| 1 | 1 | Reserved |

### 12.3.2.6 SCI Control Register 2 (SCICR2)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-9. SCI Control Register 2 (SCICR2)**

Read: Anytime

Write: Anytime

**Table 12-10. SCICR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TIE | **Transmitter Interrupt Enable Bit** — TIE enables the transmit data register empty flag, TDRE, to generate interrupt requests.<br>0 TDRE interrupt requests disabled<br>1 TDRE interrupt requests enabled |
| 6<br>TCIE | **Transmission Complete Interrupt Enable Bit** — TCIE enables the transmission complete flag, TC, to generate interrupt requests.<br>0 TC interrupt requests disabled<br>1 TC interrupt requests enabled |
| 5<br>RIE | **Receiver Full Interrupt Enable Bit** — RIE enables the receive data register full flag, RDRF, or the overrun flag, OR, to generate interrupt requests.<br>0 RDRF and OR interrupt requests disabled<br>1 RDRF and OR interrupt requests enabled |
| 4<br>ILIE | **Idle Line Interrupt Enable Bit** — ILIE enables the idle line flag, IDLE, to generate interrupt requests.<br>0 IDLE interrupt requests disabled<br>1 IDLE interrupt requests enabled |
| 3<br>TE | **Transmitter Enable Bit** — TE enables the SCI transmitter and configures the TXD pin as being controlled by the SCI. The TE bit can be used to queue an idle preamble.<br>0 Transmitter disabled<br>1 Transmitter enabled |
| 2<br>RE | **Receiver Enable Bit** — RE enables the SCI receiver.<br>0 Receiver disabled<br>1 Receiver enabled |
| 1<br>RWU | **Receiver Wakeup Bit** — Standby state<br>0 Normal operation.<br>1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU. |
| 0<br>SBK | **Send Break Bit** — Toggling SBK sends one break character (10 or 11 logic 0s, respectively 13 or 14 logics 0s if BRK13 is set). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters (10 or 11 bits, respectively 13 or 14 bits).<br>0 No break characters<br>1 Transmit break characters |

### 12.3.2.7 SCI Status Register 1 (SCISR1)

The SCISR1 and SCISR2 registers provides inputs to the MCU for generation of SCI interrupts. Also, these registers can be polled by the MCU to check the status of these bits. The flag-clearing procedures require that the status register be read followed by a read or write to the SCI data register.It is permissible to execute other instructions between the two steps as long as it does not compromise the handling of I/O, but the order of operations is important for flag clearing.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-10. SCI Status Register 1 (SCISR1)**

Read: Anytime

Write: Has no meaning or effect

**Table 12-11. SCISR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TDRE | **Transmit Data Register Empty Flag** — TDRE is set when the transmit shift register receives a byte from the SCI data register. When TDRE is 1, the transmit data register (SCIDRH/L) is empty and can receive a new value to transmit.Clear TDRE by reading SCI status register 1 (SCISR1), with TDRE set and then writing to SCI data register low (SCIDRL).<br>0 No byte transferred to transmit shift register<br>1 Byte transferred to transmit shift register; transmit data register empty |
| 6<br>TC | **Transmit Complete Flag** — TC is set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted.When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL). TC is cleared automatically when data, preamble, or break is queued and ready to be sent. TC is cleared in the event of a simultaneous set and clear of the TC flag (transmission not complete).<br>0 Transmission in progress<br>1 No transmission in progress |
| 5<br>RDRF | **Receive Data Register Full Flag** — RDRF is set when the data in the receive shift register transfers to the SCI data register. Clear RDRF by reading SCI status register 1 (SCISR1) with RDRF set and then reading SCI data register low (SCIDRL).<br>0 Data not available in SCI data register<br>1 Received data available in SCI data register |
| 4<br>IDLE | **Idle Line Flag** — IDLE is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M =1) appear on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag.Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL).<br>0 Receiver input is either active now or has never become active since the IDLE flag was last cleared<br>1 Receiver input has become idle<br>**Note:** When the receiver wakeup bit (RWU) is set, an idle line condition does not set the IDLE flag. |

**Table 12-11. SCISR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>OR | **Overrun Flag** — OR is set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. Clear OR by reading SCI status register 1 (SCISR1) with OR set and then reading SCI data register low (SCIDRL).<br>0 No overrun<br>1 Overrun<br>**Note:** OR flag may read back as set when RDRF flag is clear. This may happen if the following sequence of events occurs:<br>1. After the first frame is received, read status register SCISR1 (returns RDRF set and OR flag clear);<br>2. Receive second frame without reading the first frame in the data register (the second frame is not received and OR flag is set);<br>3. Read data register SCIDRL (returns first frame and clears RDRF flag in the status register);<br>4. Read status register SCISR1 (returns RDRF clear and OR set).<br>Event 3 may be at exactly the same time as event 2 or any time after. When this happens, a dummy SCIDRL read following event 4 will be required to clear the OR flag if further frames are to be received. |
| 2<br>NF | **Noise Flag** — NF is set when the SCI detects noise on the receiver input. NF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear NF by reading SCI status register 1(SCISR1), and then reading SCI data register low (SCIDRL).<br>0 No noise<br>1 Noise |
| 1<br>FE | **Framing Error Flag** — FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading SCI status register 1 (SCISR1) with FE set and then reading the SCI data register low (SCIDRL).<br>0 No framing error<br>1 Framing error |
| 0<br>PF | **Parity Error Flag** — PF is set when the parity enable bit (PE) is set and the parity of the received data does not match the parity type bit (PT). PF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear PF by reading SCI status register 1 (SCISR1), and then reading SCI data register low (SCIDRL).<br>0 No parity error<br>1 Parity error |

### 12.3.2.8 SCI Status Register 2 (SCISR2)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-11. SCI Status Register 2 (SCISR2)**

Read: Anytime

Write: Anytime

**Table 12-12. SCISR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AMAP | **Alternative Map** — This bit controls which registers sharing the same address space are accessible. In the reset condition the SCI behaves as previous versions. Setting AMAP=1 allows the access to another set of control and status registers and hides the baud rate and SCI control Register 1.<br>0 The registers labelled SCIBDH (0x0000),SCIBDL (0x0001), SCICR1 (0x0002) are accessible<br>1 The registers labelled SCIASR1 (0x0000),SCIACR1 (0x0001), SCIACR2 (0x00002) are accessible |
| 4<br>TXPOL | **Transmit Polarity** — This bit control the polarity of the transmitted data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.<br>0 Normal polarity<br>1 Inverted polarity |
| 3<br>RXPOL | **Receive Polarity** — This bit control the polarity of the received data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.<br>0 Normal polarity<br>1 Inverted polarity |
| 2<br>BRK13 | **Break Transmit Character Length** — This bit determines whether the transmit break character is 10 or 11 bit respectively 13 or 14 bits long. The detection of a framing error is not affected by this bit.<br>0 Break character is 10 or 11 bit long<br>1 Break character is 13 or 14 bit long |
| 1<br>TXDIR | **Transmitter Pin Data Direction in Single-Wire Mode** — This bit determines whether the TXD pin is going to be used as an input or output, in the single-wire mode of operation. This bit is only relevant in the single-wire mode of operation.<br>0 TXD pin to be used as an input in single-wire mode<br>1 TXD pin to be used as an output in single-wire mode |
| 0<br>RAF | **Receiver Active Flag** — RAF is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character.<br>0 No reception in progress<br>1 Reception in progress |

### 12.3.2.9 SCI Data Registers (SCIDRH, SCIDRL)

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R8 | T8 | 0 | 0 | 0 | Reserved | Reserved | Reserved |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-12. SCI Data Registers (SCIDRH)**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-13. SCI Data Registers (SCIDRL)**

Read: Anytime; reading accesses SCI receive data register

Write: Anytime; writing accesses SCI transmit data register; writing to R8 has no effect

**Table 12-13. SCIDRH and SCIDRL Field Descriptions**

| Field | Description |
|---|---|
| SCIDRH<br>7<br>R8 | **Received Bit 8** — R8 is the ninth data bit received when the SCI is configured for 9-bit data format (M = 1). |
| SCIDRH<br>6<br>T8 | **Transmit Bit 8** — T8 is the ninth data bit transmitted when the SCI is configured for 9-bit data format (M = 1). |
| SCIDRL<br>7:0<br>R[7:0]<br>T[7:0] | **R7:R0** — Received bits seven through zero for 9-bit or 8-bit data formats<br>**T7:T0** — Transmit bits seven through zero for 9-bit or 8-bit formats |

**NOTE**

If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten.The same value is transmitted until T8 is rewritten

In 8-bit data format, only SCI data register low (SCIDRL) needs to be accessed.

When transmitting in 9-bit data format and using 8-bit write instructions, write first to SCI data register high (SCIDRH), then SCIDRL.

## 12.4 Functional Description

This section provides a complete functional description of the SCI block, detailing the operation of the design from the end user perspective in a number of subsections.

Figure 12-14 shows the structure of the SCI module. The SCI allows full duplex, asynchronous, serial communication between the CPU and remote devices, including other CPUs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

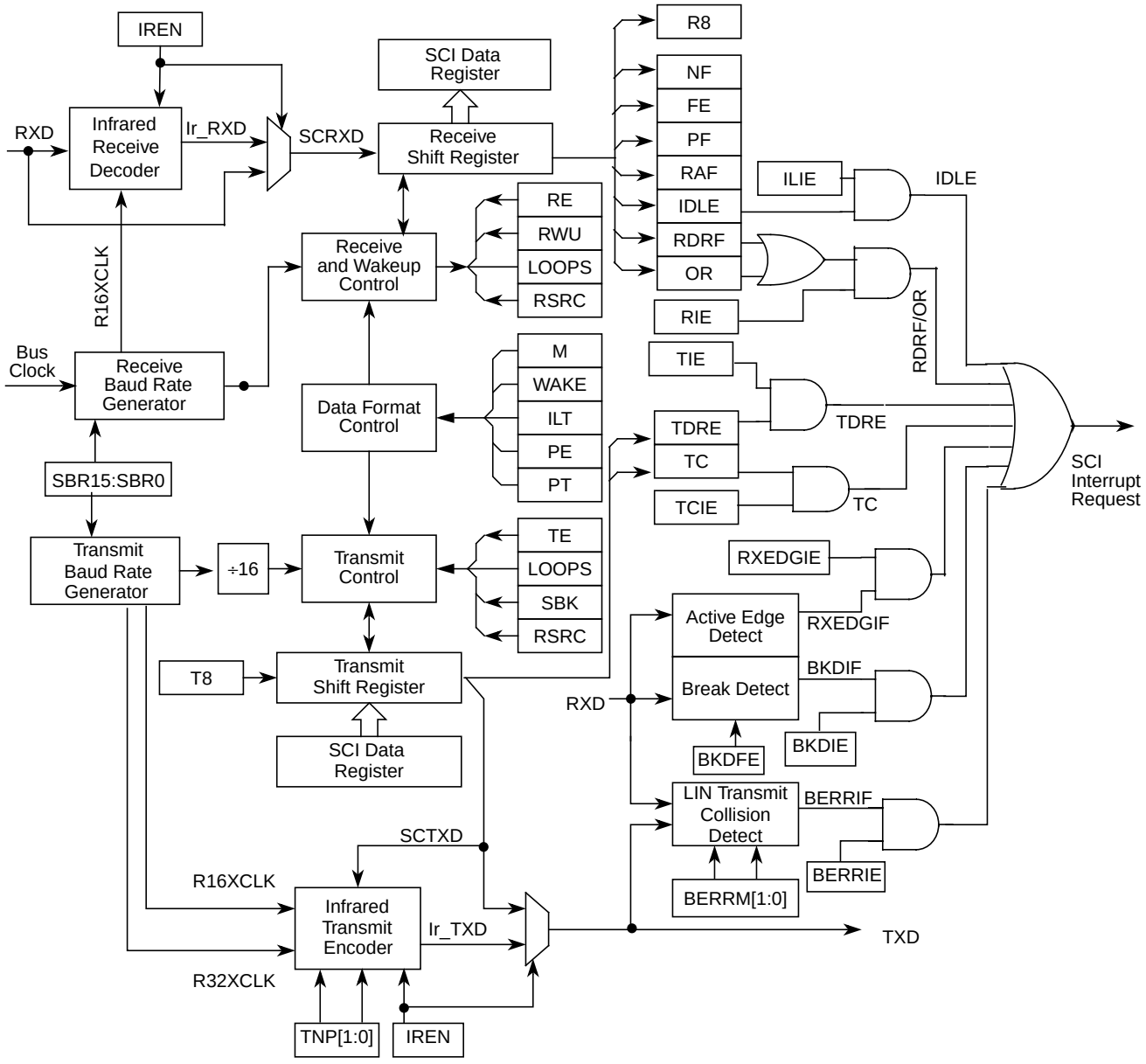



**Figure 12-14. Detailed SCI Block Diagram**

### 12.4.1 Infrared Interface Submodule

This module provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the SCI. The IrDA physical layer specification defines a half-duplex infrared communication link for exchange data. The full standard includes data rates up to 16 Mbits/s. This design covers only data rates between 2.4 Kbits/s and 115.2 Kbits/s.

The infrared submodule consists of two major blocks: the transmit encoder and the receive decoder. The SCI transmits serial bits of data which are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses should be detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder (external from the MCU). The narrow pulses are then stretched by the infrared submodule to get back to a serial bit stream to be received by the SCI.The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active low pulses.

The infrared submodule receives its clock sources from the SCI. One of these two clocks are selected in the infrared submodule in order to generate either 3/16, 1/16, 1/32 or 1/4 narrow pulses during transmission. The infrared block receives two clock sources from the SCI, R16XCLK and R32XCLK, which are configured to generate the narrow pulse width during transmission. The R16XCLK and R32XCLK are internal clocks with frequencies 16 and 32 times the baud rate respectively. Both R16XCLK and R32XCLK clocks are used for transmitting data. The receive decoder uses only the R16XCLK clock.

#### 12.4.1.1 Infrared Transmit Encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD pin. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent in the middle of the bit with a duration of 1/32, 1/16, 3/16 or 1/4 of a bit time. A narrow high pulse is transmitted for a zero bit when TXPOL is cleared, while a narrow low pulse is transmitted for a zero bit when TXPOL is set.

#### 12.4.1.2 Infrared Receive Decoder

The infrared receive block converts data from the RXD pin to the receive shift register. A narrow pulse is expected for each zero received and no pulse is expected for each one received. A narrow high pulse is expected for a zero bit when RXPOL is cleared, while a narrow low pulse is expected for a zero bit when RXPOL is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

### 12.4.2 LIN Support

This module provides some basic support for the LIN protocol. At first this is a break detect circuitry making it easier for the LIN software to distinguish a break character from an incoming data stream. As a further addition is supports a collision detection at the bit level as well as cancelling pending transmissions.

### 12.4.3 Data Format

The SCI uses the standard NRZ mark/space data format. When Infrared is enabled, the SCI uses RZI data format where zeroes are represented by light pulses and ones remain low. See Figure 12-15 below.

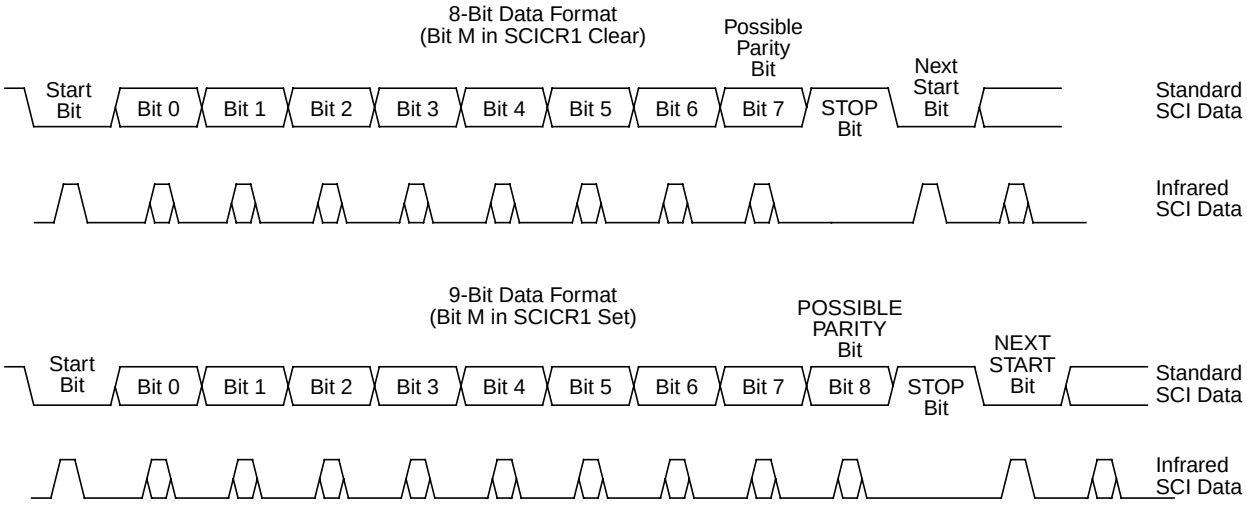



**Figure 12-15. SCI Data Formats**

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI control register 1 configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits. Setting the M bit configures the SCI for nine-bit data characters. A frame with nine data bits has a total of 11 bits.

**Table 12-14. Example of 8-Bit Data Formats**

| Start Bit | Data Bits | Address Bits | Parity Bits | Stop Bit |
|---|---|---|---|---|
| 1 | 8 | 0 | 0 | 1 |
| 1 | 7 | 0 | 1 | 1 |
| 1 | 7 | 1[1] | 0 | 1 |

1 The address bit identifies the frame as an address character. See Section 12.4.6.6, “Receiver Wakeup”.

When the SCI is configured for 9-bit data characters, the ninth data bit is the T8 bit in SCI data register high (SCIDRH). It remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits.

**Table 12-15. Example of 9-Bit Data Formats**

| Start Bit | Data Bits | Address Bits | Parity Bits | Stop Bit |
|---|---|---|---|---|
| 1 | 9 | 0 | 0 | 1 |
| 1 | 8 | 0 | 1 | 1 |
| 1 | 8 | 1[1] | 0 | 1 |

1 The address bit identifies the frame as an address character. See Section 12.4.6.6, “Receiver Wakeup”.

## 12.4.4 Baud Rate Generation

A 16-bit modulus counter in the two baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 0 to 65535 written to the SBR15:SBR0 bits determines the baud rate. The value from 0 to 4095 written to the SBR15:SBR4 bits determines the baud rate clock with SBR3:SBR0 for fine adjust. The SBR bits are in the SCI baud rate registers (SCIBDH and SCIBDL) for both transmit and receive baud generator. The baud rate clock is synchronized with the bus clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time.

Baud rate generation is subject to one source of error:

- Integer division of the bus clock may not give the exact target frequency.

Table 12-16 lists some examples of achieving target baud rates with a bus clock frequency of 25 MHz.

When IREN = 0 then,

SCI baud rate = SCI bus clock / (SCIBR[15:0])

**Table 12-16. Baud Rates (Example: Bus Clock = 25 MHz)**

| Bits SBR[15:0] | Receiver[1] Clock (Hz) | Transmitter[2] Clock (Hz) | Target Baud Rate | Error (%) |
|---|---|---|---|---|
| 109 | 3669724.8 | 229,357.8 | 230,400 | .452 |
| 217 | 1843318.0 | 115,207.4 | 115,200 | .006 |
| 651 | 614439.3 | 38,402.5 | 38,400 | .006 |
| 1302 | 307219.7 | 19,201.2 | 19,200 | .006 |
| 2604 | 153,609.8 | 9600.6 | 9,600 | .006 |
| 5208 | 76,804.9 | 4800.3 | 4,800 | .006 |
| 10417 | 38,398.8 | 2399.9 | 2,400 | .003 |
| 20833 | 19,200.3 | 1200.02 | 1,200 | .00 |
| 41667 | 9599.9 | 600.0 | 600 | .00 |
| 65535 | 6103.6 | 381.5 | | |

1 16x faster then baud rate

2 divide 1/16 form transmit baud generator

## 12.4.5 Transmitter

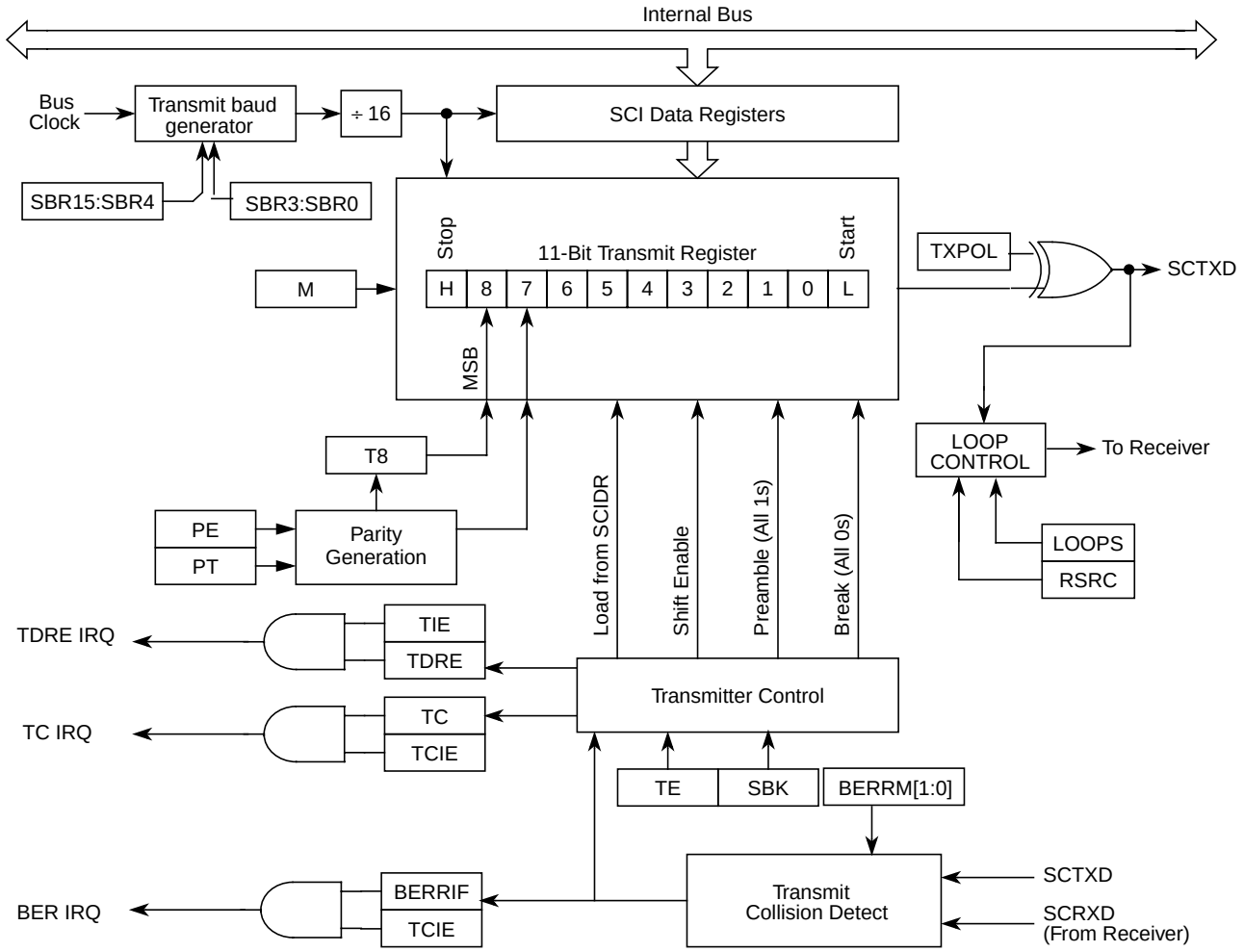



**Figure 12-16. Transmitter Block Diagram**

### 12.4.5.1 Transmitter Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

### 12.4.5.2 Character Transmission

To transmit data, the MCU writes the data bits to the SCI data registers (SCIDRH/SCIDRL), which in turn are transferred to the transmitter shift register. The transmit shift register then shifts a frame out through the TXD pin, after it has prefaced them with a start bit and appended them with a stop bit. The SCI data registers (SCIDRH and SCIDRL) are the write-only buffers between the internal data bus and the transmit shift register.

The SCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDRH/L) to the transmitter shift register.The transmit driver routine may respond to this flag by writing another byte to the Transmitter buffer (SCIDRH/SCIDRL), while the shift register is still shifting out the first byte.

To initiate an SCI transmission:

1. Configure the SCI:
   a) Select a baud rate. Write this value to the SCI baud registers (SCIBDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the SCIBDH has no effect without also writing to SCIBDL.
   b) Write to SCICR1 to configure word length, parity, and other configuration bits (LOOPS,RSRC,M,WAKE,ILT,PE,PT).
   c) Enable the transmitter, interrupts, receive, and wake up as required, by writing to the SCICR2 register bits (TIE,TCIE,RIE,ILIE,TE,RE,RWU,SBK). A preamble or idle character will now be shifted out of the transmitter shift register.
2. Transmit Procedure for each byte:
   a) Poll the TDRE flag by reading the SCISR1 or responding to the TDRE interrupt. Keep in mind that the TDRE bit resets to one.
   b) If the TDRE flag is set, write the data to be transmitted to SCIDRH/L, where the ninth bit is written to the T8 bit in SCIDRH if the SCI is in 9-bit data format. A new transmission will not result until the TDRE flag has been cleared.
3. Repeat step 2 for each subsequent transmission.

**NOTE**

The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDRH/L, which happens, generally speaking, a little over half-way through the stop bit of the previous frame. Specifically, this transfer occurs 9/16ths of a bit time AFTER the start of the stop bit of the previous frame.

Writing the TE bit from 0 to a 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCI data register into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (MSB) of the data character is the parity bit.

The transmit data register empty flag, TDRE, in SCI status register 1 (SCISR1) becomes set when the SCI data register transfers a byte to the transmit shift register. The TDRE flag indicates that the SCI data register can accept new data from the internal data bus. If the transmit interrupt enable bit, TIE, in SCI control register 2 (SCICR2) is also set, the TDRE flag generates a transmitter interrupt request.

When the transmit shift register is not transmitting a frame, the TXD pin goes to the idle condition, logic 1. If at any time software clears the TE bit in SCI control register 2 (SCICR2), the transmitter enable signal goes low and the transmit signal goes idle.

If software clears TE while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last byte of the first message to SCIDRH/L.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to SCIDRH/L.

#### 12.4.5.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCI control register 2 (SCICR2) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCI control register 1 (SCICR1). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

The SCI recognizes a break character when there are 10 or 11(M = 0 or M = 1) consecutive zero received. Depending if the break detect feature is enabled or not receiving a break character has these effects on SCI registers.

If the break detect feature is disabled (BKDFE = 0):

- Sets the framing error flag, FE
- Sets the receive data register full flag, RDRF
- Clears the SCI data registers (SCIDRH/L)
- May set the overrun flag, OR, noise flag, NF, parity error flag, PE, or the receiver active flag, RAF (see 3.4.4 and 3.4.5 SCI Status Register 1 and 2)

If the break detect feature is enabled (BKDFE = 1) there are two scenarios[1]

The break is detected right from a start bit or is detected during a byte reception.

- Sets the break detect interrupt flag, BKDIF
- Does not change the data register full flag, RDRF or overrun flag OR
- Does not change the framing error flag FE, parity error flag PE.
- Does not clear the SCI data registers (SCIDRH/L)
- May set noise flag NF, or receiver active flag RAF.

1. A Break character in this context are either 10 or 11 consecutive zero received bits

Figure 12-17 shows two cases of break detect. In trace RXD_1 the break symbol starts with the start bit, while in RXD_2 the break starts in the middle of a transmission. If BRKDFE = 1, in RXD_1 case there will be no byte transferred to the receive buffer and the RDRF flag will not be modified. Also no framing error or parity error will be flagged from this transfer. In RXD_2 case, however the break signal starts later during the transmission. At the expected stop bit position the byte received so far will be transferred to the receive buffer, the receive data register full flag will be set, a framing error and if enabled and appropriate a parity error will be set. Once the break is detected the BRKDIF flag will be set.

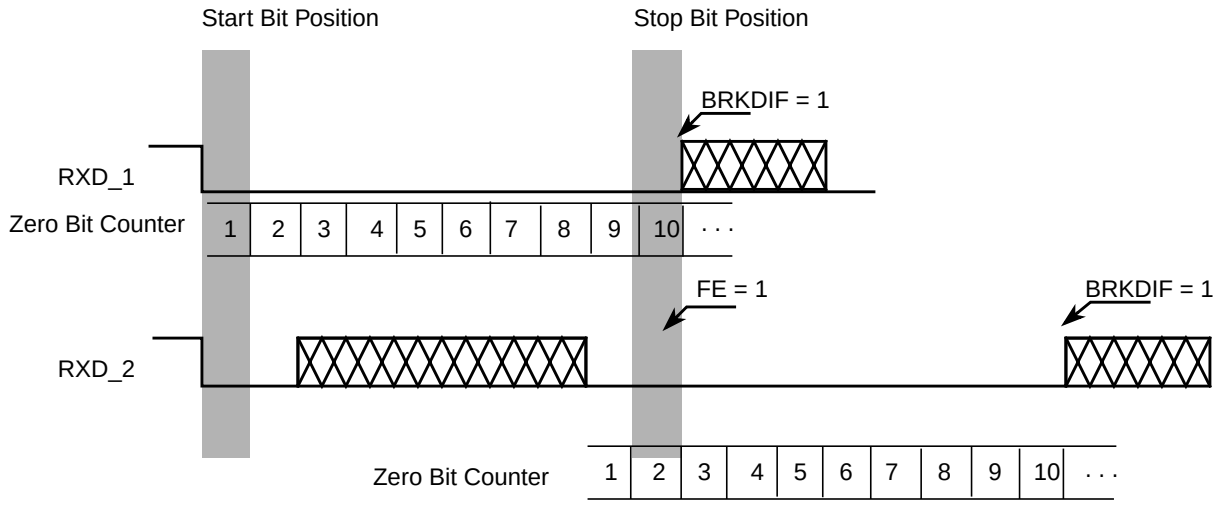



**Figure 12-17. Break Detection if BRKDFE = 1 (M = 0)**

### 12.4.5.4 Idle Characters

An idle character (or preamble) contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCI control register 1 (SCICR1). The preamble is a synchronizing idle character that begins the first transmission initiated after writing the TE bit from 0 to 1.

If the TE bit is cleared during a transmission, the TXD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

**NOTE**

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current frame shifts out through the TXD pin. Setting TE after the stop bit appears on TXD causes data previously written to the SCI data register to be lost. Toggle the TE bit for a queued idle character while the TDRE flag is set and immediately before writing the next byte to the SCI data register.

If the TE bit is clear and the transmission is complete, the SCI is not the master of the TXD pin

### 12.4.5.5 LIN Transmit Collision Detection

This module allows to check for collisions on the LIN bus.

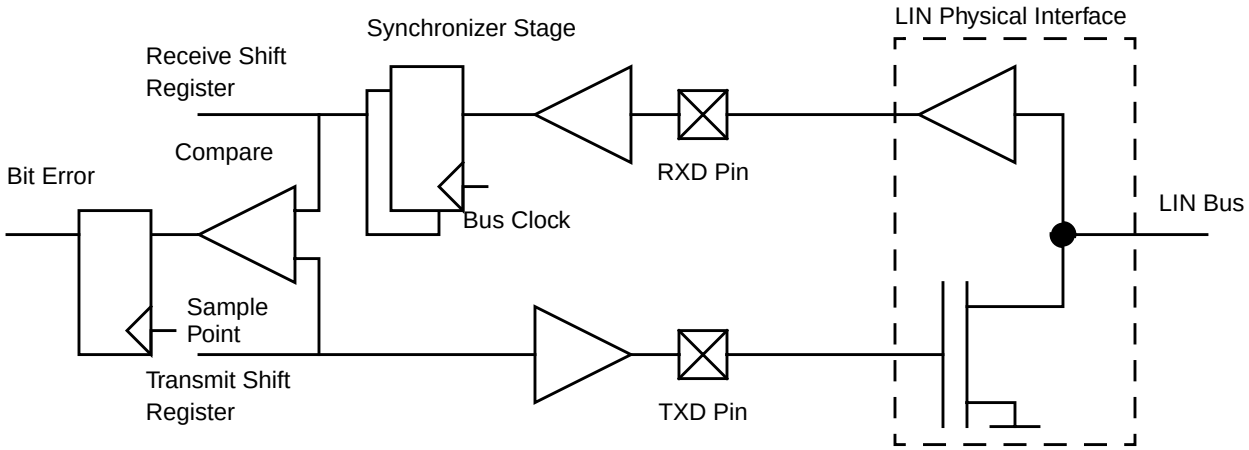



**Figure 12-18. Collision Detect Principle**

If the bit error circuit is enabled (BERRM[1:0] = 0:1 or = 1:0]), the error detect circuit will compare the transmitted and the received data stream at a point in time and flag any mismatch. The timing checks run when transmitter is active (not idle). As soon as a mismatch between the transmitted data and the received data is detected the following happens:

- The next bit transmitted will have a high level (TXPOL = 0) or low level (TXPOL = 1)
- The transmission is aborted and the byte in transmit buffer is discarded.
- the transmit data register empty and the transmission complete flag will be set
- The bit error interrupt flag, BERRIF, will be set.
- No further transmissions will take place until the BERRIF is cleared.

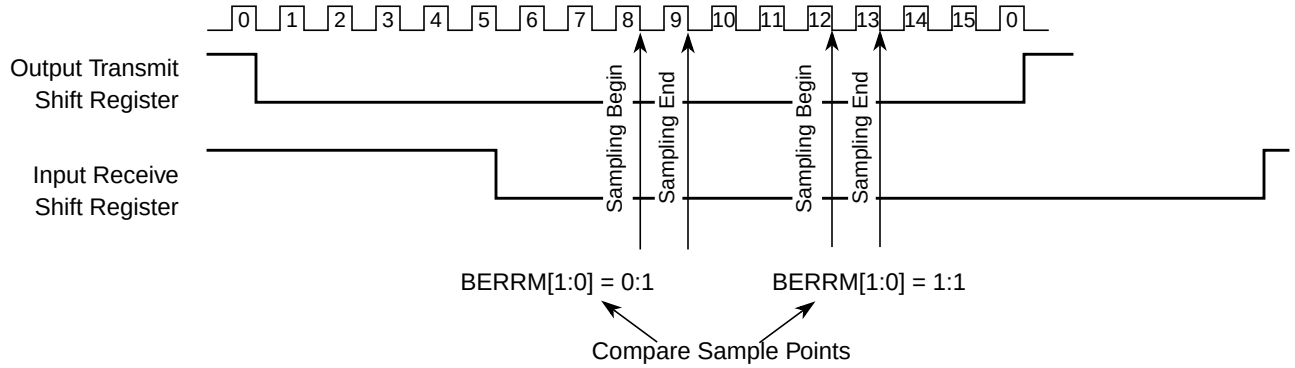



**Figure 12-19. Timing Diagram Bit Error Detection**

If the bit error detect feature is disabled, the bit error interrupt flag is cleared.

**NOTE**

The RXPOL and TXPOL bit should be set the same when transmission collision detect feature is enabled, otherwise the bit error interrupt flag may be set incorrectly.

## 12.4.6 Receiver

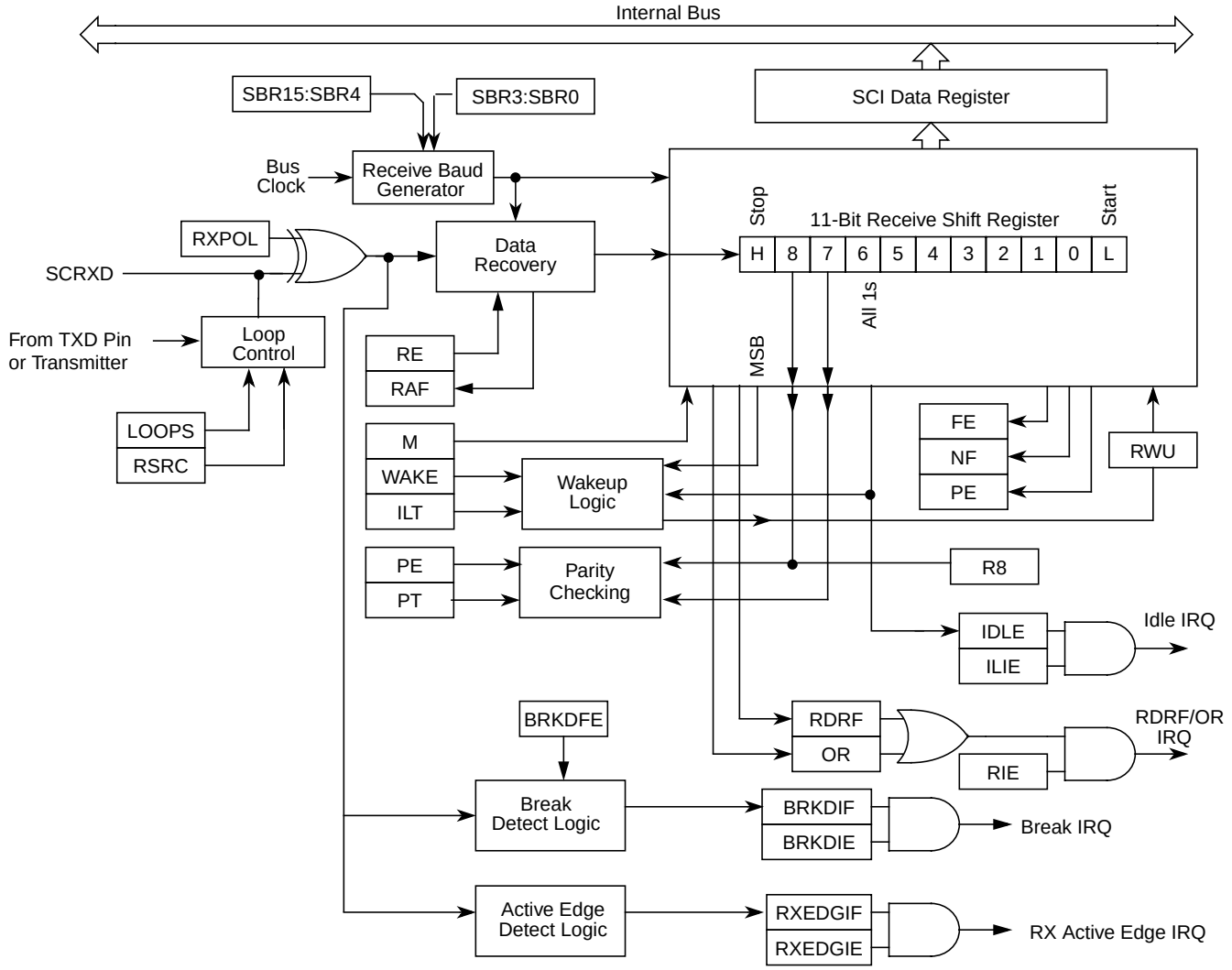



**Figure 12-20. SCI Receiver Block Diagram**

### 12.4.6.1 Receiver Character Length

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When receiving 9-bit data, bit R8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

### 12.4.6.2 Character Reception

During an SCI reception, the receive shift register shifts a frame in from the RXD pin. The SCI data register is the read-only buffer between the internal data bus and the receive shift register.

After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the SCI data register. The receive data register full flag, RDRF, in SCI status register 1 (SCISR1) becomes set,

indicating that the received byte can be read. If the receive interrupt enable bit, RIE, in SCI control register 2 (SCICR2) is also set, the RDRF flag generates an RDRF interrupt request.

### 12.4.6.3 Data Sampling

The RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see Figure 12-21) is re-synchronized immediatelly at bus clock edge:

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s.When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

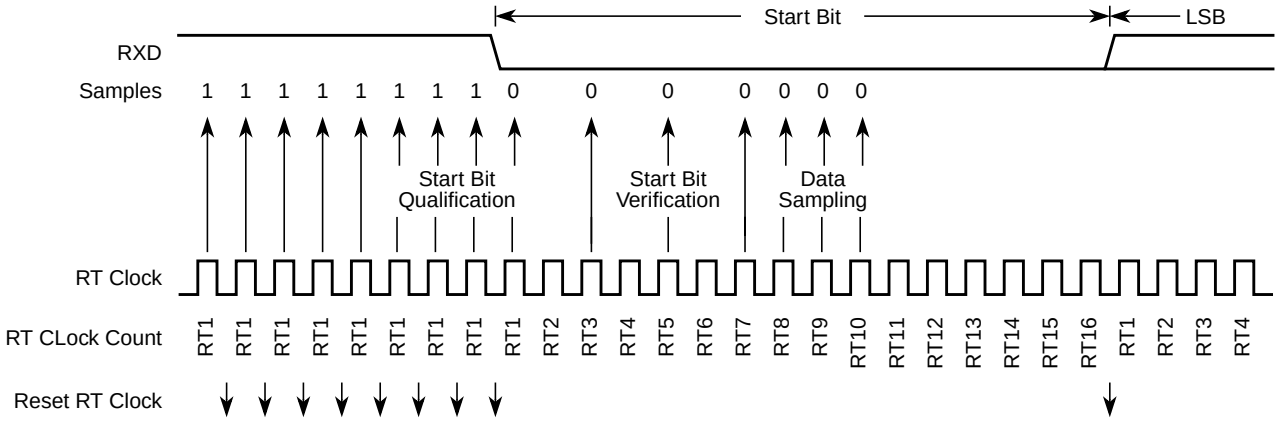



**Figure 12-21. Receiver Data Sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Figure 12-17 summarizes the results of the start bit verification samples.

**Table 12-17. Start Bit Verification**

| RT3, RT5, and RT7 Samples | Start Bit Verification | Noise Flag |
|---|---|---|
| 000 | Yes | 0 |
| 001 | Yes | 1 |
| 010 | Yes | 1 |
| 011 | No | 0 |
| 100 | Yes | 1 |
| 101 | No | 0 |
| 110 | No | 0 |
| 111 | No | 0 |

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 12-18 summarizes the results of the data bit samples.

**Table 12-18. Data Bit Recovery**

| RT8, RT9, and RT10 Samples | Data Bit Determination | Noise Flag |
|---|---|---|
| 000 | 0 | 0 |
| 001 | 0 | 1 |
| 010 | 0 | 1 |
| 011 | 1 | 1 |
| 100 | 0 | 1 |
| 101 | 1 | 1 |
| 110 | 1 | 1 |
| 111 | 1 | 0 |

**NOTE**

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 12-19 summarizes the results of the stop bit samples.

**Table 12-19. Stop Bit Recovery**

| RT8, RT9, and RT10 Samples | Framing Error Flag | Noise Flag |
|---|---|---|
| 000 | 1 | 0 |
| 001 | 1 | 1 |
| 010 | 1 | 1 |
| 011 | 0 | 1 |
| 100 | 1 | 1 |
| 101 | 0 | 1 |
| 110 | 0 | 1 |
| 111 | 0 | 0 |

In Figure 12-22 the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

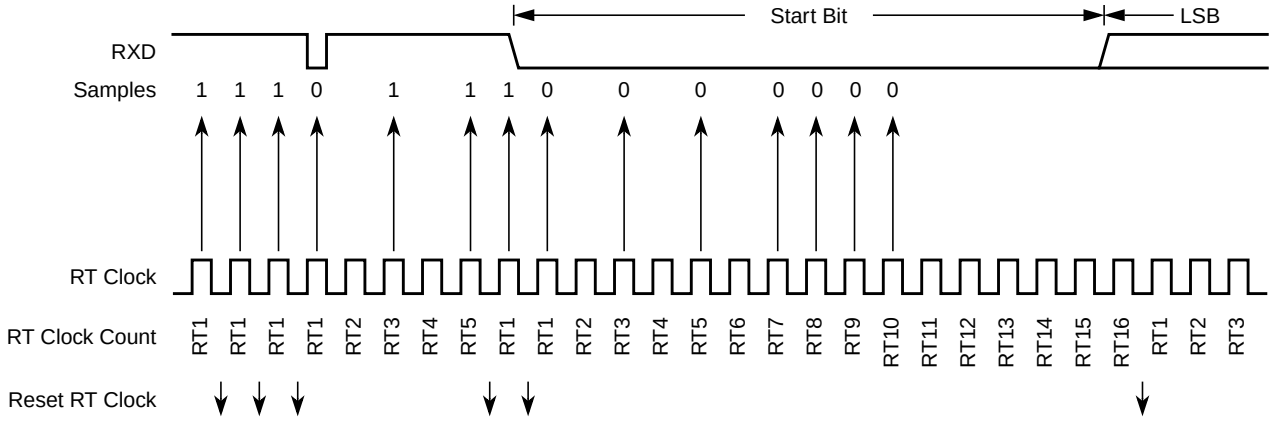



**Figure 12-22. Start Bit Search Example 1**

In Figure 12-23, verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

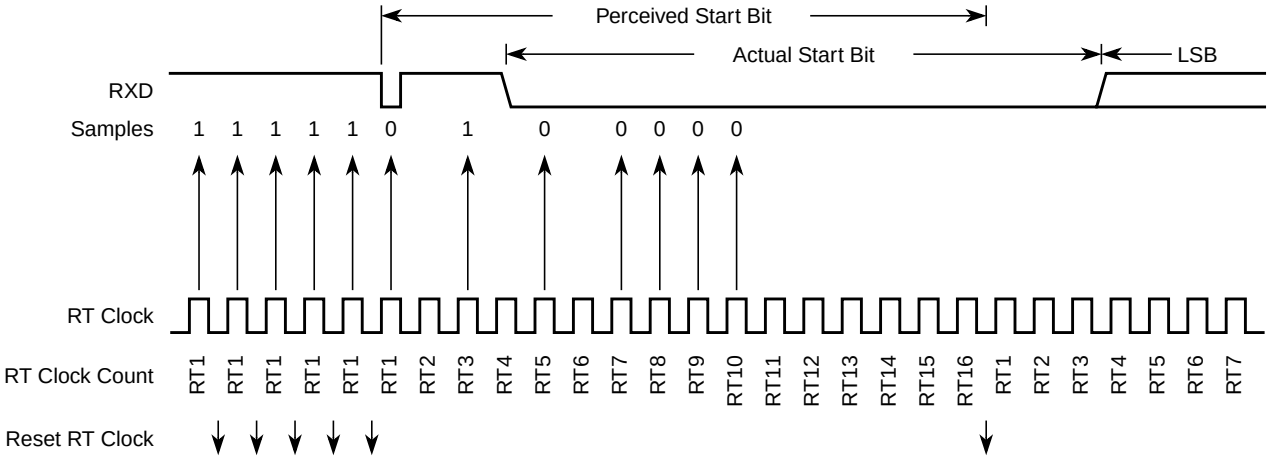



**Figure 12-23. Start Bit Search Example 2**

In Figure 12-24, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

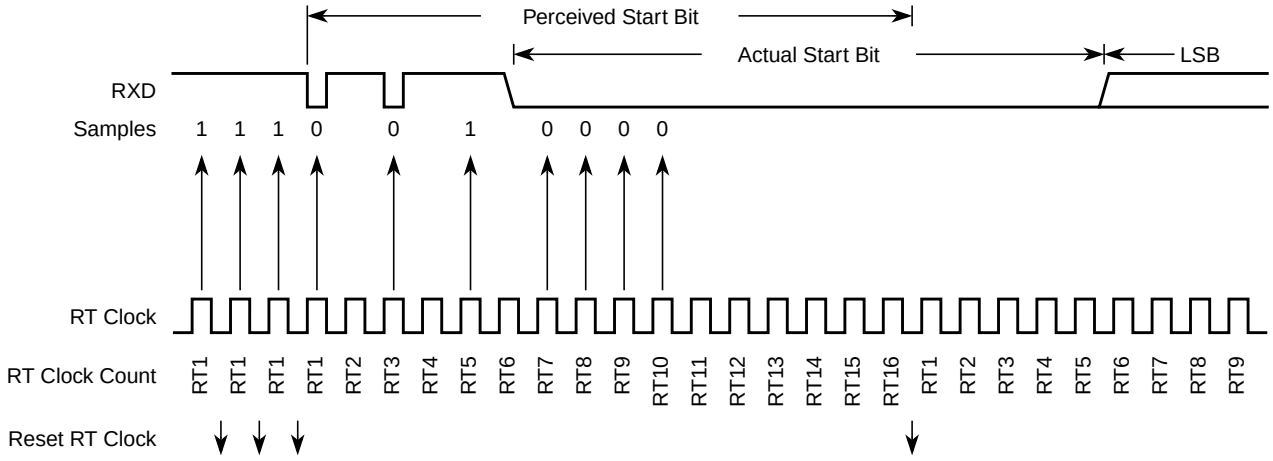



**Figure 12-24. Start Bit Search Example 3**

Figure 12-25 shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

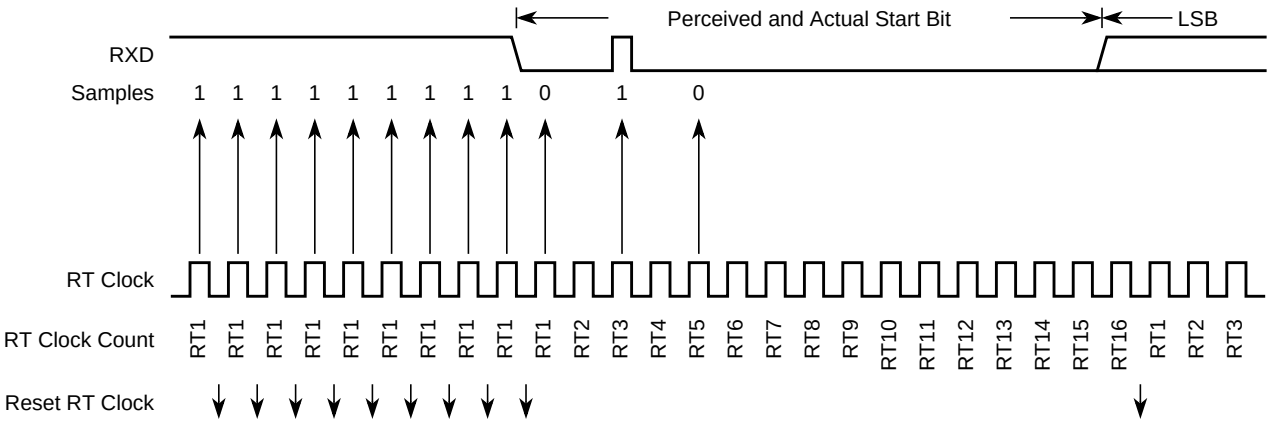



**Figure 12-25. Start Bit Search Example 4**

Figure 12-26 shows a burst of noise near the beginning of the start bit that resets the RT clock. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

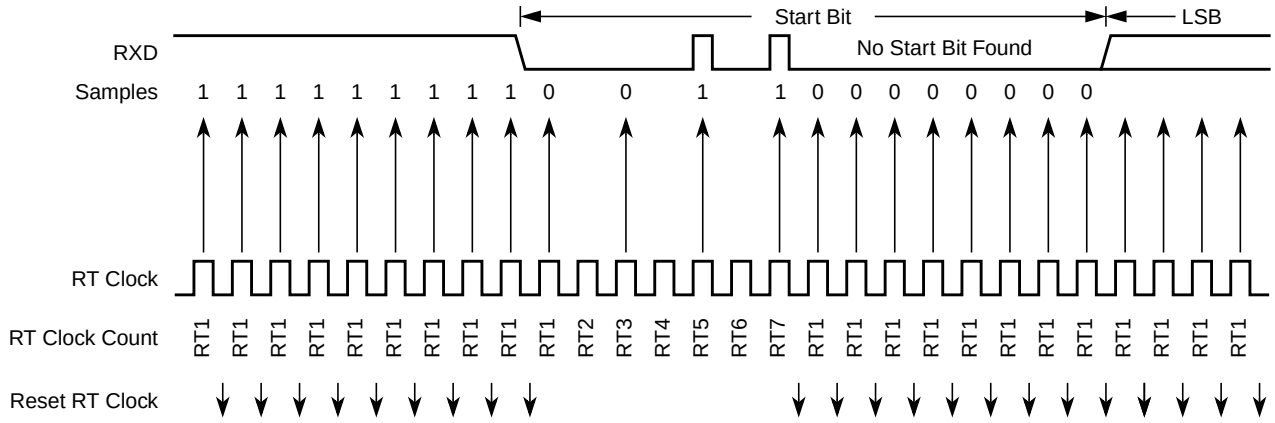



**Figure 12-26. Start Bit Search Example 5**

In Figure 12-27, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored.

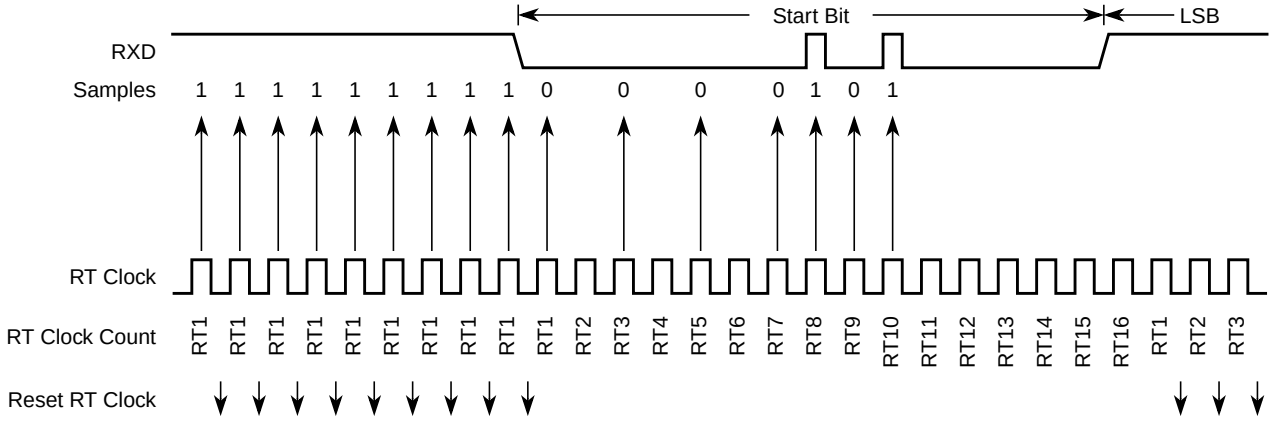



**Figure 12-27. Start Bit Search Example 6**

### 12.4.6.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, FE, in SCI status register 1 (SCISR1). A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set.

### 12.4.6.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero.

As the receiver samples an incoming frame, it re-synchronizes the RT clock on any valid falling edge within the frame. Re synchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

#### 12.4.6.5.1 Slow Data Tolerance

Figure 12-28 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

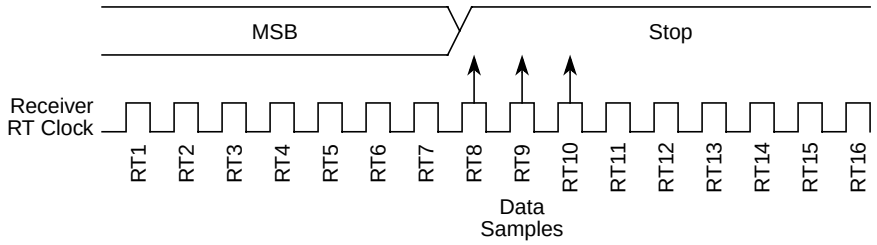



**Figure 12-28. Slow Data**

Let’s take RTr as receiver RT clock and RTt as transmitter RT clock.

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles +7 RTr cycles = 151 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 12-28, the receiver counts 151 RTr cycles at the point when the count of the transmitting device is 9 bit times x 16 RTt cycles = 144 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

((151 – 144) / 151) x 100 = 4.63%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 7 RTr cycles = 167 RTr cycles to start data sampling of the stop bit.

With the misaligned character shown in Figure 12-28, the receiver counts 167 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

((167 – 160) / 167) X 100 = 4.19%

#### 12.4.6.5.2 Fast Data Tolerance

Figure 12-29 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

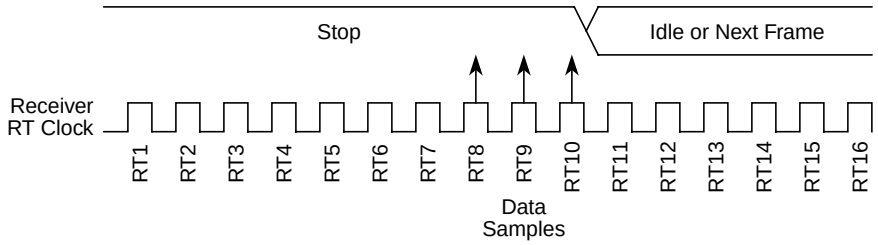



**Figure 12-29. Fast Data**

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles + 9 RTr cycles = 153 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 12-29, the receiver counts 153 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

((160 – 153) / 160) x 100 = 4.375%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 9 RTr cycles = 169 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 12-29, the receiver counts 169 RTr cycles at the point when the count of the transmitting device is 11 bit times x 16 RTt cycles = 176 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

((176 – 169) /176) x 100 = 3.98%

**NOTE**

Due to asynchronous sample and internal logic, there is maximal 2 bus cycles between startbit edge and 1st RT clock, and cause to additional tolerance loss at worst case. The loss should be 2/SBR/10*100%, it is small.For example, for highspeed baud=230400 with 25MHz bus, SBR should be 109, and the tolerance loss is 2/109/10*100=0.18%, and fast data tolerance is 4.375%-0.18%=4.195%.

### 12.4.6.6 Receiver Wakeup

To enable the SCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCI control register 2 (SCICR2) puts the receiver into standby state during which receiver interrupts are disabled.The SCI will still load the receive data into the SCIDRH/L registers, but it will not set the RDRF flag.

The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message.

The WAKE bit in SCI control register 1 (SCICR1) determines how the SCI is brought out of the standby state to process an incoming message. The WAKE bit enables either idle line wakeup or address mark wakeup.

#### 12.4.6.6.1 Idle Input line Wakeup (WAKE = 0)

In this wakeup method, an idle condition on the RXD pin clears the RWU bit and wakes up the SCI. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on the RXD pin.

Idle line wakeup requires that messages be separated by at least one idle character and that no message contains idle characters.

The idle character that wakes a receiver does not set the receiver idle bit, IDLE, or the receive data register full flag, RDRF.

The idle line type bit, ILT, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit. ILT is in SCI control register 1 (SCICR1).

#### 12.4.6.6.2 Address Mark Wakeup (WAKE = 1)

In this wakeup method, a logic 1 in the most significant bit (MSB) position of a frame clears the RWU bit and wakes up the SCI. The logic 1 in the MSB position marks a frame as an address frame that contains addressing information. All receivers evaluate the addressing information, and the receivers for which the message is addressed process the frames that follow.Any receiver for which a message is not addressed can set its RWU bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another address frame appears on the RXD pin.

The logic 1 MSB of an address frame clears the receiver's RWU bit before the stop bit is received and sets the RDRF flag.

Address mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames.

**NOTE**

With the WAKE bit clear, setting the RWU bit after the RXD pin has been idle can cause the receiver to wake up immediately.

### 12.4.7 Single-Wire Operation

Normally, the SCI uses two pins for transmitting and receiving. In single-wire operation, the RXD pin is disconnected from the SCI. The SCI uses the TXD pin for both receiving and transmitting.

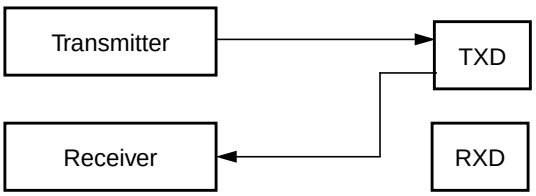



**Figure 12-30. Single-Wire Operation (LOOPS = 1, RSRC = 1)**

Enable single-wire operation by setting the LOOPS bit and the receiver source bit, RSRC, in SCI control register 1 (SCICR1). Setting the LOOPS bit disables the path from the RXD pin to the receiver. Setting the RSRC bit connects the TXD pin to the receiver. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).The TXDIR bit (SCISR2[1]) determines whether the TXD pin is going to be used as an input (TXDIR = 0) or an output (TXDIR = 1) in this mode of operation.

**NOTE**

In single-wire operation data from the TXD pin is inverted if RXPOL is set.

### 12.4.8 Loop Operation

In loop operation the transmitter output goes to the receiver input. The RXD pin is disconnected from the SCI.

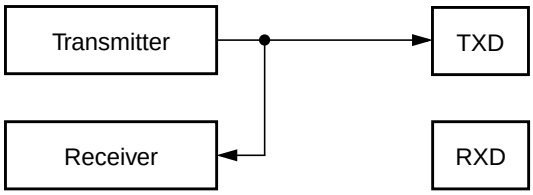



**Figure 12-31. Loop Operation (LOOPS = 1, RSRC = 0)**

Enable loop operation by setting the LOOPS bit and clearing the RSRC bit in SCI control register 1 (SCICR1). Setting the LOOPS bit disables the path from the RXD pin to the receiver. Clearing the RSRC bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (TE = 1 and RE = 1).

**NOTE**

In loop operation data from the transmitter is not recognized by the receiver if RXPOL and TXPOL are not the same.

## 12.5 Initialization/Application Information

### 12.5.1 Reset Initialization

See Section 12.3.2, “Register Descriptions”.

## 12.5.2 Modes of Operation

### 12.5.2.1 Run Mode

Normal mode of operation.

To initialize a SCI transmission, see Section 12.4.5.2, “Character Transmission”.

### 12.5.2.2 Wait Mode

SCI operation in wait mode depends on the state of the SCISWAI bit in the SCI control register 1 (SCICR1).

- If SCISWAI is clear, the SCI operates normally when the CPU is in wait mode.
- If SCISWAI is set, SCI clock generation ceases and the SCI module enters a power-conservation state when the CPU is in wait mode. Setting SCISWAI does not affect the state of the receiver enable bit, RE, or the transmitter enable bit, TE.

  If SCISWAI is set, any transmission or reception in progress stops at wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of wait mode. Exiting wait mode by reset aborts any transmission or reception in progress and resets the SCI.

### 12.5.2.3 Stop Mode

The SCI is inactive during stop mode for reduced power consumption. The STOP instruction does not affect the SCI register states, but the SCI bus clock will be disabled. The SCI operation resumes from where it left off after an external interrupt brings the CPU out of stop mode. Exiting stop mode by reset aborts any transmission or reception in progress and resets the SCI.

The receive input active edge detect circuit is still active in stop mode. An active edge on the receive input can be used to bring the CPU out of stop mode.

## 12.5.3 Interrupt Operation

This section describes the interrupt originated by the SCI block.The MCU must service the interrupt requests. Table 12-20 lists the eight interrupt sources of the SCI.

**Table 12-20. SCI Interrupt Sources**

| Interrupt | Source | Local Enable | Description |
|---|---|---|---|
| TDRE | SCISR1[7] | TIE | Active high level. Indicates that a byte was transferred from SCIDRH/L to the transmit shift register. |
| TC | SCISR1[6] | TCIE | Active high level. Indicates that a transmit is complete. |
| RDRF | SCISR1[5] | RIE | Active high level. The RDRF interrupt indicates that received data is available in the SCI data register. |
| OR | SCISR1[3] | | Active high level. This interrupt indicates that an overrun condition has occurred. |
| IDLE | SCISR1[4] | ILIE | Active high level. Indicates that receiver input has become idle. |

**Table 12-20. SCI Interrupt Sources**

| | | | |
|---|---|---|---|
| RXEDGIF | SCIASR1[7] | RXEDGIE | Active high level. Indicates that an active edge (falling for RXPOL = 0, rising for RXPOL = 1) was detected. |
| BERRIF | SCIASR1[1] | BERRIE | Active high level. Indicates that a mismatch between transmitted and received data in a single wire application has happened. |
| BKDIF | SCIASR1[0] | BRKDIE | Active high level. Indicates that a break character has been received. |

### 12.5.3.1 Description of Interrupt Operation

The SCI only originates interrupt requests. The following is a description of how the SCI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt number are chip dependent. The SCI only has a single interrupt line (SCI Interrupt Signal, active high operation) and all the following interrupts, when generated, are ORed together and issued through that port.

#### 12.5.3.1.1 TDRE Description

The TDRE interrupt is set high by the SCI when the transmit shift register receives a byte from the SCI data register. A TDRE interrupt indicates that the transmit data register (SCIDRH/L) is empty and that a new byte can be written to the SCIDRH/L for transmission.Clear TDRE by reading SCI status register 1 with TDRE set and then writing to SCI data register low (SCIDRL).

#### 12.5.3.1.2 TC Description

The TC interrupt is set by the SCI when a transmission has been completed. Transmission is completed when all bits including the stop bit (if transmitted) have been shifted out and no data is queued to be transmitted. No stop bit is transmitted when sending a break character and the TC flag is set (providing there is no more data queued for transmission) when the break character has been shifted out. A TC interrupt indicates that there is no transmission in progress. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, the TXD pin becomes idle (logic 1). Clear TC by reading SCI status register 1 (SCISR1) with TC set and then writing to SCI data register low (SCIDRL).TC is cleared automatically when data, preamble, or break is queued and ready to be sent.

#### 12.5.3.1.3 RDRF Description

The RDRF interrupt is set when the data in the receive shift register transfers to the SCI data register. A RDRF interrupt indicates that the received data has been transferred to the SCI data register and that the byte can now be read by the MCU. The RDRF interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

#### 12.5.3.1.4 OR Description

The OR interrupt is set when software fails to read the SCI data register before the receive shift register receives the next frame. The newly acquired data in the shift register will be lost in this case, but the data already in the SCI data registers is not affected. The OR interrupt is cleared by reading the SCI status register one (SCISR1) and then reading SCI data register low (SCIDRL).

#### 12.5.3.1.5 IDLE Description

The IDLE interrupt is set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag. Clear IDLE by reading SCI status register 1 (SCISR1) with IDLE set and then reading SCI data register low (SCIDRL).

#### 12.5.3.1.6 RXEDGIF Description

The RXEDGIF interrupt is set when an active edge (falling if RXPOL = 0, rising if RXPOL = 1) on the RXD pin is detected. Clear RXEDGIF by writing a “1” to the SCIASR1 SCI alternative status register 1.

#### 12.5.3.1.7 BERRIF Description

The BERRIF interrupt is set when a mismatch between the transmitted and the received data in a single wire application like LIN was detected. Clear BERRIF by writing a “1” to the SCIASR1 SCI alternative status register 1. This flag is also cleared if the bit error detect feature is disabled.

#### 12.5.3.1.8 BKDIF Description

The BKDIF interrupt is set when a break signal was received. Clear BKDIF by writing a “1” to the SCIASR1 SCI alternative status register 1. This flag is also cleared if break detect feature is disabled.

### 12.5.4 Recovery from Wait Mode

The SCI interrupt request can be used to bring the CPU out of wait mode.

### 12.5.5 Recovery from Stop Mode

An active edge on the receive input can be used to bring the CPU out of stop mode.

# Chapter 13
# Serial Peripheral Interface (S12SPIV5)

**Table 13-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V05.00 | 24 Mar 2005 | 13.3.2/13-513 | - Added 16-bit transfer width feature. |

## 13.1 Introduction

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

### 13.1.1 Glossary of Terms

| | |
|---|---|
| SPI | Serial Peripheral Interface |
| $\overline{SS}$ | Slave Select |
| SCK | Serial Clock |
| MOSI | Master Output, Slave Input |
| MISO | Master Input, Slave Output |
| MOMI | Master Output, Master Input |
| SISO | Slave Input, Slave Output |

### 13.1.2 Features

The SPI includes these distinctive features:

- Master mode and slave mode
- Selectable 8 or 16-bit transfer width
- Bidirectional mode
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Double-buffered data register
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode

### 13.1.3 Modes of Operation

The SPI functions in three modes: run, wait, and stop.

- Run mode

  This is the basic mode of operation.
- Wait mode

  SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPICR2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in run mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into run mode. If the SPI is configured as a slave, reception and transmission of data continues, so that the slave stays synchronized to the master.
- Stop mode

  The SPI is inactive in stop mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into run mode. If the SPI is configured as a slave, reception and transmission of data continues, so that the slave stays synchronized to the master.

For a detailed description of operating modes, please refer to Section 13.4.7, “Low Power Mode Options”.

### 13.1.4 Block Diagram

Figure 13-1 gives an overview on the SPI architecture. The main parts of the SPI are status, control and data registers, shifter logic, baud rate generator, master/slave control logic, and port control logic.

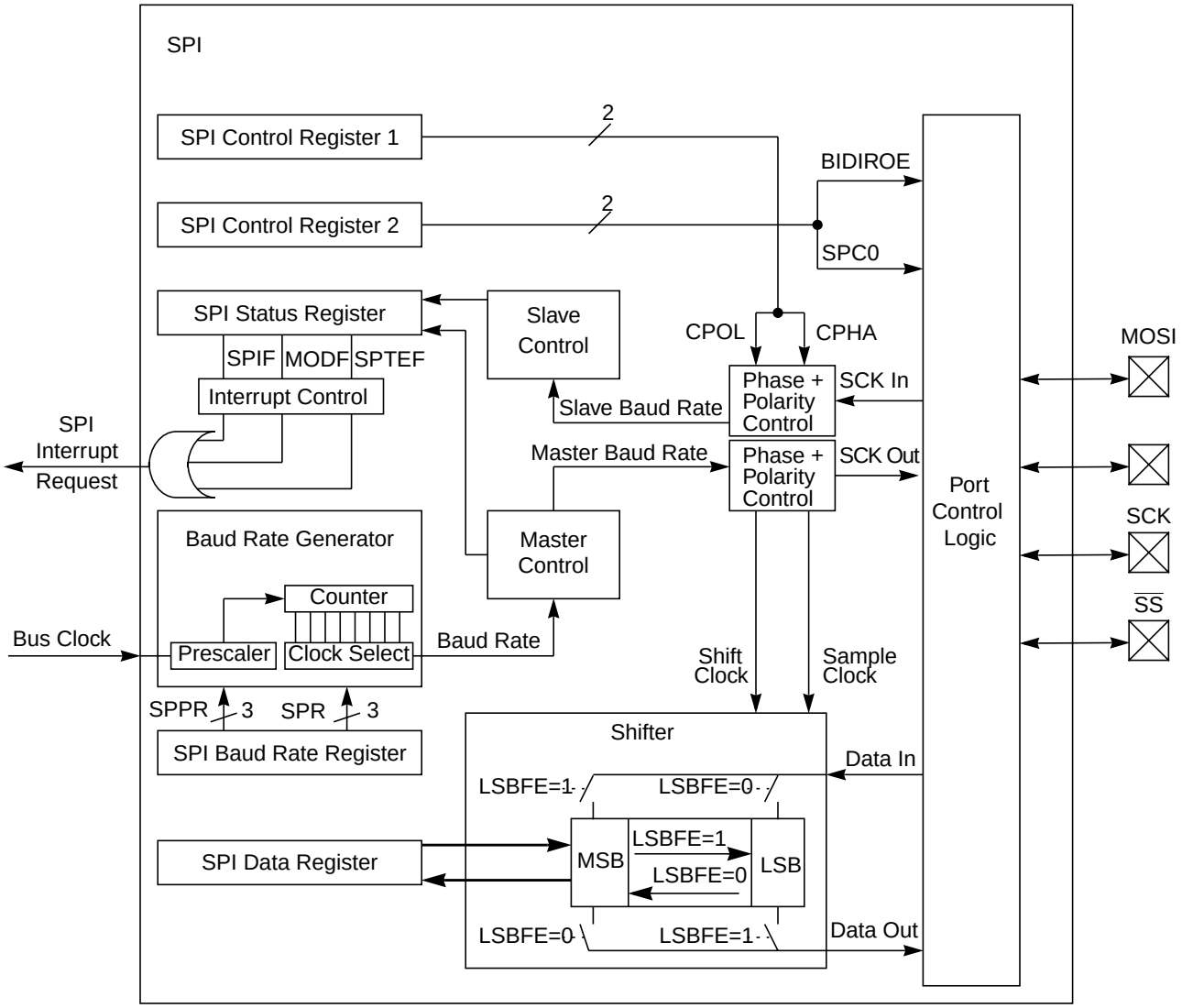



**Figure 13-1. SPI Block Diagram**

## 13.2 External Signal Description

This section lists the name and description of all ports including inputs and outputs that do, or may, connect off chip. The SPI module has a total of four external pins.

### 13.2.1 MOSI — Master Out/Slave In Pin

This pin is used to transmit data out of the SPI module when it is configured as a master and receive data when it is configured as slave.

### 13.2.2 MISO — Master In/Slave Out Pin

This pin is used to transmit data out of the SPI module when it is configured as a slave and receive data when it is configured as master.

### 13.2.3 $\overline{SS}$ — Slave Select Pin

This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when it is configured as a master and it is used as an input to receive the slave select signal when the SPI is configured as slave.

### 13.2.4 SCK — Serial Clock Pin

In master mode, this is the synchronous output clock. In slave mode, this is the synchronous input clock.

## 13.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the SPI.

### 13.3.1 Module Memory Map

The memory map for the SPI is given in Figure 13-2. The address listed for each register is the sum of a base address and an address offset. The base address is defined at the SoC level and the address offset is defined at the module level. Reads from the reserved bits return zeros and writes to the reserved bits have no effect.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 SPICR1 | R | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| | W | | | | | | | | |
| 0x0001 SPICR2 | R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| | W | | | | | | | | |
| 0x0002 SPIBR | R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| | W | | | | | | | | |
| 0x0003 SPISR | R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0004 SPIDRH | R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| | W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| 0x0005 SPIDRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 0x0006 Reserved | R | | | | | | | | |
| | W | | | | | | | | |
| 0x0007 Reserved | R | | | | | | | | |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 13-2. SPI Register Summary**

## 13.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

### 13.3.2.1 SPI Control Register 1 (SPICR1)

Module Base +0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 13-3. SPI Control Register 1 (SPICR1)**

Read: Anytime

Write: Anytime

**Table 13-2. SPICR1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIE | **SPI Interrupt Enable Bit** — This bit enables SPI interrupt requests, if SPIF or MODF status flag is set.<br>0 SPI interrupts disabled.<br>1 SPI interrupts enabled. |
| 6<br>SPE | **SPI System Enable Bit** — This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, status bits in SPISR register are reset.<br>0 SPI disabled (lower power consumption).<br>1 SPI enabled, port pins are dedicated to SPI functions. |
| 5<br>SPTIE | **SPI Transmit Interrupt Enable** — This bit enables SPI interrupt requests, if SPTEF flag is set.<br>0 SPTEF interrupt disabled.<br>1 SPTEF interrupt enabled. |
| 4<br>MSTR | **SPI Master/Slave Mode Select Bit** — This bit selects whether the SPI operates in master or slave mode. Switching the SPI from master to slave or vice versa forces the SPI system into idle state.<br>0 SPI is in slave mode.<br>1 SPI is in master mode. |
| 3<br>CPOL | **SPI Clock Polarity Bit** — This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 Active-high clocks selected. In idle state SCK is low.<br>1 Active-low clocks selected. In idle state SCK is high. |
| 2<br>CPHA | **SPI Clock Phase Bit** — This bit is used to select the SPI clock format. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 Sampling of data occurs at odd edges (1,3,5,...) of the SCK clock.<br>1 Sampling of data occurs at even edges (2,4,6,...) of the SCK clock. |

**Table 13-2. SPICR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SSOE | **Slave Select Output Enable** — The $\overline{SS}$ output feature is enabled only in master mode, if MODFEN is set, by asserting the SSOE as shown in Table 13-3. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. |
| 0<br>LSBFE | **LSB-First Enable** — This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in the highest bit position. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 Data is transferred most significant bit first.<br>1 Data is transferred least significant bit first. |

**Table 13-3. $\overline{SS}$ Input / Output Selection**

| MODFEN | SSOE | Master Mode | Slave Mode |
|---|---|---|---|
| 0 | 0 | $\overline{SS}$ not used by SPI | $\overline{SS}$ input |
| 0 | 1 | $\overline{SS}$ not used by SPI | $\overline{SS}$ input |
| 1 | 0 | $\overline{SS}$ input with MODF feature | $\overline{SS}$ input |
| 1 | 1 | $\overline{SS}$ is slave select output | $\overline{SS}$ input |

### 13.3.2.2 SPI Control Register 2 (SPICR2)

Module Base +0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-4. SPI Control Register 2 (SPICR2)**

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

**Table 13-4. SPICR2 Field Descriptions**

| Field | Description |
|---|---|
| 6<br>XFRW | **Transfer Width —** This bit is used for selecting the data transfer width. If 8-bit transfer width is selected, SPIDRL becomes the dedicated data register and SPIDRH is unused. If 16-bit transfer width is selected, SPIDRH and SPIDRL form a 16-bit data register. Please refer to Section 13.3.2.4, “SPI Status Register (SPISR) for information about transmit/receive data handling and the interrupt flag clearing mechanism. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 8-bit Transfer Width (n = 8)[1]<br>1 16-bit Transfer Width (n = 16)[1] |
| 4<br>MODFEN | **Mode Fault Enable Bit** — This bit allows the MODF failure to be detected. If the SPI is in master mode and MODFEN is cleared, then the $\overline{SS}$ port pin is not used by the SPI. In slave mode, the $\overline{SS}$ is available only as an input regardless of the value of MODFEN. For an overview on the impact of the MODFEN bit on the $\overline{SS}$ port pin configuration, refer to Table 13-3. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 $\overline{SS}$ port pin is not used by the SPI.<br>1 $\overline{SS}$ port pin with MODF feature. |
| 3<br>BIDIROE | **Output Enable in the Bidirectional Mode of Operation** — This bit controls the MOSI and MISO output buffer of the SPI, when in bidirectional mode of operation (SPC0 is set). In master mode, this bit controls the output buffer of the MOSI port, in slave mode it controls the output buffer of the MISO port. In master mode, with SPC0 set, a change of this bit will abort a transmission in progress and force the SPI into idle state.<br>0 Output buffer disabled.<br>1 Output buffer enabled. |
| 1<br>SPISWAI | **SPI Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode.<br>0 SPI clock operates normally in wait mode.<br>1 Stop SPI clock generation when in wait mode. |
| 0<br>SPC0 | **Serial Pin Control Bit 0** — This bit enables bidirectional pin configurations as shown in Table 13-5. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. |

[1] n is used later in this document as a placeholder for the selected transfer width.

**Table 13-5. Bidirectional Pin Configurations**

| Pin Mode | SPC0 | BIDIROE | MISO | MOSI |
|---|---|---|---|---|
| **Master Mode of Operation** | | | | |
| Normal | 0 | X | Master In | Master Out |
| Bidirectional | 1 | 0 | MISO not used by SPI | Master In |
| | | 1 | | Master I/O |
| **Slave Mode of Operation** | | | | |
| Normal | 0 | X | Slave Out | Slave In |
| Bidirectional | 1 | 0 | Slave In | MOSI not used by SPI |
| | | 1 | Slave I/O | |

### 13.3.2.3 SPI Baud Rate Register (SPIBR)

Module Base +0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-5. SPI Baud Rate Register (SPIBR)**

Read: Anytime

Write: Anytime; writes to the reserved bits have no effect

**Table 13-6. SPIBR Field Descriptions**

| Field | Description |
|---|---|
| 6–4<br>SPPR[2:0] | **SPI Baud Rate Preselection Bits** — These bits specify the SPI baud rates as shown in Table 13-7. In master mode, a change of these bits will abort a transmission in progress and force the SPI system into idle state. |
| 2–0<br>SPR[2:0] | **SPI Baud Rate Selection Bits** — These bits specify the SPI baud rates as shown in Table 13-7. In master mode, a change of these bits will abort a transmission in progress and force the SPI system into idle state. |

The baud rate divisor equation is as follows:

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)} \qquad \textit{Eqn. 13-1}$$

The baud rate can be calculated with the following equation:

$$\text{Baud Rate} = \text{BusClock} / \text{BaudRateDivisor} \qquad \textit{Eqn. 13-2}$$

**NOTE**

For maximum allowed baud rates, please refer to the SPI Electrical Specification in the Electricals chapter of this data sheet.

**Table 13-7. Example SPI Baud Rate Selection (25 MHz Bus Clock) (Sheet 1 of 3)**

| SPPR2 | SPPR1 | SPPR0 | SPR2 | SPR1 | SPR0 | Baud Rate Divisor | Baud Rate |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 12.5 Mbit/s |
| 0 | 0 | 0 | 0 | 0 | 1 | 4 | 6.25 Mbit/s |
| 0 | 0 | 0 | 0 | 1 | 0 | 8 | 3.125 Mbit/s |
| 0 | 0 | 0 | 0 | 1 | 1 | 16 | 1.5625 Mbit/s |
| 0 | 0 | 0 | 1 | 0 | 0 | 32 | 781.25 kbit/s |
| 0 | 0 | 0 | 1 | 0 | 1 | 64 | 390.63 kbit/s |
| 0 | 0 | 0 | 1 | 1 | 0 | 128 | 195.31 kbit/s |
| 0 | 0 | 0 | 1 | 1 | 1 | 256 | 97.66 kbit/s |
| 0 | 0 | 1 | 0 | 0 | 0 | 4 | 6.25 Mbit/s |
| 0 | 0 | 1 | 0 | 0 | 1 | 8 | 3.125 Mbit/s |
| 0 | 0 | 1 | 0 | 1 | 0 | 16 | 1.5625 Mbit/s |
| 0 | 0 | 1 | 0 | 1 | 1 | 32 | 781.25 kbit/s |

**Table 13-7. Example SPI Baud Rate Selection (25 MHz Bus Clock) (Sheet 2 of 3)**

| SPPR2 | SPPR1 | SPPR0 | SPR2 | SPR1 | SPR0 | Baud Rate Divisor | Baud Rate |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 64 | 390.63 kbit/s |
| 0 | 0 | 1 | 1 | 0 | 1 | 128 | 195.31 kbit/s |
| 0 | 0 | 1 | 1 | 1 | 0 | 256 | 97.66 kbit/s |
| 0 | 0 | 1 | 1 | 1 | 1 | 512 | 48.83 kbit/s |
| 0 | 1 | 0 | 0 | 0 | 0 | 6 | 4.16667 Mbit/s |
| 0 | 1 | 0 | 0 | 0 | 1 | 12 | 2.08333 Mbit/s |
| 0 | 1 | 0 | 0 | 1 | 0 | 24 | 1.04167 Mbit/s |
| 0 | 1 | 0 | 0 | 1 | 1 | 48 | 520.83 kbit/s |
| 0 | 1 | 0 | 1 | 0 | 0 | 96 | 260.42 kbit/s |
| 0 | 1 | 0 | 1 | 0 | 1 | 192 | 130.21 kbit/s |
| 0 | 1 | 0 | 1 | 1 | 0 | 384 | 65.10 kbit/s |
| 0 | 1 | 0 | 1 | 1 | 1 | 768 | 32.55 kbit/s |
| 0 | 1 | 1 | 0 | 0 | 0 | 8 | 3.125 Mbit/s |
| 0 | 1 | 1 | 0 | 0 | 1 | 16 | 1.5625 Mbit/s |
| 0 | 1 | 1 | 0 | 1 | 0 | 32 | 781.25 kbit/s |
| 0 | 1 | 1 | 0 | 1 | 1 | 64 | 390.63 kbit/s |
| 0 | 1 | 1 | 1 | 0 | 0 | 128 | 195.31 kbit/s |
| 0 | 1 | 1 | 1 | 0 | 1 | 256 | 97.66 kbit/s |
| 0 | 1 | 1 | 1 | 1 | 0 | 512 | 48.83 kbit/s |
| 0 | 1 | 1 | 1 | 1 | 1 | 1024 | 24.41 kbit/s |
| 1 | 0 | 0 | 0 | 0 | 0 | 10 | 2.5 Mbit/s |
| 1 | 0 | 0 | 0 | 0 | 1 | 20 | 1.25 Mbit/s |
| 1 | 0 | 0 | 0 | 1 | 0 | 40 | 625 kbit/s |
| 1 | 0 | 0 | 0 | 1 | 1 | 80 | 312.5 kbit/s |
| 1 | 0 | 0 | 1 | 0 | 0 | 160 | 156.25 kbit/s |
| 1 | 0 | 0 | 1 | 0 | 1 | 320 | 78.13 kbit/s |
| 1 | 0 | 0 | 1 | 1 | 0 | 640 | 39.06 kbit/s |
| 1 | 0 | 0 | 1 | 1 | 1 | 1280 | 19.53 kbit/s |
| 1 | 0 | 1 | 0 | 0 | 0 | 12 | 2.08333 Mbit/s |
| 1 | 0 | 1 | 0 | 0 | 1 | 24 | 1.04167 Mbit/s |
| 1 | 0 | 1 | 0 | 1 | 0 | 48 | 520.83 kbit/s |
| 1 | 0 | 1 | 0 | 1 | 1 | 96 | 260.42 kbit/s |
| 1 | 0 | 1 | 1 | 0 | 0 | 192 | 130.21 kbit/s |
| 1 | 0 | 1 | 1 | 0 | 1 | 384 | 65.10 kbit/s |
| 1 | 0 | 1 | 1 | 1 | 0 | 768 | 32.55 kbit/s |
| 1 | 0 | 1 | 1 | 1 | 1 | 1536 | 16.28 kbit/s |
| 1 | 1 | 0 | 0 | 0 | 0 | 14 | 1.78571 Mbit/s |
| 1 | 1 | 0 | 0 | 0 | 1 | 28 | 892.86 kbit/s |
| 1 | 1 | 0 | 0 | 1 | 0 | 56 | 446.43 kbit/s |
| 1 | 1 | 0 | 0 | 1 | 1 | 112 | 223.21 kbit/s |
| 1 | 1 | 0 | 1 | 0 | 0 | 224 | 111.61 kbit/s |
| 1 | 1 | 0 | 1 | 0 | 1 | 448 | 55.80 kbit/s |

**Table 13-7. Example SPI Baud Rate Selection (25 MHz Bus Clock) (Sheet 3 of 3)**

| SPPR2 | SPPR1 | SPPR0 | SPR2 | SPR1 | SPR0 | Baud Rate Divisor | Baud Rate |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 896 | 27.90 kbit/s |
| 1 | 1 | 0 | 1 | 1 | 1 | 1792 | 13.95 kbit/s |
| 1 | 1 | 1 | 0 | 0 | 0 | 16 | 1.5625 Mbit/s |
| 1 | 1 | 1 | 0 | 0 | 1 | 32 | 781.25 kbit/s |
| 1 | 1 | 1 | 0 | 1 | 0 | 64 | 390.63 kbit/s |
| 1 | 1 | 1 | 0 | 1 | 1 | 128 | 195.31 kbit/s |
| 1 | 1 | 1 | 1 | 0 | 0 | 256 | 97.66 kbit/s |
| 1 | 1 | 1 | 1 | 0 | 1 | 512 | 48.83 kbit/s |
| 1 | 1 | 1 | 1 | 1 | 0 | 1024 | 24.41 kbit/s |
| 1 | 1 | 1 | 1 | 1 | 1 | 2048 | 12.21 kbit/s |

### 13.3.2.4 SPI Status Register (SPISR)

Module Base +0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-6. SPI Status Register (SPISR)**

Read: Anytime

Write: Has no effect

**Table 13-8. SPISR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIF | **SPIF Interrupt Flag** — This bit is set after received data has been transferred into the SPI data register. For information about clearing SPIF Flag, please refer to Table 13-9.<br>0 Transfer not yet complete.<br>1 New data copied to SPIDR. |
| 5<br>SPTEF | **SPI Transmit Empty Interrupt Flag** — If set, this bit indicates that the transmit data register is empty. For information about clearing this bit and placing data into the transmit data register, please refer to Table 13-10.<br>0 SPI data register not empty.<br>1 SPI data register empty. |
| 4<br>MODF | **Mode Fault Flag** — This bit is set if the $\overline{SS}$ input becomes low while the SPI is configured as a master and mode fault detection is enabled, MODFEN bit of SPICR2 register is set. Refer to MODFEN bit description in Section 13.3.2.2, “SPI Control Register 2 (SPICR2)”. The flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to the SPI control register 1.<br>0 Mode fault has not occurred.<br>1 Mode fault has occurred. |

**Table 13-9. SPIF Interrupt Flag Clearing Sequence**

| XFRW Bit | SPIF Interrupt Flag Clearing Sequence | | | |
|---|---|---|---|---|
| 0 | Read SPISR with SPIF == 1 | **then** | Read SPIDRL | |
| 1 | Read SPISR with SPIF == 1 | **then** | Byte Read SPIDRL [1] | |
| | | | **or** | |
| | | | Byte Read SPIDRH [2] | Byte Read SPIDRL |
| | | | **or** | |
| | | | Word Read (SPIDRH:SPIDRL) | |

1 Data in SPIDRH is lost in this case.

2 SPIDRH can be read repeatedly without any effect on SPIF. SPIF Flag is cleared only by the read of SPIDRL after reading SPISR with SPIF == 1.

**Table 13-10. SPTEF Interrupt Flag Clearing Sequence**

| XFRW Bit | SPTEF Interrupt Flag Clearing Sequence | | | |
|---|---|---|---|---|
| 0 | Read SPISR with SPTEF == 1 | **then** | Write to SPIDRL [1] | |
| 1 | Read SPISR with SPTEF == 1 | **then** | Byte Write to SPIDRL [12] | |
| | | | **or** | |
| | | | Byte Write to SPIDRH [13] | Byte Write to SPIDRL [1] |
| | | | **or** | |
| | | | Word Write to (SPIDRH:SPIDRL) [1] | |

1 Any write to SPIDRH or SPIDRL with SPTEF == 0 is effectively ignored.

2 Data in SPIDRH is undefined in this case.

3 SPIDRH can be written repeatedly without any effect on SPTEF. SPTEF Flag is cleared only by writing to SPIDRL after reading SPISR with SPTEF == 1.

### 13.3.2.5 SPI Data Register (SPIDR = SPIDRH:SPIDRL)

Module Base +0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| W | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-7. SPI Data Register High (SPIDRH)**

Module Base +0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-8. SPI Data Register Low (SPIDRL)**

Read: Anytime; read data only valid when SPIF is set

Write: Anytime

The SPI data register is both the input and output register for SPI data. A write to this register allows data to be queued and transmitted. For an SPI configured as a master, queued data is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag SPTEF in the SPISR register indicates when the SPI data register is ready to accept new data.

Received data in the SPIDR is valid when SPIF is set.

If SPIF is cleared and data has been received, the received data is transferred from the receive shift register to the SPIDR and SPIF is set.

If SPIF is set and not serviced, and a second data value has been received, the second received data is kept as valid data in the receive shift register until the start of another transmission. The data in the SPIDR does not change.

If SPIF is set and valid data is in the receive shift register, and SPIF is serviced before the start of a third transmission, the data in the receive shift register is transferred into the SPIDR and SPIF remains set (see Figure 13-9).

If SPIF is set and valid data is in the receive shift register, and SPIF is serviced after the start of a third transmission, the data in the receive shift register has become invalid and is not transferred into the SPIDR (see Figure 13-10).

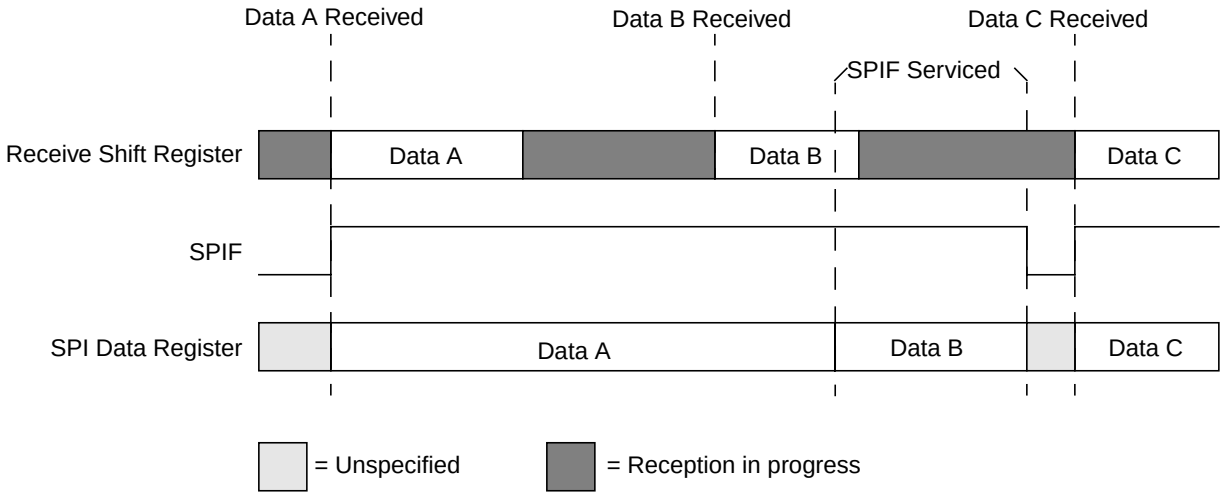



**Figure 13-9. Reception with SPIF serviced in Time**

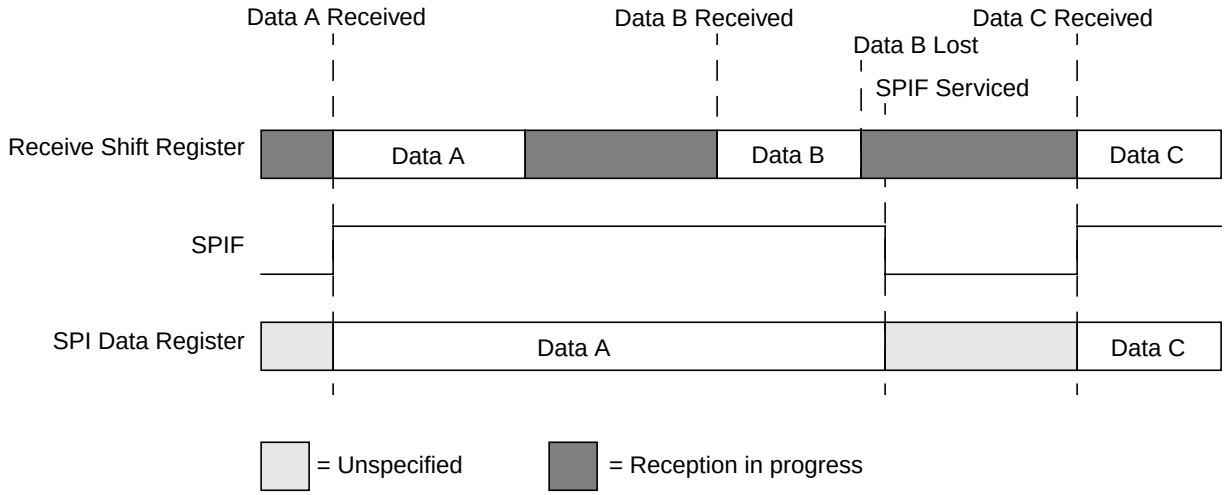



**Figure 13-10. Reception with SPIF serviced too late**

## 13.4 Functional Description

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or SPI operation can be interrupt driven.

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI control register 1. While SPE is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select ($\overline{SS}$)
- Serial clock (SCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

The main element of the SPI system is the SPI data register. The n-bit[1] data register in the master and the n-bit[1] data register in the slave are linked by the MOSI and MISO pins to form a distributed 2n-bit[1] register. When a data transfer operation is performed, this 2n-bit[1] register is serially shifted n[1] bit positions by the S-clock from the master, so data is exchanged between the master and the slave. Data written to the master SPI data register becomes the output data for the slave, and data read from the master SPI data register after a transfer operation is the input data from the slave.

A read of SPISR with SPTEF = 1 followed by a write to SPIDR puts data into the transmit data register. When a transfer is complete and SPIF is cleared, received data is moved into the receive data register. This data register acts as the SPI receive data register for reads and as the SPI transmit data register for writes. A common SPI data register address is shared for reading data from the read data buffer and for writing data to the transmit data register.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI control register 1 (SPICR1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SCK edges or on even numbered SCK edges (see Section 13.4.3, “Transmission Formats”).

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI control register1 is set, master mode is selected, when the MSTR bit is clear, slave mode is selected.

**NOTE**

A change of CPOL or MSTR bit while there is a received byte pending in the receive shift register will destroy the received byte and must be avoided.

### 13.4.1 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI data register. If the shift register is empty, data immediately transfers to the shift register. Data begins shifting out on the MOSI pin under the control of the serial clock.

- Serial clock

  The SPR2, SPR1, and SPR0 baud rate selection bits, in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI baud rate register, control the baud rate generator and determine the speed of the transmission. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.
- MOSI, MISO pin

  In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.
- $\overline{SS}$ pin

  If MODFEN and SSOE are set, the $\overline{SS}$ pin is configured as slave select output. The $\overline{SS}$ output becomes low during each transmission and is high when the SPI is in idle state.

1. n depends on the selected transfer width, please refer to Section 13.3.2.2, “SPI Control Register 2 (SPICR2)

If MODFEN is set and SSOE is cleared, the $\overline{SS}$ pin is configured as input for detecting mode fault error. If the $\overline{SS}$ input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SCK, MOSI, and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI status register (SPISR). If the SPI interrupt enable bit (SPIE) is set when the MODF flag becomes set, then an SPI interrupt sequence is also requested.

When a write to the SPI data register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI control register 1 (see Section 13.4.3, “Transmission Formats”).

**NOTE**

A change of the bits CPOL, CPHA, SSOE, LSBFE, XFRW, MODFEN, SPC0, or BIDIROE with SPC0 set, SPPR2-SPPR0 and SPR2-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master must ensure that the remote slave is returned to idle state.

## 13.4.2 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI control register 1 is clear.

- Serial clock

  In slave mode, SCK is the SPI clock input from the master.
- MISO, MOSI pin

  In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI control register 2.
- $\overline{SS}$ pin

  The $\overline{SS}$ pin is the slave select input. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be low. $\overline{SS}$ must remain low until the transmission is complete. If $\overline{SS}$ goes high, the SPI is forced into idle state.

  The $\overline{SS}$ input also controls the serial data output pin, if $\overline{SS}$ is high (not selected), the serial data output pin is high impedance, and, if $\overline{SS}$ is low, the first bit in the SPI data register is driven out of the serial data output pin. Also, if the slave is not selected ($\overline{SS}$ is high), then the SCK input is ignored and no internal shifting of the SPI shift register occurs.

  Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

**NOTE**

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI control register 1 is clear, odd numbered edges on the SCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the $\overline{SS}$ input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the nth[1] shift, the transfer is considered complete and the received data is transferred into the SPI data register. To indicate transfer is complete, the SPIF flag in the SPI status register is set.

**NOTE**

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, or BIDIROE with SPC0 set in slave mode will corrupt a transmission in progress and must be avoided.

## 13.4.3 Transmission Formats

During an SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. Optionally, on a master SPI device, the slave select line can be used to indicate multiple-master bus contention.

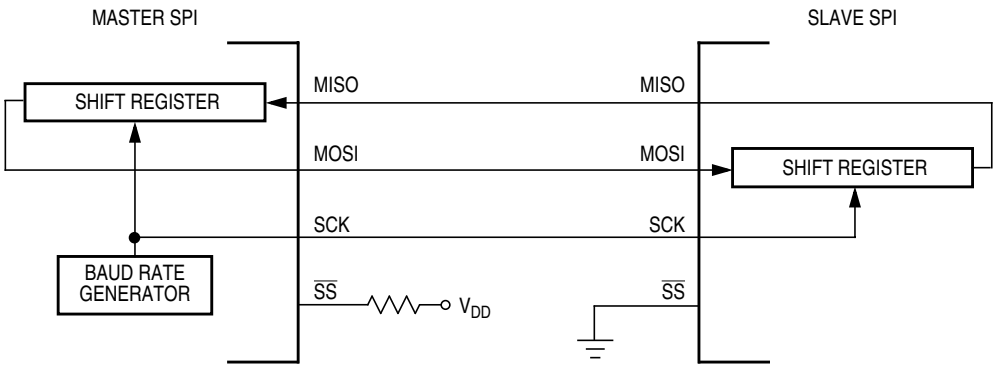



**Figure 13-11. Master/Slave Transfer Block Diagram**

1. n depends on the selected transfer width, please refer to Section 13.3.2.2, "SPI Control Register 2 (SPICR2)

#### 13.4.3.1 Clock Phase and Polarity Controls

Using two bits in the SPI control register 1, software selects one of four combinations of serial clock phase and polarity.

The CPOL clock polarity control bit specifies an active high or low clock and has no significant effect on the transmission format.

The CPHA clock phase control bit selects one of two fundamentally different transmission formats.

Clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

#### 13.4.3.2 CPHA = 0 Transfer Format

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after $\overline{SS}$ has become low.

A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the shift register, depending on LSBFE bit.

After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total of 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

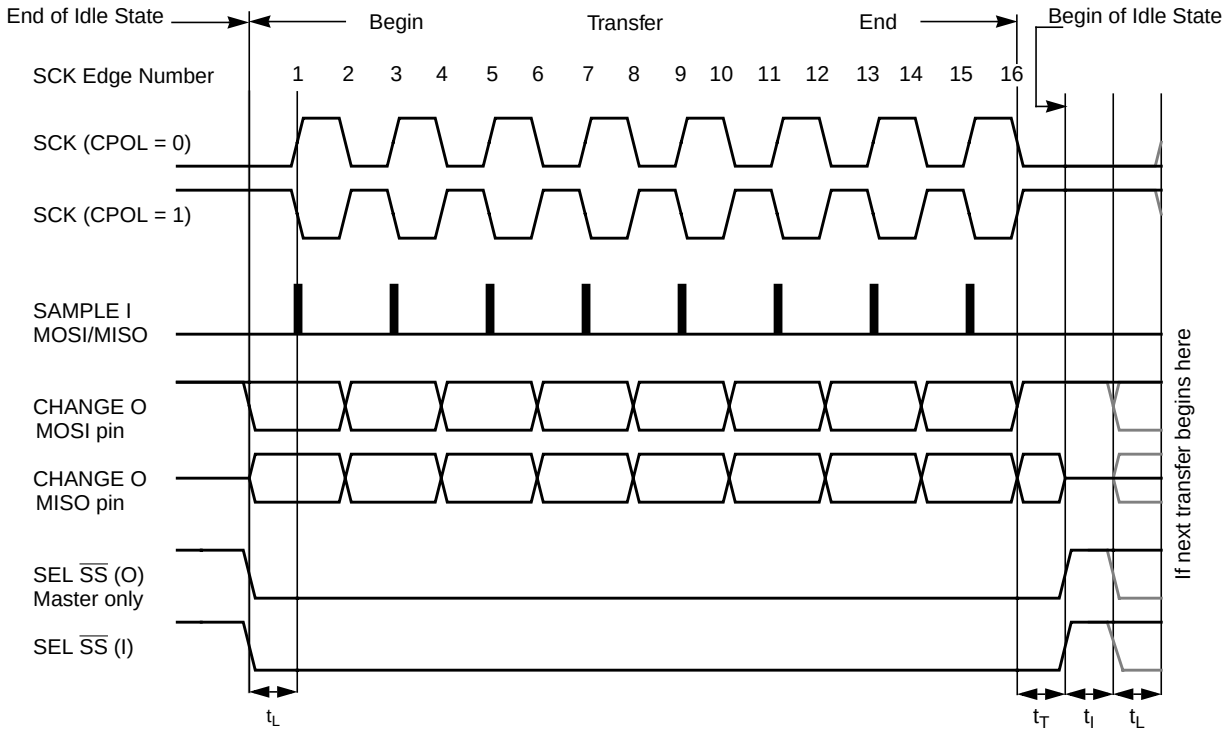
Data reception is double buffered. Data is shifted serially into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After $2n^1$ (last) SCK edges:

- Data that was previously in the master SPI data register should now be in the slave data register and the data that was in the slave data register should be in the master.
- The SPIF flag in the SPI status register is set, indicating that the transfer is complete.

Figure 13-12 is a timing diagram of an SPI transfer where CPHA = 0. SCK waveforms are shown for CPOL = 0 and CPOL = 1. The diagram may be interpreted as a master or slave timing diagram because the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave and the MOSI signal is the output from the master. The $\overline{SS}$ pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.

1. n depends on the selected transfer width, please refer to Section 13.3.2.2, "SPI Control Register 2 (SPICR2)



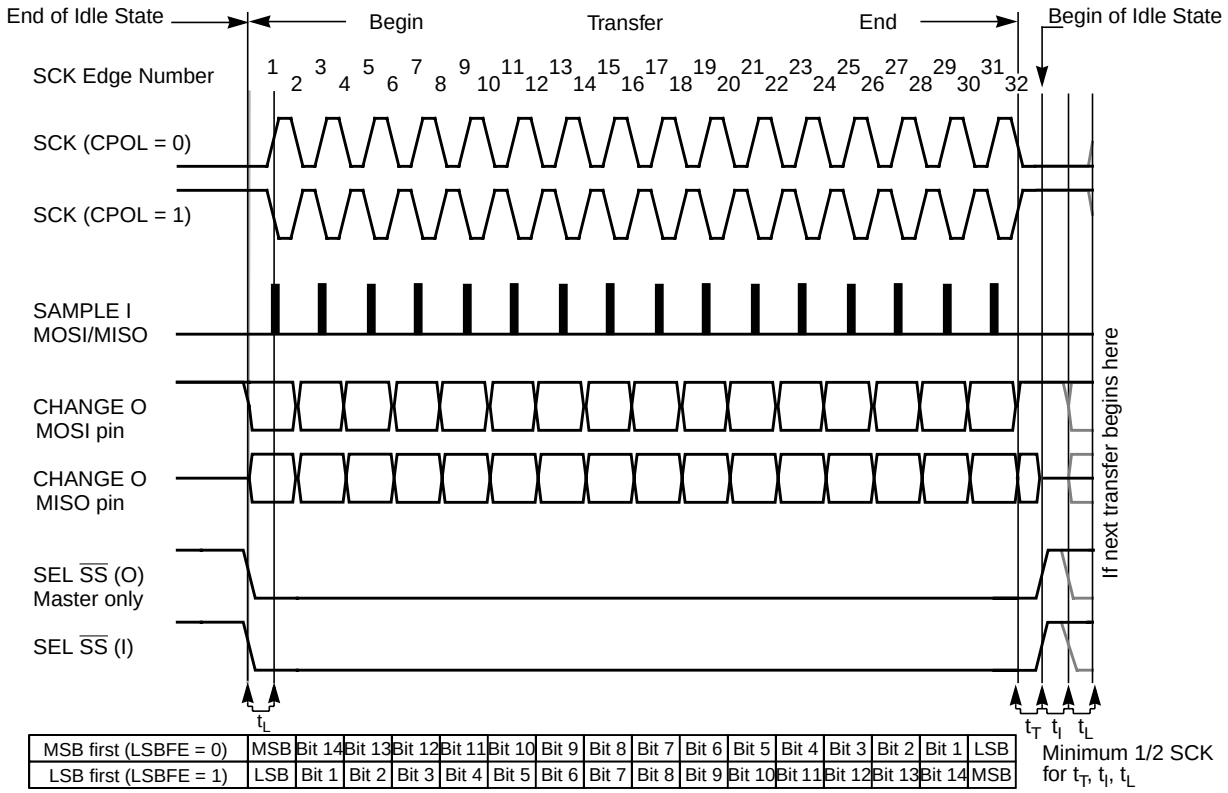

$t_L$ = Minimum leading time before the first SCK edge

$t_T$ = Minimum trailing time after the last SCK edge

$t_I$ = Minimum idling time between transfers (minimum $\overline{SS}$ high time)

$t_L$, $t_T$, and $t_I$ are guaranteed for the master mode and required for the slave mode.

**Figure 13-12. SPI Clock Format 0 (CPHA = 0), with 8-bit Transfer Width selected (XFRW = 0)**




$t_L$ = Minimum leading time before the first SCK edge
$t_T$ = Minimum trailing time after the last SCK edge
$t_I$ = Minimum idling time between transfers (minimum $\overline{SS}$ high time)
$t_L$, $t_T$, and $t_I$ are guaranteed for the master mode and required for the slave mode.

**Figure 13-13. SPI Clock Format 0 (CPHA = 0), with 16-Bit Transfer Width selected (XFRW = 1)**

In slave mode, if the $\overline{SS}$ line is not deasserted between the successive transmissions then the content of the SPI data register is not transmitted; instead the last received data is transmitted. If the $\overline{SS}$ line is deasserted for at least minimum idle time (half SCK cycle) between successive transmissions, then the content of the SPI data register is transmitted.

In master mode, with slave select output enabled the $\overline{SS}$ line is always deasserted and reasserted between successive transfers for at least minimum idle time.

### 13.4.3.3 CPHA = 1 Transfer Format

Some peripherals require the first SCK edge before the first data bit becomes available at the data out pin, the second edge clocks data into the system. In this format, the first SCK edge is issued by setting the CPHA bit at the beginning of the n[1]-cycle transfer operation.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its first data bit to the serial data input pin of the master.

A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

1. n depends on the selected transfer width, please refer to Section 13.3.2.2, “SPI Control Register 2 (SPICR2)

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the SPI shift register, depending on LSBFE bit. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pin on the slave.

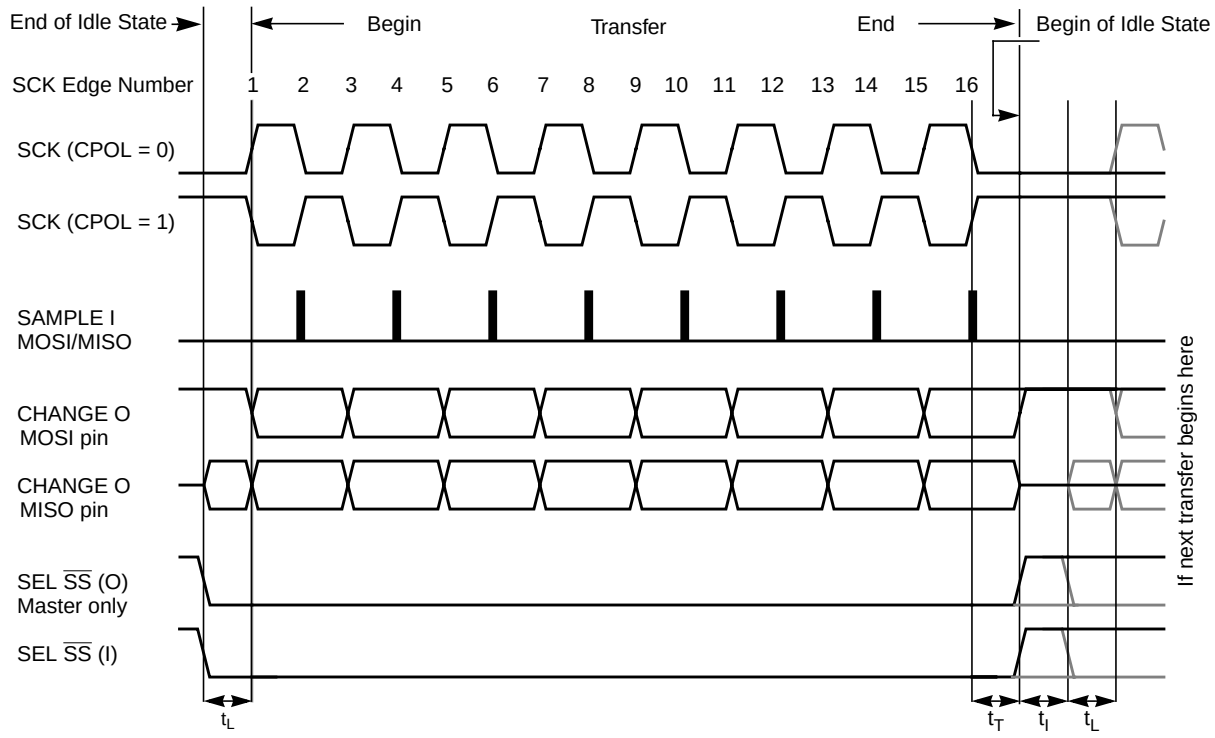
This process continues for a total of n[1] edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

Data reception is double buffered, data is serially shifted into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After 2n[1] SCK edges:

- Data that was previously in the SPI data register of the master is now in the data register of the slave, and data that was in the data register of the slave is in the master.
- The SPIF flag bit in SPISR is set indicating that the transfer is complete.

Figure 13-14 shows two clocking variations for CPHA = 1. The diagram may be interpreted as a master or slave timing diagram because the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The $\overline{SS}$ pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.




$t_L$ = Minimum leading time before the first SCK edge, not required for back-to-back transfers
$t_T$ = Minimum trailing time after the last SCK edge
$t_I$ = Minimum idling time between transfers (minimum $\overline{SS}$ high time), not required for back-to-back transfers

**Figure 13-14. SPI Clock Format 1 (CPHA = 1), with 8-Bit Transfer Width selected (XFRW = 0)**

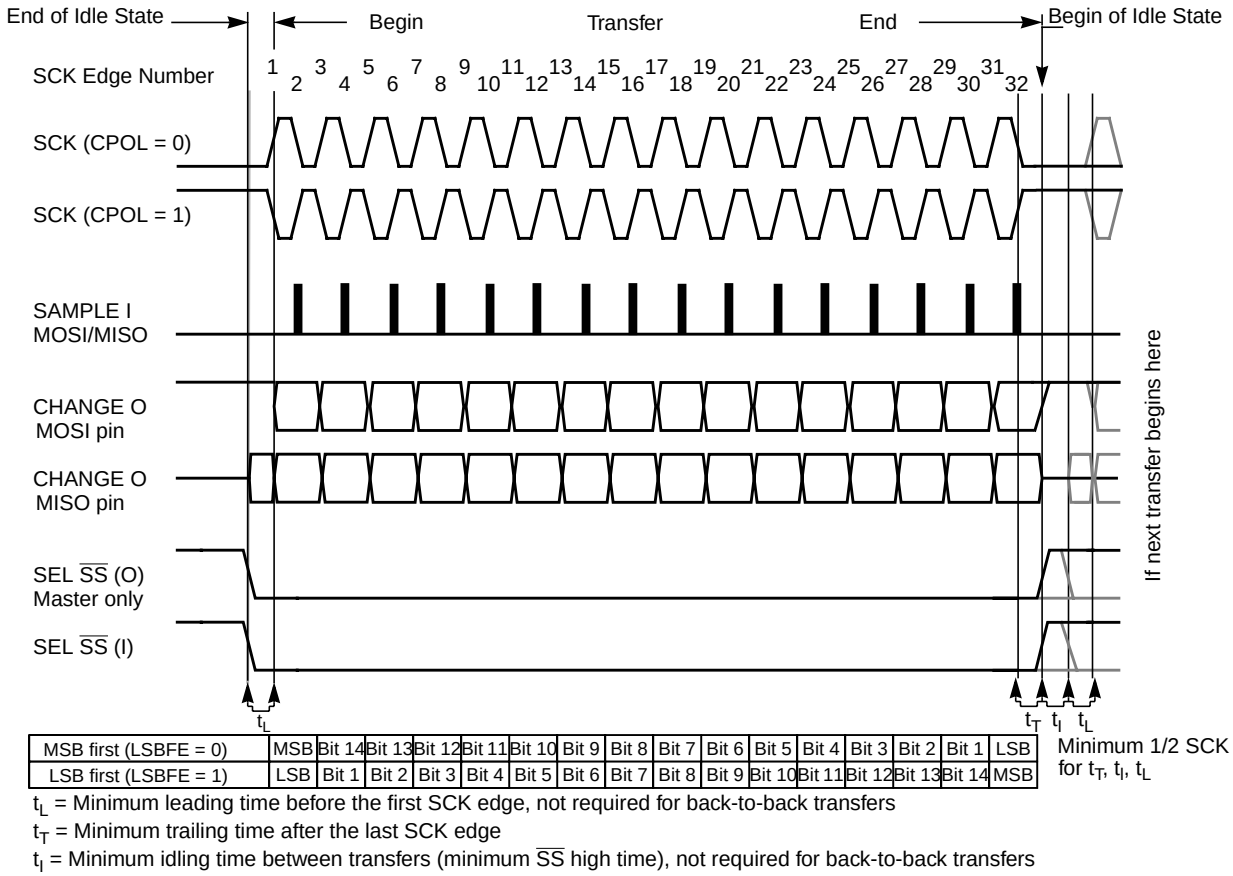



**Figure 13-15. SPI Clock Format 1 (CPHA = 1), with 16-Bit Transfer Width selected (XFRW = 1)**

The $\overline{SS}$ line can remain active low between successive transfers (can be tied low at all times). This format is sometimes preferred in systems having a single fixed master and a single slave that drive the MISO data line.

- Back-to-back transfers in master mode

  In master mode, if a transmission has completed and new data is available in the SPI data register, this data is sent out immediately without a trailing and minimum idle time.

The SPI interrupt request flag (SPIF) is common to both the master and slave modes. SPIF gets set one half SCK cycle after the last SCK edge.

## 13.4.4 SPI Baud Rate Generation

Baud rate generation consists of a series of divider stages. Six bits in the SPI baud rate register (SPPR2, SPPR1, SPPR0, SPR2, SPR1, and SPR0) determine the divisor to the SPI module clock which results in the SPI baud rate.

The SPI clock rate is determined by the product of the value in the baud rate preselection bits (SPPR2–SPPR0) and the value in the baud rate selection bits (SPR2–SPR0). The module clock divisor equation is shown in Equation 13-3.

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)} \qquad \textit{Eqn. 13-3}$$

When all bits are clear (the default condition), the SPI module clock is divided by 2. When the selection bits (SPR2–SPR0) are 001 and the preselection bits (SPPR2–SPPR0) are 000, the module clock divisor becomes 4. When the selection bits are 010, the module clock divisor becomes 8, etc.

When the preselection bits are 001, the divisor determined by the selection bits is multiplied by 2. When the preselection bits are 010, the divisor is multiplied by 3, etc. See Table 13-7 for baud rate calculations for all bit conditions, based on a 25 MHz bus clock. The two sets of selects allows the clock to be divided by a non-power of two to achieve other baud rates such as divide by 6, divide by 10, etc.

The baud rate generator is activated only when the SPI is in master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease $I_{DD}$ current.

**NOTE**

For maximum allowed baud rates, please refer to the SPI Electrical Specification in the Electricals chapter of this data sheet.

## 13.4.5 Special Features

### 13.4.5.1 $\overline{SS}$ Output

The $\overline{SS}$ output feature automatically drives the $\overline{SS}$ pin low during transmission to select external devices and drives it high during idle to deselect external devices. When $\overline{SS}$ output is selected, the $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external device.

The $\overline{SS}$ output is available only in master mode during normal SPI operation by asserting SSOE and MODFEN bit as shown in Table 13-3.

The mode fault feature is disabled while $\overline{SS}$ output is enabled.

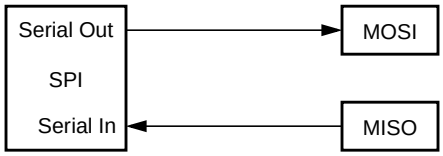
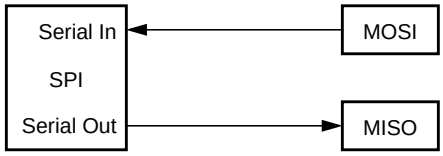
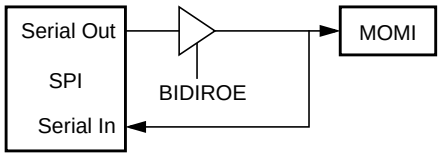
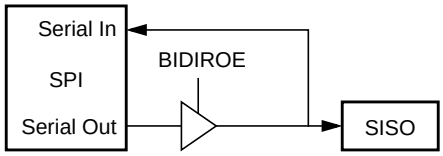
**NOTE**

Care must be taken when using the $\overline{SS}$ output feature in a multimaster system because the mode fault feature is not available for detecting system errors between masters.

### 13.4.5.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI control register 2 (see Table 13-11). In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

**Table 13-11. Normal Mode and Bidirectional Mode**

| When SPE = 1 | Master Mode MSTR = 1 | Slave Mode MSTR = 0 |
|---|---|---|
| **Normal Mode SPC0 = 0** |  |  |
| **Bidirectional Mode SPC0 = 1** |  |  |

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

- The SCK is output for the master mode and input for the slave mode.
- The $\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.
- The bidirectional mode does not affect SCK and $\overline{SS}$ functions.

**NOTE**

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case MISO becomes occupied by the SPI and MOSI is not used. This must be considered, if the MISO pin is used for another purpose.

## 13.4.6 Error Conditions

The SPI has one error condition:

- Mode fault error

### 13.4.6.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SCK lines simultaneously. This condition is not permitted in normal operation, the MODF bit in the SPI status register is set automatically, provided the MODFEN bit is set.

In the special case where the SPI is in master mode and MODFEN bit is cleared, the $\overline{SS}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. In case

the SPI system is configured as a slave, the $\overline{SS}$ pin is a dedicated input pin. Mode fault error doesn't occur in slave mode.

If a mode fault error occurs, the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So SCK, MISO, and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for a SPI system configured in master mode, output enable of the MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to SPI control register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

**NOTE**

If a mode fault error occurs and a received data byte is pending in the receive shift register, this data byte will be lost.

## 13.4.7 Low Power Mode Options

### 13.4.7.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers remain accessible, but clocks to the core of this module are disabled.

### 13.4.7.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI control register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SCK continues to be driven from the master. This keeps the slave synchronized to the master and the SCK.

    If the master transmits several bytes while the slave is in wait mode, the slave will continue to send out bytes consistent with the operation mode at the start of wait mode (i.e., if the slave is currently sending its SPIDR to the master, it will continue to send the same byte. Else if the slave is currently sending the last received byte from the master, it will continue to send each previous master byte).

**NOTE**

Care must be taken when expecting data from a master while the slave is in wait or stop mode. Even though the shift register will continue to operate, the rest of the SPI is shut down (i.e., a SPIF interrupt will **not** be generated until exiting stop or wait mode). Also, the byte from the shift register will not be copied into the SPIDR register until after the slave SPI has exited wait or stop mode. In slave mode, a received byte pending in the receive shift register will be lost when entering wait or stop mode. An SPIF flag and SPIDR copy is generated only if wait mode is entered or exited during a tranmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither a SPIF nor a SPIDR copy will occur.

#### 13.4.7.3 SPI in Stop Mode

Stop mode is dependent on the system. The SPI enters stop mode when the module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the CPU enters stop mode, the transmission is frozen until the CPU exits stop mode. After stop, data to and from the external SPI is exchanged correctly. In slave mode, the SPI will stay synchronized with the master.

The stop mode is not dependent on the SPISWAI bit.

#### 13.4.7.4 Reset

The reset values of registers and signals are described in Section 13.3, “Memory Map and Register Definition”, which details the registers and their bit fields.

- If a data transmission occurs in slave mode after reset without a write to SPIDR, it will transmit garbage, or the data last received from the master before the reset.
- Reading from the SPIDR after reset will always read zeros.

#### 13.4.7.5 Interrupts

The SPI only originates interrupt requests when SPI is enabled (SPE bit in SPICR1 set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

The interrupt flags MODF, SPIF, and SPTEF are logically ORed to generate an interrupt request.

##### 13.4.7.5.1 MODF

MODF occurs when the master detects an error on the $\overline{SS}$ pin. The master SPI must be configured for the MODF feature (see Table 13-3). After MODF is set, the current transfer is aborted and the following bit is changed:

- MSTR = 0, The master bit in SPICR1 resets.

The MODF interrupt is reflected in the status register MODF flag. Clearing the flag will also clear the interrupt. This interrupt will stay active while the MODF flag is set. MODF has an automatic clearing process which is described in Section 13.3.2.4, “SPI Status Register (SPISR)”.

#### 13.4.7.5.2 SPIF

SPIF occurs when new data has been received and copied to the SPI data register. After SPIF is set, it does not clear until it is serviced. SPIF has an automatic clearing process, which is described in Section 13.3.2.4, “SPI Status Register (SPISR)”.

#### 13.4.7.5.3 SPTEF

SPTEF occurs when the SPI data register is ready to accept new data. After SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process, which is described in Section 13.3.2.4, “SPI Status Register (SPISR)”.

# Chapter 14
# Inter-Integrated Circuit (IICV3) Block Description

**Table 14-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.03 | 28 Jul 2006 | 14.7.1.7/14-559 | - Update flow-chart of interrupt routine for 10-bit address |
| V01.04 | 17 Nov 2006 | 14.3.1.2/14-539 | - Revise Table1-5 |
| V01.05 | 14 Aug 2007 | 14.3.1.1/14-539 | - Backward compatible for IBAD bit name |

## 14.1 Introduction

The inter-IC bus (IIC) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. Being a two-wire device, the IIC bus minimizes the need for large numbers of connections between devices, and eliminates the need for an address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

### 14.1.1 Features

The IIC module has the following key features:

- Compatible with I2C bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation

- Acknowledge bit generation/detection
- Bus busy detection
- General Call Address detection
- Compliant to ten-bit address

### 14.1.2 Modes of Operation

The IIC functions the same in normal, special, and emulation modes. It has two low power modes: wait and stop modes.

### 14.1.3 Block Diagram

The block diagram of the IIC module is shown in Figure 14-1.

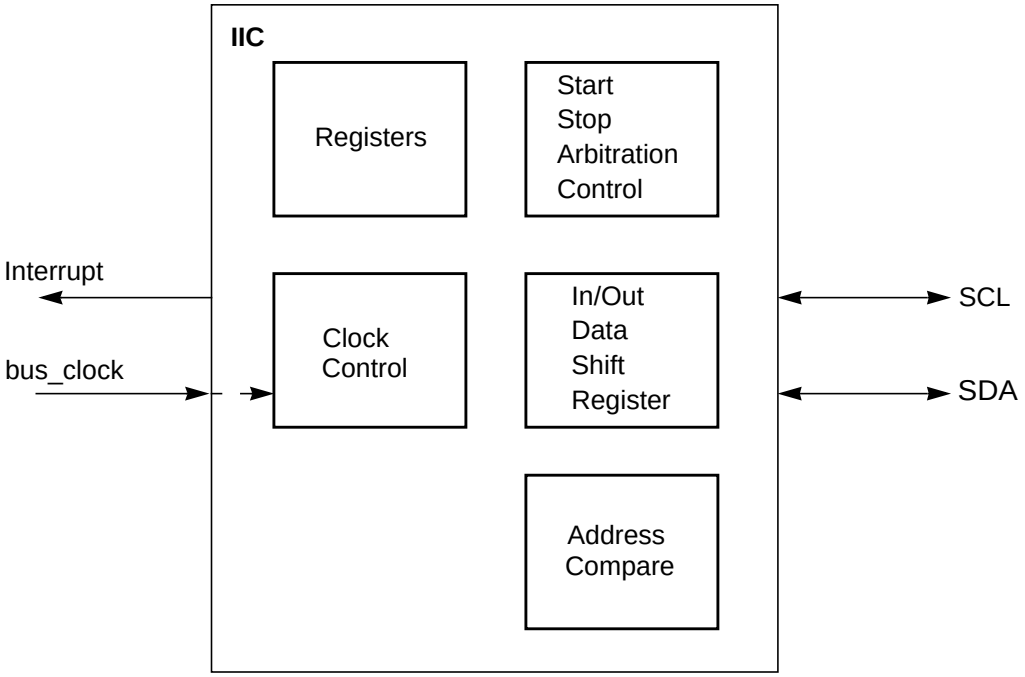



**Figure 14-1. IIC Block Diagram**

## 14.2 External Signal Description

The IICV3 module has two external pins.

### 14.2.1 IIC_SCL — Serial Clock Line Pin

This is the bidirectional serial clock line (SCL) of the module, compatible to the IIC bus specification.

### 14.2.2 IIC_SDA — Serial Data Line Pin

This is the bidirectional serial data line (SDA) of the module, compatible to the IIC bus specification.

## 14.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers for the IIC module.

### 14.3.1 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 IBAD | R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| | W | | | | | | | | |
| 0x0001 IBFD | R | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| | W | | | | | | | | |
| 0x0002 IBCR | R | IBEN | IBIE | MS/$\overline{SL}$ | Tx/$\overline{Rx}$ | TXAK | 0 | 0 | IBSWAI |
| | W | | | | | | RSTA | | |
| 0x0003 IBSR | R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| | W | | | | | | | | |
| 0x0004 IBDR | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | W | | | | | | | | |
| 0x0005 IBCR2 | R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 14-2. IIC Register Summary**

### 14.3.1.1 IIC Address Register (IBAD)

Module Base +0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-3. IIC Bus Address Register (IBAD)**

Read and write anytime

This register contains the address the IIC bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

**Table 14-2. IBAD Field Descriptions**

| Field | Description |
|---|---|
| 7:1<br>ADR[7:1] | **Slave Address** — Bit 1 to bit 7 contain the specific slave address to be used by the IIC bus module.The default mode of IIC bus is slave mode for an address match on the bus. |
| 0<br>Reserved | Reserved — Bit 0 of the IBAD is reserved for future compatibility. This bit will always read 0. |

### 14.3.1.2 IIC Frequency Divider Register (IBFD)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-4. IIC Bus Frequency Divider Register (IBFD)**

Read and write anytime

**Table 14-3. IBFD Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>IBC[7:0] | **I Bus Clock Rate 7:0** — This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider — IBC7:6, prescaled shift register — IBC5:3 select the prescaler divider and IBC2-0 select the shift register tap point. The IBC bits are decoded to give the tap and prescale values as shown in Table 14-4. |

**Table 14-4. I-Bus Tap and Prescale Values**

| IBC2-0 (bin) | SCL Tap (clocks) | SDA Tap (clocks) |
|---|---|---|
| 000 | 5 | 1 |
| 001 | 6 | 1 |
| 010 | 7 | 2 |
| 011 | 8 | 2 |
| 100 | 9 | 3 |
| 101 | 10 | 3 |
| 110 | 12 | 4 |
| 111 | 15 | 4 |

**Table 14-5. Prescale Divider Encoding**

| IBC5-3 (bin) | scl2start (clocks) | scl2stop (clocks) | scl2tap (clocks) | tap2tap (clocks) |
|---|---|---|---|---|
| 000 | 2 | 7 | 4 | 1 |
| 001 | 2 | 7 | 4 | 2 |
| 010 | 2 | 9 | 6 | 4 |
| 011 | 6 | 9 | 6 | 8 |
| 100 | 14 | 17 | 14 | 16 |
| 101 | 30 | 33 | 30 | 32 |
| 110 | 62 | 65 | 62 | 64 |
| 111 | 126 | 129 | 126 | 128 |

**Table 14-6. Multiplier Factor**

| IBC7-6 | MUL |
|---|---|
| 00 | 01 |
| 01 | 02 |
| 10 | 04 |
| 11 | RESERVED |

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 14-4, all subsequent tap points are separated by $2^{IBC5-3}$ as shown in the tap2tap column in Table 14-5. The SCL Tap is used to generated the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.

IBC7–6 defines the multiplier factor MUL. The values of MUL are shown in the Table 14-6.

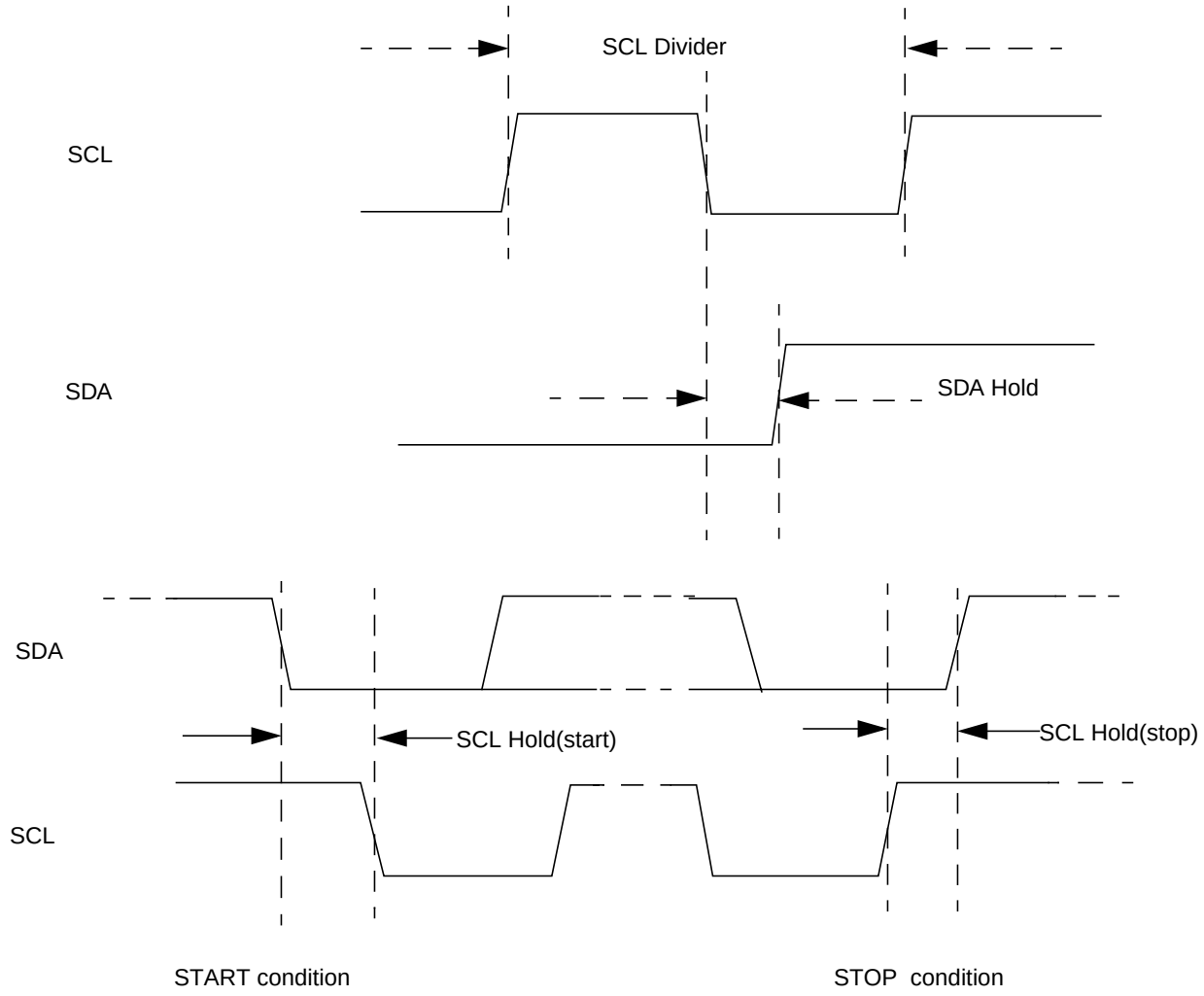



**Figure 14-5. SCL Divider and SDA Hold**

The equation used to generate the divider values from the IBFD bits is:

**SCL Divider = MUL x {2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)}**

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 14-7. The equation used to generate the SDA Hold value from the IBFD bits is:

**SDA Hold = MUL x {scl2tap + [(SDA_Tap - 1) x tap2tap] + 3}**

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

**SCL Hold(start) = MUL x [scl2start + (SCL_Tap - 1) x tap2tap]**

**SCL Hold(stop) = MUL x [scl2stop + (SCL_Tap - 1) x tap2tap]**

**Table 14-7. IIC Divider and Hold Values (Sheet 1 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| **MUL=1** | | | | |

**Table 14-7. IIC Divider and Hold Values (Sheet 2 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 00 | 20/22 | 7 | 6 | 11 |
| 01 | 22/24 | 7 | 7 | 12 |
| 02 | 24/26 | 8 | 8 | 13 |
| 03 | 26/28 | 8 | 9 | 14 |
| 04 | 28/30 | 9 | 10 | 15 |
| 05 | 30/32 | 9 | 11 | 16 |
| 06 | 34/36 | 10 | 13 | 18 |
| 07 | 40/42 | 10 | 16 | 21 |
| 08 | 28/32 | 7 | 10 | 15 |
| 09 | 32/36 | 7 | 12 | 17 |
| 0A | 36/40 | 9 | 14 | 19 |
| 0B | 40/44 | 9 | 16 | 21 |
| 0C | 44/48 | 11 | 18 | 23 |
| 0D | 48/52 | 11 | 20 | 25 |
| 0E | 56/60 | 13 | 24 | 29 |
| 0F | 68/72 | 13 | 30 | 35 |
| 10 | 48 | 9 | 18 | 25 |
| 11 | 56 | 9 | 22 | 29 |
| 12 | 64 | 13 | 26 | 33 |
| 13 | 72 | 13 | 30 | 37 |
| 14 | 80 | 17 | 34 | 41 |
| 15 | 88 | 17 | 38 | 45 |
| 16 | 104 | 21 | 46 | 53 |
| 17 | 128 | 21 | 58 | 65 |
| 18 | 80 | 9 | 38 | 41 |
| 19 | 96 | 9 | 46 | 49 |
| 1A | 112 | 17 | 54 | 57 |
| 1B | 128 | 17 | 62 | 65 |
| 1C | 144 | 25 | 70 | 73 |
| 1D | 160 | 25 | 78 | 81 |
| 1E | 192 | 33 | 94 | 97 |
| 1F | 240 | 33 | 118 | 121 |
| 20 | 160 | 17 | 78 | 81 |
| 21 | 192 | 17 | 94 | 97 |
| 22 | 224 | 33 | 110 | 113 |
| 23 | 256 | 33 | 126 | 129 |
| 24 | 288 | 49 | 142 | 145 |
| 25 | 320 | 49 | 158 | 161 |
| 26 | 384 | 65 | 190 | 193 |
| 27 | 480 | 65 | 238 | 241 |
| 28 | 320 | 33 | 158 | 161 |
| 29 | 384 | 33 | 190 | 193 |
| 2A | 448 | 65 | 222 | 225 |
| 2B | 512 | 65 | 254 | 257 |
| 2C | 576 | 97 | 286 | 289 |

**Table 14-7. IIC Divider and Hold Values (Sheet 3 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 2D | 640 | 97 | 318 | 321 |
| 2E | 768 | 129 | 382 | 385 |
| 2F | 960 | 129 | 478 | 481 |
| 30 | 640 | 65 | 318 | 321 |
| 31 | 768 | 65 | 382 | 385 |
| 32 | 896 | 129 | 446 | 449 |
| 33 | 1024 | 129 | 510 | 513 |
| 34 | 1152 | 193 | 574 | 577 |
| 35 | 1280 | 193 | 638 | 641 |
| 36 | 1536 | 257 | 766 | 769 |
| 37 | 1920 | 257 | 958 | 961 |
| 38 | 1280 | 129 | 638 | 641 |
| 39 | 1536 | 129 | 766 | 769 |
| 3A | 1792 | 257 | 894 | 897 |
| 3B | 2048 | 257 | 1022 | 1025 |
| 3C | 2304 | 385 | 1150 | 1153 |
| 3D | 2560 | 385 | 1278 | 1281 |
| 3E | 3072 | 513 | 1534 | 1537 |
| 3F | 3840 | 513 | 1918 | 1921 |
| **MUL=2** | | | | |
| 40 | 40 | 14 | 12 | 22 |
| 41 | 44 | 14 | 14 | 24 |
| 42 | 48 | 16 | 16 | 26 |
| 43 | 52 | 16 | 18 | 28 |
| 44 | 56 | 18 | 20 | 30 |
| 45 | 60 | 18 | 22 | 32 |
| 46 | 68 | 20 | 26 | 36 |
| 47 | 80 | 20 | 32 | 42 |
| 48 | 56 | 14 | 20 | 30 |
| 49 | 64 | 14 | 24 | 34 |
| 4A | 72 | 18 | 28 | 38 |
| 4B | 80 | 18 | 32 | 42 |
| 4C | 88 | 22 | 36 | 46 |
| 4D | 96 | 22 | 40 | 50 |
| 4E | 112 | 26 | 48 | 58 |
| 4F | 136 | 26 | 60 | 70 |
| 50 | 96 | 18 | 36 | 50 |
| 51 | 112 | 18 | 44 | 58 |
| 52 | 128 | 26 | 52 | 66 |
| 53 | 144 | 26 | 60 | 74 |
| 54 | 160 | 34 | 68 | 82 |
| 55 | 176 | 34 | 76 | 90 |
| 56 | 208 | 42 | 92 | 106 |
| 57 | 256 | 42 | 116 | 130 |
| 58 | 160 | 18 | 76 | 82 |

**Table 14-7. IIC Divider and Hold Values (Sheet 4 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 59 | 192 | 18 | 92 | 98 |
| 5A | 224 | 34 | 108 | 114 |
| 5B | 256 | 34 | 124 | 130 |
| 5C | 288 | 50 | 140 | 146 |
| 5D | 320 | 50 | 156 | 162 |
| 5E | 384 | 66 | 188 | 194 |
| 5F | 480 | 66 | 236 | 242 |
| 60 | 320 | 34 | 156 | 162 |
| 61 | 384 | 34 | 188 | 194 |
| 62 | 448 | 66 | 220 | 226 |
| 63 | 512 | 66 | 252 | 258 |
| 64 | 576 | 98 | 284 | 290 |
| 65 | 640 | 98 | 316 | 322 |
| 66 | 768 | 130 | 380 | 386 |
| 67 | 960 | 130 | 476 | 482 |
| 68 | 640 | 66 | 316 | 322 |
| 69 | 768 | 66 | 380 | 386 |
| 6A | 896 | 130 | 444 | 450 |
| 6B | 1024 | 130 | 508 | 514 |
| 6C | 1152 | 194 | 572 | 578 |
| 6D | 1280 | 194 | 636 | 642 |
| 6E | 1536 | 258 | 764 | 770 |
| 6F | 1920 | 258 | 956 | 962 |
| 70 | 1280 | 130 | 636 | 642 |
| 71 | 1536 | 130 | 764 | 770 |
| 72 | 1792 | 258 | 892 | 898 |
| 73 | 2048 | 258 | 1020 | 1026 |
| 74 | 2304 | 386 | 1148 | 1154 |
| 75 | 2560 | 386 | 1276 | 1282 |
| 76 | 3072 | 514 | 1532 | 1538 |
| 77 | 3840 | 514 | 1916 | 1922 |
| 78 | 2560 | 258 | 1276 | 1282 |
| 79 | 3072 | 258 | 1532 | 1538 |
| 7A | 3584 | 514 | 1788 | 1794 |
| 7B | 4096 | 514 | 2044 | 2050 |
| 7C | 4608 | 770 | 2300 | 2306 |
| 7D | 5120 | 770 | 2556 | 2562 |
| 7E | 6144 | 1026 | 3068 | 3074 |
| 7F | 7680 | 1026 | 3836 | 3842 |
| **MUL=4** | | | | |
| 80 | 72 | 28 | 24 | 44 |
| 81 | 80 | 28 | 28 | 48 |
| 82 | 88 | 32 | 32 | 52 |
| 83 | 96 | 32 | 36 | 56 |
| 84 | 104 | 36 | 40 | 60 |

**Table 14-7. IIC Divider and Hold Values (Sheet 5 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| 85 | 112 | 36 | 44 | 64 |
| 86 | 128 | 40 | 52 | 72 |
| 87 | 152 | 40 | 64 | 84 |
| 88 | 112 | 28 | 40 | 60 |
| 89 | 128 | 28 | 48 | 68 |
| 8A | 144 | 36 | 56 | 76 |
| 8B | 160 | 36 | 64 | 84 |
| 8C | 176 | 44 | 72 | 92 |
| 8D | 192 | 44 | 80 | 100 |
| 8E | 224 | 52 | 96 | 116 |
| 8F | 272 | 52 | 120 | 140 |
| 90 | 192 | 36 | 72 | 100 |
| 91 | 224 | 36 | 88 | 116 |
| 92 | 256 | 52 | 104 | 132 |
| 93 | 288 | 52 | 120 | 148 |
| 94 | 320 | 68 | 136 | 164 |
| 95 | 352 | 68 | 152 | 180 |
| 96 | 416 | 84 | 184 | 212 |
| 97 | 512 | 84 | 232 | 260 |
| 98 | 320 | 36 | 152 | 164 |
| 99 | 384 | 36 | 184 | 196 |
| 9A | 448 | 68 | 216 | 228 |
| 9B | 512 | 68 | 248 | 260 |
| 9C | 576 | 100 | 280 | 292 |
| 9D | 640 | 100 | 312 | 324 |
| 9E | 768 | 132 | 376 | 388 |
| 9F | 960 | 132 | 472 | 484 |
| A0 | 640 | 68 | 312 | 324 |
| A1 | 768 | 68 | 376 | 388 |
| A2 | 896 | 132 | 440 | 452 |
| A3 | 1024 | 132 | 504 | 516 |
| A4 | 1152 | 196 | 568 | 580 |
| A5 | 1280 | 196 | 632 | 644 |
| A6 | 1536 | 260 | 760 | 772 |
| A7 | 1920 | 260 | 952 | 964 |
| A8 | 1280 | 132 | 632 | 644 |
| A9 | 1536 | 132 | 760 | 772 |
| AA | 1792 | 260 | 888 | 900 |
| AB | 2048 | 260 | 1016 | 1028 |
| AC | 2304 | 388 | 1144 | 1156 |
| AD | 2560 | 388 | 1272 | 1284 |
| AE | 3072 | 516 | 1528 | 1540 |
| AF | 3840 | 516 | 1912 | 1924 |
| B0 | 2560 | 260 | 1272 | 1284 |
| B1 | 3072 | 260 | 1528 | 1540 |

**Table 14-7. IIC Divider and Hold Values (Sheet 6 of 6)**

| IBC[7:0] (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|
| B2 | 3584 | 516 | 1784 | 1796 |
| B3 | 4096 | 516 | 2040 | 2052 |
| B4 | 4608 | 772 | 2296 | 2308 |
| B5 | 5120 | 772 | 2552 | 2564 |
| B6 | 6144 | 1028 | 3064 | 3076 |
| B7 | 7680 | 1028 | 3832 | 3844 |
| B8 | 5120 | 516 | 2552 | 2564 |
| B9 | 6144 | 516 | 3064 | 3076 |
| BA | 7168 | 1028 | 3576 | 3588 |
| BB | 8192 | 1028 | 4088 | 4100 |
| BC | 9216 | 1540 | 4600 | 4612 |
| BD | 10240 | 1540 | 5112 | 5124 |
| BE | 12288 | 2052 | 6136 | 6148 |
| BF | 15360 | 2052 | 7672 | 7684 |

Note:Since the bus frequency is speeding up,the SCL Divider could be expanded by it.Therefore,in the table,when IBC[7:0] is from $00 to $0F,the SCL Divider is revised by the format value1/value2.Value1 is the divider under the low frequency.Value2 is the divider under the high frequency.How to select the divider depends on the bus frequency.When IBC[7:0] is from $10 to $BF,the divider is not changed.

### 14.3.1.3 IIC Control Register (IBCR)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IBEN | IBIE | MS/SL | Tx/Rx | TXAK | 0 | 0 | IBSWAI |
| W | | | | | | RSTA | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-6. IIC Bus Control Register (IBCR)**

Read and write anytime

**Table 14-8. IBCR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IBEN | **I-Bus Enable** — This bit controls the software reset of the entire IIC bus module.<br>0 The module is reset and disabled. This is the power-on reset situation. When low the interface is held in reset but registers can be accessed<br>1 The IIC bus module is enabled.This bit must be set before any other IBCR bits have any effect<br>If the IIC bus module is enabled in the middle of a byte transfer the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the IIC bus module losing arbitration, after which bus operation would return to normal. |
| 6<br>IBIE | **I-Bus Interrupt Enable**<br>0 Interrupts from the IIC bus module are disabled. Note that this does not clear any currently pending interrupt condition<br>1 Interrupts from the IIC bus module are enabled. An IIC bus interrupt occurs provided the IBIF bit in the status register is also set. |
| 5<br>MS/SL | **Master/Slave Mode Select Bit** — Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus, and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave.A STOP signal should only be generated if the IBIF flag is set. MS/$\overline{SL}$ is cleared without generating a STOP signal when the master loses arbitration.<br>0 Slave Mode<br>1 Master Mode |
| 4<br>Tx/Rx | **Transmit/Receive Mode Select Bit** — This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high.<br>0 Receive<br>1 Transmit |
| 3<br>TXAK | **Transmit Acknowledge Enable** — This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The IIC module will always acknowledge address matches, provided it is enabled, regardless of the value of TXAK. Note that values written to this bit are only used when the IIC bus is a receiver, not a transmitter.<br>0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte data<br>1 No acknowledge signal response is sent (i.e., acknowledge bit = 1) |
| 2<br>RSTA | **Repeat Start** — Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration.<br>1 Generate repeat start cycle |
| 1<br>RESERVED | **Reserved** — Bit 1 of the IBCR is reserved for future compatibility. This bit will always read 0. |
| 0<br>IBSWAI | **I Bus Interface Stop in Wait Mode**<br>0 IIC bus module clock operates normally<br>1 Halt IIC bus module clock generation in wait mode |

Wait mode is entered via execution of a CPU WAI instruction. In the event that the IBSWAI bit is set, all clocks internal to the IIC will be stopped and any transmission currently in progress will halt.If the CPU were woken up by a source other than the IIC module, then clocks would restart and the IIC would resume

from where was during the previous transmission. It is not possible for the IIC to wake up the CPU when its internal clocks are stopped.

If it were the case that the IBSWAI bit was cleared when the WAI instruction was executed, the IIC internal clocks and interface would remain alive, continuing the operation which was currently underway. It is also possible to configure the IIC such that it will wake up the CPU via an interrupt at the conclusion of the current operation. See the discussion on the IBIF and IBIE bits in the IBSR and IBCR, respectively.

### 14.3.1.4 IIC Status Register (IBSR)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-7. IIC Bus Status Register (IBSR)**

This status register is read-only with exception of bit 1 (IBIF) and bit 4 (IBAL), which are software clearable.

**Table 14-9. IBSR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TCF | **Data Transferring Bit** — While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.<br>0 Transfer in progress<br>1 Transfer complete |
| 6<br>IAAS | **Addressed as a Slave Bit** — When its own specific address (I-bus address register) is matched with the calling address or it receives the general call address with GCEN== 1,this bit is set.The CPU is interrupted provided the IBIE is set.Then the CPU needs to check the SRW bit and set its Tx/$\overline{Rx}$ mode accordingly.Writing to the I-bus control register clears this bit.<br>0 Not addressed<br>1 Addressed as a slave |
| 5<br>IBB | Bus Busy Bit<br>0 This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state.<br>1 Bus is busy |
| 4<br>IBAL | **Arbitration Lost** — The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances:<br>1. SDA sampled low when the master drives a high during an address or data transmit cycle.<br>2. SDA sampled low when the master drives a high during the acknowledge bit of a data receive cycle.<br>3. A start cycle is attempted when the bus is busy.<br>4. A repeated start cycle is requested in slave mode.<br>5. A stop condition is detected when the master did not request it.<br>This bit must be cleared by software, by writing a one to it. A write of 0 has no effect on this bit. |

**Table 14-9. IBSR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>RESERVED | **Reserved** — Bit 3 of IBSR is reserved for future use. A read operation on this bit will return 0. |
| 2<br>SRW | **Slave Read/Write** — When IAAS is set this bit indicates the value of the R/W command bit of the calling address sent from the master<br>This bit is only valid when the I-bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated.<br>Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.<br>0 Slave receive, master writing to slave<br>1 Slave transmit, master reading from slave |
| 1<br>IBIF | **I-Bus Interrupt** — The IBIF bit is set when one of the following conditions occurs:<br>— Arbitration lost (IBAL bit set)<br>— Data transfer complete (TCF bit set)<br>— Addressed as slave (IAAS bit set)<br>It will cause a processor interrupt request if the IBIE bit is set. This bit must be cleared by software, writing a one to it. A write of 0 has no effect on this bit. |
| 0<br>RXAK | **Received Acknowledge** — The value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock.<br>0 Acknowledge received<br>1 No acknowledge received |

### 14.3.1.5 IIC Data I/O Register (IBDR)

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-8. IIC Bus Data I/O Register (IBDR)**

In master transmit mode, when data is written to the IBDR a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.Note that the Tx/Rx bit in the IBCR must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MS/$\overline{\text{SL}}$ is used for the address transfer and should com.prise of the calling address (in position D7:D1) concatenated with the required R/$\overline{\text{W}}$ bit (in position D0).

### 14.3.1.6 IIC Control Register 2(IBCR2)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-9. IIC Bus Control Register 2(IBCR2)**

This register contains the variables used in general call and in ten-bit address.

Read and write anytime

**Table 14-10. IBCR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>GCEN | **General Call Enable**.<br>0 General call is disabled. The module dont receive any general call data and address.<br>1 enable general call. It indicates that the module can receive address and any data. |
| 6<br>ADTYPE | **Address Type—** This bit selects the address length. The variable must be configured correctly before IIC enters slave mode.<br>0 7-bit address<br>1 10-bit address |
| 5,4,3<br>RESERVED | **Reserved** — Bit 5,4 and 3 of the IBCR2 are reserved for future compatibility. These bits will always read 0. |
| 2:0<br>ADR[10:8] | **Slave Address [10:8]** —These 3 bits represent the MSB of the 10-bit address when address type is asserted (ADTYPE = 1). |

## 14.4 Functional Description

This section provides a complete functional description of the IICV3.

### 14.4.1 I-Bus Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. Logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in Figure 14-10.

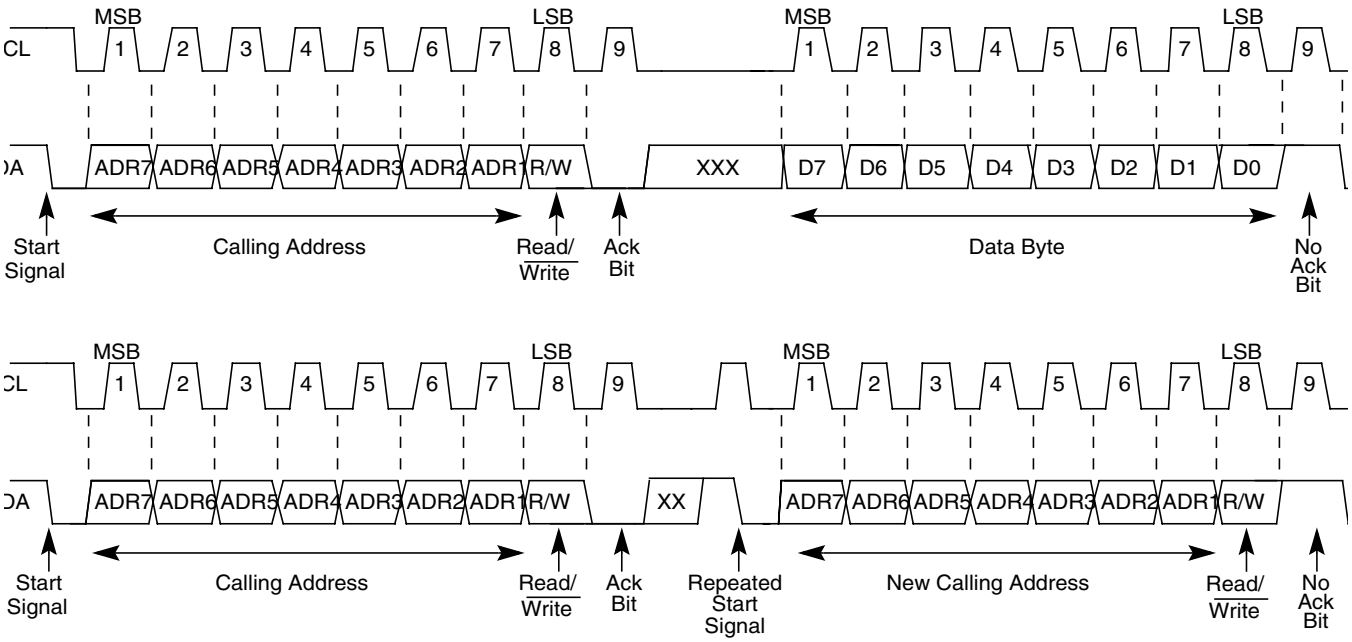



**Figure 14-10. IIC-Bus Transmission Signals**

### 14.4.1.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal.As shown in Figure 14-10, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

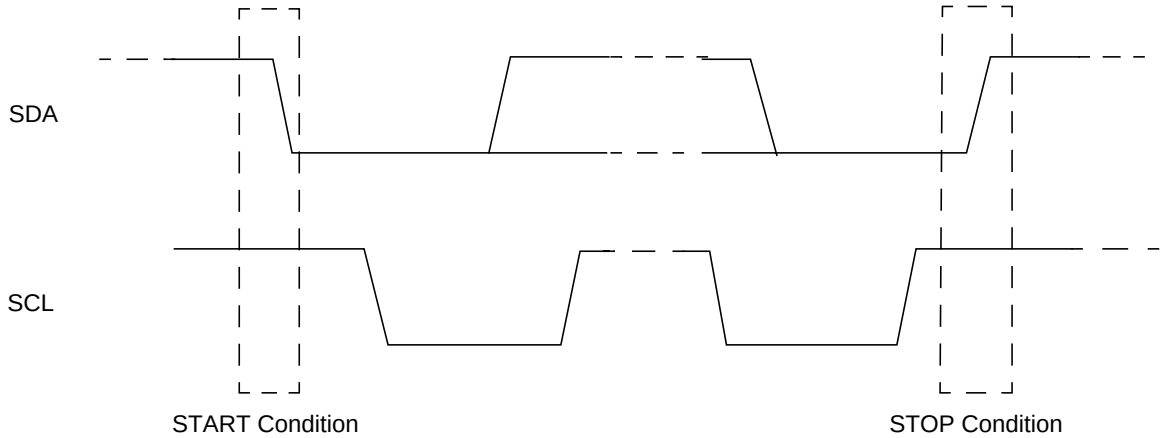



**Figure 14-11. Start and Stop Conditions**

#### 14.4.1.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

If the calling address is 10-bit, another byte is followed by the first byte.Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 14-10).

No two slaves in the system may have the same address. If the IIC bus is master, it must not transmit an address that is equal to its own slave address. The IIC bus cannot be master and slave at the same time.However, if arbitration is lost during an address cycle the IIC bus will revert to slave mode and operate correctly even if it is being addressed by another master.

#### 14.4.1.3 Data Transfer

As soon as successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 14-10. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate STOP or START signal.Note in order to release the bus correctly,after no-acknowledge to the master,the slave must be immediately switched to receiver and a following dummy reading of the IBDR is necessary.

#### 14.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 14-10).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

#### 14.4.1.5 Repeated START Signal

As shown in Figure 14-10, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

#### 14.4.1.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

#### 14.4.1.7 Clock Synchronization

Because wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period and as soon as a device's clock has gone low, it holds the SCL line low until the clock high state is reached.However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 14-11). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods.The first device to complete its high period pulls the SCL line low again.

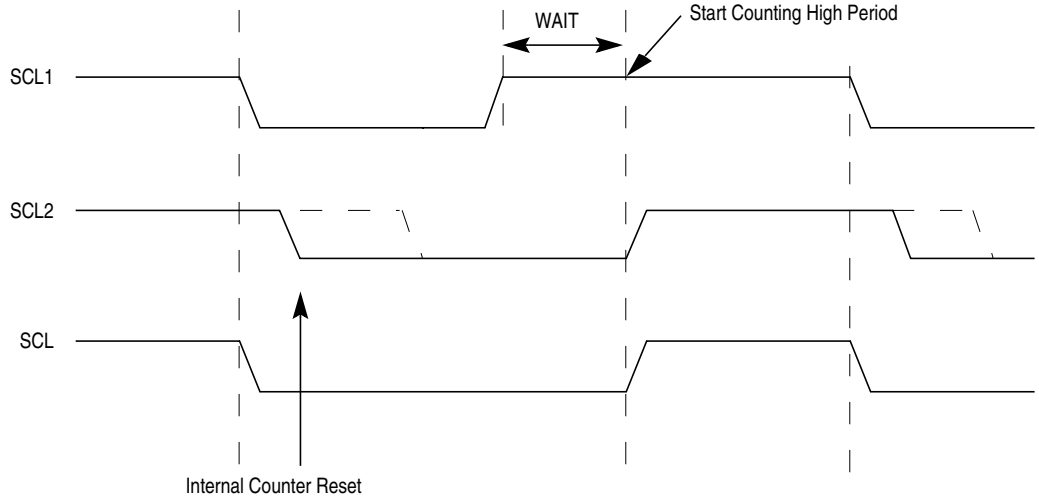



**Figure 14-12. IIC-Bus Clock Synchronization**

#### 14.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

#### 14.4.1.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it.If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

#### 14.4.1.10 Ten-bit Address

A ten-bit address is indicated if the first 5 bits of the first address byte are 0x11110. The following rules apply to the first address byte.

| SLAVE ADDRESS | R/W BIT | DESCRIPTION |
|---|---|---|
| 0000000 | 0 | General call address |
| 0000010 | x | Reserved for different bus format |
| 0000011 | x | Reserved for future purposes |
| 11111XX | x | Reserved for future purposes |
| 11110XX | x | 10-bit slave addressing |

**Figure 14-13. Definition of bits in the first byte.**

The address type is identified by ADTYPE. When ADTYPE is 0, 7-bit address is applied. Reversely, the address is 10-bit address.Generally, there are two cases of 10-bit address.See the Figure 14-14 and Figure 14-15.

| S | Slave Add1st 7bits 11110+ADR10+ADR9 | R/W 0 | A1 | Slave Add 2nd byte ADR[8:1] | A2 | Data | A3 |
|---|---|---|---|---|---|---|---|

**Figure 14-14. A master-transmitter addresses a slave-receiver with a 10-bit address**

| S | Slave Add1st 7bits 11110+ADR10+ADR9 | R/W 0 | A1 | Slave Add 2nd byte ADR[8:1] | A2 | Sr | Slave Add 1st 7bits 11110+ADR10+ADR9 | R/W 1 | A3 | Data | A4 |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Figure 14-15. A master-receiver addresses a slave-transmitter with a 10-bit address.**

In the Figure 14-15, the first two bytes are the similar to Figure 14-14. After the repeated START(Sr),the first slave address is transmitted again, but the R/W is 1, meaning that the slave is acted as a transmitter.

#### 14.4.1.11 General Call Address

To broadcast using a general call, a device must first generate the general call address($00), then after receiving acknowledge, it must transmit data.

In communication, as a slave device, provided the GCEN is asserted, a device acknowledges the broadcast and receives data until the GCEN is disabled or the master device releases the bus or generates a new transfer. In the broadcast, slaves always act as receivers. In general call, IAAS is also used to indicate the address match.

In order to distinguish whether the address match is the normal address match or the general call address match, IBDR should be read after the address byte has been received. If the data is $00, the match is general call address match. The meaning of the general call address is always specified in the first data byte and must be dealt with by S/W, the IIC hardware does not decode and process the first data byte.

When one byte transfer is done, the received data can be read from IBDR. The user can control the procedure by enabling or disabling GCEN.

### 14.4.2 Operation in Run Mode

This is the basic mode of operation.

### 14.4.3 Operation in Wait Mode

IIC operation in wait mode can be configured. Depending on the state of internal bits, the IIC can operate normally when the CPU is in wait mode or the IIC clock generation can be turned off and the IIC module enters a power conservation state during wait mode. In the later case, any transmission or reception in progress stops at wait mode entry.

### 14.4.4 Operation in Stop Mode

The IIC is inactive in stop mode for reduced power consumption. The STOP instruction does not affect IIC register states.

## 14.5 Resets

The reset state of each individual bit is listed in Section 14.3, “Memory Map and Register Definition,” which details the registers and their bit-fields.

## 14.6 Interrupts

IICV3 uses only one interrupt vector.

**Table 14-11. Interrupt Summary**

| Interrupt | Offset | Vector | Priority | Source | Description |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| IIC Interrupt | — | — | — | IBAL, TCF, IAAS bits in IBSR register | When either of IBAL, TCF or IAAS bits is set may cause an interrupt based on arbitration lost, transfer complete or address detect conditions |

Internally there are three types of interrupts in IIC. The interrupt service routine can determine the interrupt type by reading the status register.

IIC Interrupt can be generated on

1. Arbitration lost condition (IBAL bit set)
2. Byte transfer condition (TCF bit set)
3. Address detect condition (IAAS bit set)

The IIC interrupt is enabled by the IBIE bit in the IIC control register. It must be cleared by writing 0 to the IBF bit in the interrupt service routine.

## 14.7 Application Information

### 14.7.1 IIC Programming Examples

#### 14.7.1.1 Initialization Sequence

Reset will put the IIC bus control register to its default status. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the frequency divider register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the ADTYPE of IBCR2 to define the address length, 7 bits or 10 bits.
3. Update the IIC bus address register (IBAD) to define its slave address. If 10-bit address is applied IBCR2 should be updated to define the rest bits of address.
4. Set the IBEN bit of the IIC bus control register (IBCR) to enable the IIC interface system.
5. Modify the bits of the IIC bus control register (IBCR) to select master/slave mode, transmit/receive mode and interrupt enable or not.
6. If supported general call, the GCEN in IBCR2 should be asserted.

#### 14.7.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the IIC bus busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system

clock and the SCL period it may be necessary to wait until the IIC is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of a program which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
CHFLAG      BRSET   IBSR,#$20,*     ;WAIT FOR IBB FLAG TO CLEAR
TXSTART     BSET    IBCR,#$30       ;SET TRANSMIT AND MASTER MODE;i.e. GENERATE START CONDITION
            MOVB    CALLING,IBDR    ;TRANSMIT THE CALLING ADDRESS, D0=R/W
IBFREE      BRCLR   IBSR,#$20,*     ;WAIT FOR IBB FLAG TO SET
```

### 14.7.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The IIC bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. Software must clear the IBIF bit in the interrupt routine first. The TCF bit will be cleared by reading from the IIC bus data I/O register (IBDR) in receive mode or writing to IBDR in transmit mode.

Software may service the IIC I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit because their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly.For slave mode data cycles (IAAS=0) the SRW bit is not valid, the Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a 'master transmitter' in the interrupt routine.

```
ISR         BCLR    IBSR,#$02           ;CLEAR THE IBIF FLAG
            BRCLR   IBCR,#$20,SLAVE     ;BRANCH IF IN SLAVE MODE
            BRCLR   IBCR,#$10,RECEIVE   ;BRANCH IF IN RECEIVE MODE
            BRSET   IBSR,#$01,END       ;IF NO ACK, END OF TRANSMISSION
TRANSMIT    MOVB    DATABUF,IBDR        ;TRANSMIT NEXT BYTE OF DATA
```

### 14.7.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
MASTX   TST   TXCNT          ;GET VALUE FROM THE TRANSMITING COUNTER
        BEQ   END            ;END IF NO MORE DATA
        BRSET IBSR,#$01,END  ;END IF NO ACK
        MOVB  DATABUF,IBDR   ;TRANSMIT NEXT BYTE OF DATA
        DEC   TXCNT          ;DECREASE THE TXCNT
        BRA   EMASTX         ;EXIT
END     BCLR  IBCR,#$20      ;GENERATE A STOP CONDITION
EMASTX  RTI                  ;RETURN FROM INTERRUPT
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

```
MASR    DEC   RXCNT          ;DECREASE THE RXCNT
        BEQ   ENMASR         ;LAST BYTE TO BE READ
        MOVB  RXCNT,D1       ;CHECK SECOND LAST BYTE
        DEC   D1             ;TO BE READ
        BNE   NXMAR          ;NOT LAST OR SECOND LAST
LAMAR   BSET  IBCR,#$08      ;SECOND LAST, DISABLE ACK
                             ;TRANSMITTING
        BRA   NXMAR
ENMASR  BCLR  IBCR,#$20      ;LAST ONE, GENERATE 'STOP' SIGNAL
NXMAR   MOVB  IBDR,RXBUF     ;READ DATA AND STORE
        RTI
```

### 14.7.1.5 Generation of Repeated START

At the end of data transfer, if the master continues to want to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
RESTART BSET  IBCR,#$04      ;ANOTHER START (RESTART)
        MOVB  CALLING,IBDR   ;TRANSMIT THE CALLING ADDRESS;D0=R/W
```

### 14.7.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

#### 14.7.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices which lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL continues to be generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware will inhibit the transmission; switch the MS/SL bit from 1 to 0 without generating STOP condition; generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

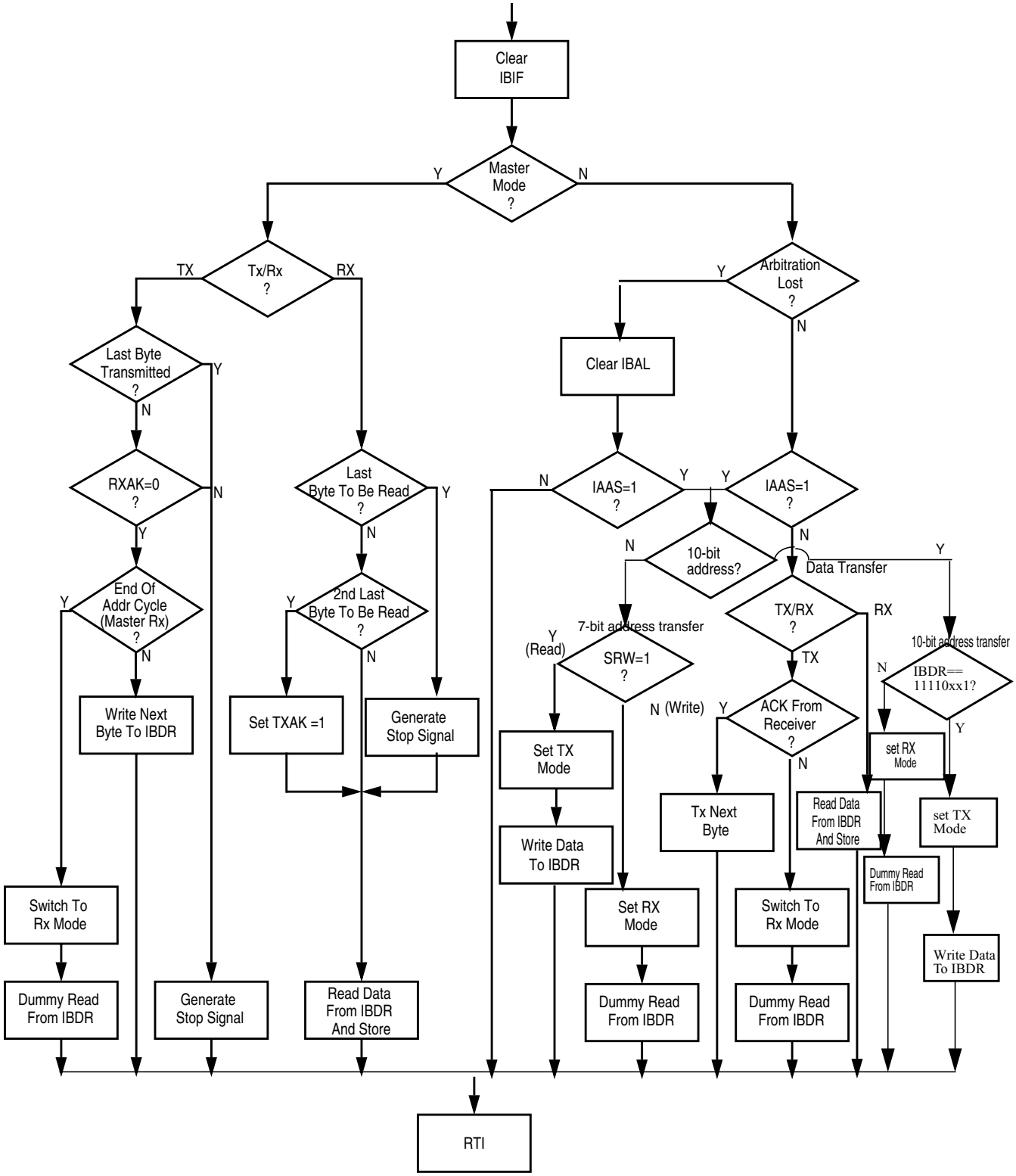



**Figure 14-16. Flow-Chart of Typical IIC Interrupt Routine**

Caution:When IIC is configured as 10-bit address,the point of the data array in interrupt routine must be reset after it’s addressed.

# Chapter 15
# Liquid Crystal Display (LCD40F4BV3) Block Description

## Revision History

**Table 15-1. LCD40F4BV3 Revision History**

| Version Number | Revision Date | Effective Date | Author | Description of Changes |
|---|---|---|---|---|
| 01.00 | 26-Jul-00 | | | initial LCD module spec |
| 01.08 | 27-Mar-08 | | | New specification for 9S12HY family based on 9S12H family specification |
| 01.09 | 25-Apr-08 | | | Update for 9S12HY defining last registers as unimplemented |
| 02.01 | 29-Jul-09 | | | add pseudo stop feature |
| 03.01 | 07-Jun-11 | | | update lcd clock prescaler on page 15-570 and table on page 15-575Change IRC clock to RTC clock<br>Add 15.4.5, “LCD Clock Inputs & Reset Behavior |
| 03.02 | 20-Jun-11 | | | update on page 15-571,LCDRSTP will control both pseudo stop and full stop<br>update on page 15-575,LCDRAM should be reset zero |
| 03.03 | 21-Jun-11 | | | add Section Figure 15-2., “LCD40F4BV3 Register Summary |
| 03.04 | 21-Oct-11 | | | Update Section Figure 15-2., “LCD40F4BV3 Register Summary,change name to LCDFPENR0-4 |
| 03.05 | 05-Sep-12 | | | rewording of Section 15.3.2.4, “LCD RAM (LCDRAM) |

## 15.1 Introduction

The LCD40F4BV3 driver module has 40 frontplane drivers and 4 backplane drivers so that a maximum of 160 LCD segments are controllable. Each segment is controlled by a corresponding bit in the LCD RAM. Four multiplex modes (1/1, 1/2, 1/3, 1/4 duty), and three bias (1/1, 1/2, 1/3) methods are available. The $V_0$ voltage is the lowest level of the output waveform and $V_3$ becomes the highest level. All frontplane and backplane pins can be multiplexed with other port functions.

The LCD40F4BV3 driver system consists of five major sub-modules:

- Timing and Control – consists of registers and control logic for frame clock generation, bias voltage level select, frame duty select, backplane select, and frontplane select/enable to produce the required frame frequency and voltage waveforms.

- LCD RAM – contains the data to be displayed on the LCD. Data can be read from or written to the display RAM at any time.
- Frontplane Drivers – consists of 40 frontplane drivers.
- Backplane Drivers – consists of 4 backplane drivers.
- Voltage Generator – Based on voltage applied to VLCD, it generates the voltage levels for the timing and control logic to produce the frontplane and backplane waveforms.

### 15.1.1 Features

The LCD40F4BV3 includes these distinctive features:

- Supports five LCD operation modes
- 40 frontplane drivers
- 4 backplane drivers
  - — Each frontplane has an enable bit respectively
- Programmable frame clock generator
- Programmable bias voltage level selector
- On-chip generation of 4 different output voltage levels

### 15.1.2 Modes of Operation

The LCD40F4BV3 module supports five operation modes with different numbers of backplanes and different biasing levels. During wait mode the LCD operation can be suspended under software control. Depending on the state of internal bits, the LCD can operate normally or the LCD clock generation can be turned off and the LCD40F4BV3 module enters a power conservation state.

This is a high level description only, detailed descriptions of operating modes are contained in Section 15.4.2, “Operation in Wait Mode”, and Section 15.4.3, “Operation in Stop Mode”.

### 15.1.3 Block Diagram

Figure 15-1 is a block diagram of the LCD40F4BV3 module.

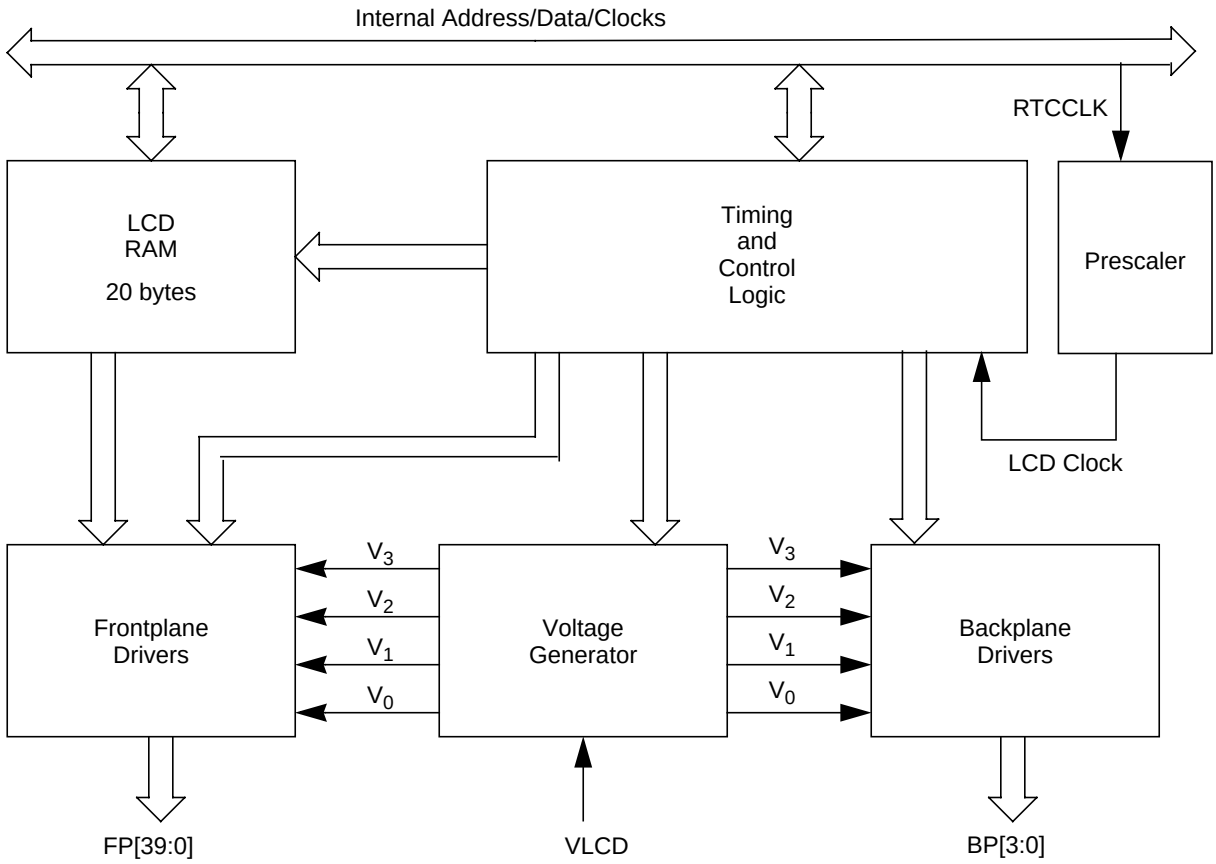



**Figure 15-1. LCD40F4BV3 Block Diagram**

## 15.2 External Signal Description

The LCD40F4BV3 module has a total of 45 external pins.

**Table 15-2. Signal Properties**

| Name | Port | Function | Reset State |
|---|---|---|---|
| 4 backplane waveforms | BP[3:0] | Backplane waveform signals that connect directly to the pads | High impedance |
| 40 frontplane waveforms | FP[39:0] | Frontplane waveform signals that connect directly to the pads | High impedance |
| LCD voltage | VLCD | LCD supply voltage | — |

### 15.2.1 BP[3:0] — Analog Backplane Pins

This output signal vector represents the analog backplane waveforms of the LCD40F4BV3 module and is connected directly to the corresponding pads.

### 15.2.2 FP[39:0] — Analog Frontplane Pins

This output signal vector represents the analog frontplane waveforms of the LCD40F4BV3 module and is connected directly to the corresponding pads.

### 15.2.3 VLCD — LCD Supply Voltage Pin

Positive supply voltage for the LCD waveform generation.

## 15.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

### 15.3.1 Module Memory Map

The memory map for the LCD40F4BV3 module is given in Table 15-3. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the LCD40F4BV3 module and the address offset for each register.

**Table 15-3. LCD40F4BV3 Memory Map**

| Address Offset | Use | Access |
|---|---|---|
| 0x0000 | LCD Control Register 0 (LCDCR0) | Read/Write |
| 0x0001 | LCD Control Register 1 (LCDCR1) | Read/Write |
| 0x0002 | LCD Frontplane Enable Register 0 (LCDFPENR0) | Read/Write |
| 0x0003 | LCD Frontplane Enable Register 1 (LCDFPENR1) | Read/Write |
| 0x0004 | LCD Frontplane Enable Register 2 (LCDFPENR2) | Read/Write |
| 0x0005 | LCD Frontplane Enable Register 3 (LCDFPENR3) | Read/Write |
| 0x0006 | LCD Frontplane Enable Register 4 (LCDFPENR4) | Read/Write |
| 0x0007 | Unimplemented | |
| 0x0008 | LCDRAM (Location 0) | Read/Write |
| 0x0009 | LCDRAM (Location 1) | Read/Write |
| 0x000A | LCDRAM (Location 2) | Read/Write |
| 0x000B | LCDRAM (Location 3) | Read/Write |
| 0x000C | LCDRAM (Location 4) | Read/Write |
| 0x000D | LCDRAM (Location 5) | Read/Write |
| 0x000E | LCDRAM (Location 6) | Read/Write |
| 0x000F | LCDRAM (Location 7) | Read/Write |
| 0x0010 | LCDRAM (Location 8) | Read/Write |
| 0x0011 | LCDRAM (Location 9) | Read/Write |
| 0x0012 | LCDRAM (Location 10) | Read/Write |
| 0x0013 | LCDRAM (Location 11) | Read/Write |
| 0x0014 | LCDRAM (Location 12) | Read/Write |
| 0x0015 | LCDRAM (Location 13) | Read/Write |
| 0x0016 | LCDRAM (Location 14) | Read/Write |
| 0x0017 | LCDRAM (Location 15) | Read/Write |
| 0x0018 | LCDRAM (Location 16) | Read/Write |
| 0x0019 | LCDRAM (Location 17) | Read/Write |
| 0x001A | LCDRAM (Location 18) | Read/Write |
| 0x001B | LCDRAM (Location 19) | Read/Write |
| 0x001C-0x001F | Unimplemented | |

## 15.3.2 Register Descriptions

This section consists of register descriptions. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | LCDCR0 | R<br>W | LCDEN | 0 | 0 | LCLK1 | LCLK0 | BIAS | DUTY1 | DUTY0 |
| 0x0001 | LCDCR1 | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | LCDSWAI | LCDRSTP |
| 0x0002 | LCDFPENR0 | R<br>W | FP7EN | FP6EN | FP5EN | FP4EN | FP3EN | FP2EN | FP1EN | FP0EN |
| 0x0003 | LCDFPENR1 | R<br>W | FP15EN | FP14EN | FP13EN | FP12EN | FP11EN | FP10EN | FP9EN | FP8EN |
| 0x0004 | LCDFPENR2 | R<br>W | FP23EN | FP22EN | FP21EN | FP20EN | FP19EN | FP18EN | FP17EN | FP16EN |
| 0x0005 | LCDFPENR3 | R<br>W | FP31EN | FP30EN | FP29EN | FP28EN | FP27EN | FP26EN | FP25EN | FP24EN |
| 0x0006 | LCDFPENR4 | R<br>W | FP39EN | FP38EN | FP37EN | FP36EN | FP35EN | FP34EN | FP33EN | FP32EN |
| 0x0007 | Reserved | R<br>W | | | | | | | | |
| 0x0008 | LCDRAM0 | R<br>W | FP1BP3 | FP1BP2 | FP1BP1 | FP1BP0 | FP0BP3 | FP0BP2 | FP0BP1 | FP0BP0 |
| 0x0009 | LCDRAM1 | R<br>W | FP3BP3 | FP3BP2 | FP3BP1 | FP3BP0 | FP2BP3 | FP2BP2 | FP2BP1 | FP2BP0 |
| 0x000A | LCDRAM2 | R<br>W | FP5BP3 | FP5BP2 | FP5BP1 | FP5BP0 | FP4BP3 | FP4BP2 | FP4BP1 | FP4BP0 |
| 0x000B | LCDRAM3 | R<br>W | FP7BP3 | FP7BP2 | FP7BP1 | FP7BP0 | FP6BP3 | FP6BP2 | FP6BP1 | FP6BP0 |
| 0x000C | LCDRAM4 | R<br>W | FP9BP3 | FP9BP2 | FP9BP1 | FP9BP0 | FP8BP3 | FP8BP2 | FP8BP1 | FP8BP0 |
| 0x000D | LCDRAM5 | R<br>W | FP11BP3 | FP11BP2 | FP11BP1 | FP11BP0 | FP10BP3 | FP10BP2 | FP10BP1 | FP10BP0 |
| 0x000E | LCDRAM6 | R<br>W | FP13BP3 | FP13BP2 | FP13BP1 | FP13BP0 | FP12BP3 | FP12BP2 | FP12BP1 | FP12BP0 |
| 0x000F | LCDRAM7 | R<br>W | FP15BP3 | FP15BP2 | FP15BP1 | FP15BP0 | FP14BP3 | FP14BP2 | FP14BP1 | FP14BP0 |

**Figure 15-2. LCD40F4BV3 Register Summary**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0010 | LCDRAM8 | R<br>W | FP17BP3 | FP17BP2 | FP17BP1 | FP17BP0 | FP16BP3 | FP16BP2 | FP16BP1 | FP16BP0 |
| 0x0011 | LCDRAM9 | R<br>W | FP19BP3 | FP19BP2 | FP19BP1 | FP19BP0 | FP18BP3 | FP18BP2 | FP18BP1 | FP18BP0 |
| 0x0012 | LCDRAM10 | R<br>W | FP21BP3 | FP21BP2 | FP21BP1 | FP21BP0 | FP20BP3 | FP20BP2 | FP20BP1 | FP20BP0 |
| 0x0013 | LCDRAM11 | R<br>W | FP23BP3 | FP23BP2 | FP23BP1 | FP23BP0 | FP22BP3 | FP22BP2 | FP22BP1 | FP22BP0 |
| 0x0014 | LCDRAM12 | R<br>W | FP25BP3 | FP25BP2 | FP25BP1 | FP25BP0 | FP24BP3 | FP24BP2 | FP24BP1 | FP24BP0 |
| 0x0015 | LCDRAM13 | R<br>W | FP27BP3 | FP27BP2 | FP27BP1 | FP27BP0 | FP26BP3 | FP26BP2 | FP26BP1 | FP26BP0 |
| 0x0016 | LCDRAM14 | R<br>W | FP29BP3 | FP29BP2 | FP29BP1 | FP29BP0 | FP28BP3 | FP28BP2 | FP28BP1 | FP28BP0 |
| 0x0017 | LCDRAM15 | R<br>W | FP31BP3 | FP31BP2 | FP31BP1 | FP31BP0 | FP30BP3 | FP30BP2 | FP30BP1 | FP30BP0 |
| 0x0018 | LCDRAM16 | R<br>W | FP33BP3 | FP33BP2 | FP33BP1 | FP33BP0 | FP32BP3 | FP32BP2 | FP32BP1 | FP32BP0 |
| 0x0019 | LCDRAM17 | R<br>W | FP35BP3 | FP35BP2 | FP35BP1 | FP35BP0 | FP34BP3 | FP34BP2 | FP34BP1 | FP34BP0 |
| 0x001A | LCDRAM18 | R<br>W | FP37BP3 | FP37BP2 | FP37BP1 | FP37BP0 | FP36BP3 | FP36BP2 | FP36BP1 | FP36BP0 |
| 0x001B | LCDRAM19 | R<br>W | FP39BP3 | FP39BP2 | FP39BP1 | FP39BP0 | FP38BP3 | FP38BP2 | FP38BP1 | FP38BP0 |
| 0x001C-<br>0x001F | Reserved | R<br>W | | | | | | | | |

**Figure 15-2. LCD40F4BV3 Register Summary**

### 15.3.2.1 LCD Control Register 0 (LCDCR0)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LCDEN | 0 | 0 | LCLK1 | LCLK0 | BIAS | DUTY1 | DUTY0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-3. LCD Control Register 0 (LCDCR0)**

Read: anytime

Write: LCDEN anytime. To avoid segment flicker the clock prescaler bits, the bias select bit and the duty select bits must not be changed when the LCD is enabled.

**Table 15-4. LCDCR0 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LCDEN | **LCD40F4BV3 Driver System Enable** — The LCDEN bit starts the LCD waveform generator.<br>0 All frontplane and backplane pins are disabled. In addition, the LCD40F4BV3 system is disabled and all LCD waveform generation clocks are stopped.<br>1 LCD driver system is enabled. All FP[39:0] pins with FP[39:0]EN set, will output an LCD driver waveform The BP[3:0] pins will output an LCD40F4BV3 driver waveform based on the settings of DUTY0 and DUTY1. |
| 4:3<br>LCLK[1:0] | **LCD Clock Prescaler** — The LCD clock prescaler bits determine the RTCCLK divider value to produce the LCD clock frequency. For detailed description of the correlation between LCD clock prescaler bits and the divider value please refer to Table 15-8. |
| 2<br>BIAS | **BIAS Voltage Level Select** — This bit selects the bias voltage levels during various LCD operating modes, as shown in Table 15-9. |
| 1:0<br>DUTY[1:0] | **LCD Duty Select** — The DUTY1 and DUTY0 bits select the duty (multiplex mode) of the LCD40F4BV3 driver system, as shown in Table 15-9. |

### 15.3.2.2 LCD Control Register 1 (LCDCR1)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | LCDSWAI | LCDRSTP |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Unimplemented or Reserved

**Figure 15-4. LCD Control Register 1 (LCDCR1)**

Read: anytime

Write: anytime

**Table 15-5. LCDCR1 Field Descriptions**

| Field | Description |
|---|---|
| 1<br>LCDSWAI | **LCD Stop in Wait Mode** — This bit controls the LCD operation while in wait mode.<br>0 LCD operates normally in wait mode.<br>1 Stop LCD40F4BV3 driver system when in wait mode. |
| 0<br>LCDRSTP | LCD Run in Stop Mode This bit controls the LCD operation while in pseudo stop mode and full stop mode.<br>0 Stop LCD32F4B driver system when in stop mode(pseudo stop or full stop).<br>1 LCD operates normally in stop mode. If lcd clock come from 32k osc clock which is not stoped during full stop mode, LCDRSTP=1 will control LCD run in pseudo stop or full stop mode. If lcd clock come from main osc clock, LCDRSTP=1 will control LCD run in pseudo stop mode. |

### 15.3.2.3 LCD Frontplane Enable Register 0–3 (LCDFPENR0–LCDFPENR4)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | FP7EN | FP6EN | FP5EN | FP4EN | FP3EN | FP2EN | FP1EN | FP0EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-5. LCD Frontplane Enable Register 0 (LCDFPENR0)**

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | FP15EN | FP14EN | FP13EN | FP12EN | FP11EN | FP10EN | FP9EN | FP8EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-6. LCD Frontplane Enable Register 1 (LCDFPENR1)**

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | FP23EN | FP22EN | FP21EN | FP20EN | FP19EN | FP18EN | FP17EN | FP16EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-7. LCD Frontplane Enable Register 2 (LCDFPENR2)**

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | FP31EN | FP30EN | FP29EN | FP28EN | FP27EN | FP26EN | FP25EN | FP24EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-8. LCD Frontplane Enable Register 3 (LCDFPENR3)**

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | FP39EN | FP38EN | FP37EN | FP36EN | FP35EN | FP34EN | FP33EN | FP32EN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-9. LCD Frontplane Enable Register 4 (LCDFPENR4)**

These bits enable the frontplane output waveform on the corresponding frontplane pin when LCDEN = 1.

Read: anytime

Write: anytime

**Table 15-6. LCDFPENR0–LCDFPENR4 Field Descriptions**

| Field | Description |
|---|---|
| 39:0<br>FP[39:0]EN | **Frontplane Output Enable** — The FP[39:0]EN bit enables the frontplane driver outputs. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set FP[39:0]EN bits before LCDEN is set.<br>0 Frontplane driver output disabled on FP[39:0].<br>1 Frontplane driver output enabled on FP[39:0]. |

### 15.3.2.4 LCD RAM (LCDRAM)

The LCD RAM consists of 20 bytes. After reset the LCD RAM contents will be zero, as indicated by Figure 15-10.

| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0008 LCDRAM | R<br>W | FP1BP3 | FP1BP2 | FP1BP1 | FP1BP0 | FP0BP3 | FP0BP2 | FP0BP1 | FP0BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0009 LCDRAM | R<br>W | FP3BP3 | FP3BP2 | FP3BP1 | FP3BP0 | FP2BP3 | FP2BP2 | FP2BP1 | FP2BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000A LCDRAM | R<br>W | FP5BP3 | FP5BP2 | FP5BP1 | FP5BP0 | FP4BP3 | FP4BP2 | FP4BP1 | FP4BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000B LCDRAM | R<br>W | FP7BP3 | FP7BP2 | FP7BP1 | FP7BP0 | FP6BP3 | FP6BP2 | FP6BP1 | FP6BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000C LCDRAM | R<br>W | FP9BP3 | FP9BP2 | FP9BP1 | FP9BP0 | FP8BP3 | FP8BP2 | FP8BP1 | FP8BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000D LCDRAM | R<br>W | FP11BP3 | FP11BP2 | FP11BP1 | FP11BP0 | FP10BP3 | FP10BP2 | FP10BP1 | FP10BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000E LCDRAM | R<br>W | FP13BP3 | FP13BP2 | FP13BP1 | FP13BP0 | FP12BP3 | FP12BP2 | FP12BP1 | FP12BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x000F LCDRAM | R<br>W | FP15BP3 | FP15BP2 | FP15BP1 | FP15BP0 | FP14BP3 | FP14BP2 | FP14BP1 | FP14BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0010 LCDRAM | R<br>W | FP17BP3 | FP17BP2 | FP17BP1 | FP17BP0 | FP16BP3 | FP16BP2 | FP16BP1 | FP16BP0 |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

I = Value is indeterminate

**Figure 15-10. LCD RAM (LCDRAM)**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x0011 LCDRAM | R | FP19BP3 | FP19BP2 | FP19BP1 | FP19BP0 | FP18BP3 | FP18BP2 | FP18BP1 | FP18BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0012 LCDRAM | R | FP21BP3 | FP21BP2 | FP21BP1 | FP21BP0 | FP20BP3 | FP20BP2 | FP20BP1 | FP20BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0013 LCDRAM | R | FP23BP3 | FP23BP2 | FP23BP1 | FP23BP0 | FP22BP3 | FP22BP2 | FP22BP1 | FP22BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0014 LCDRAM | R | FP25BP3 | FP25BP2 | FP25BP1 | FP25BP0 | FP24BP3 | FP24BP2 | FP24BP1 | FP24BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0015 LCDRAM | R | FP27BP3 | FP27BP2 | FP27BP1 | FP27BP0 | FP26BP3 | FP26BP2 | FP26BP1 | FP26BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0016 LCDRAM | R | FP29BP3 | FP29BP2 | FP29BP1 | FP29BP0 | FP28BP3 | FP28BP2 | FP28BP1 | FP28BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0017 LCDRAM | R | FP31BP3 | FP31BP2 | FP31BP1 | FP31BP0 | FP30BP3 | FP30BP2 | FP30BP1 | FP30BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0018 LCDRAM | R | FP33BP3 | FP33BP2 | FP33BP1 | FP33BP0 | FP32BP3 | FP32BP2 | FP32BP1 | FP32BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0019 LCDRAM | R | FP35BP3 | FP35BP2 | FP35BP1 | FP35BP0 | FP34BP3 | FP34BP2 | FP34BP1 | FP34BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x001A LCDRAM | R | FP37BP3 | FP37BP2 | FP37BP1 | FP37BP0 | FP36BP3 | FP36BP2 | FP36BP1 | FP36BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x001B LCDRAM | R | FP39BP3 | FP39BP2 | FP39BP1 | FP39BP0 | FP38BP3 | FP38BP2 | FP38BP1 | FP38BP0 |
| | W | | | | | | | | |
| | Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

I = Value is indeterminate

**Figure 15-10. LCD RAM (LCDRAM) (continued)**

Read: anytime

Write: anytime

**Table 15-7. LCD RAM Field Descriptions**

| Field | Description |
|---|---|
| 39:0<br>3:0<br>FP[39:0]<br>BP[3:0] | **LCD Segment ON** — The FP[39:0]BP[3:0] bit displays (turns on) the LCD segment connected between FP[39:0] and BP[3:0].<br>0 LCD segment OFF<br>1 LCD segment ON |

## 15.4 Functional Description

This section provides a complete functional description of the LCD40F4BV3 block, detailing the operation of the design from the end user perspective in a number of subsections.

### 15.4.1 LCD Driver Description

#### 15.4.1.1 Frontplane, Backplane, and LCD System During Reset

During a reset the following conditions exist:

- The LCD40F4BV3 system is configured in the default mode, 1/4 duty and 1/3 bias, that means all backplanes are used.
- All frontplane enable bits, FP[39:0]EN are cleared and the ON/OFF control for the display, the LCDEN bit is cleared, thereby forcing all frontplane and backplane driver outputs to the high impedance state. The MCU pin state during reset is defined by the port integration module (PIM).

#### 15.4.1.2 LCD Clock and Frame Frequency

The frequency of the source clock (RTCCLK) and divider determine the LCD clock frequency. The divider is set by the LCD clock prescaler bits, LCLK[1:0], in the LCD control register 0 (LCDCR0). Table 15-8 shows the LCD clock and frame frequency for some multiplexed mode at RTCCLK = 32768Hz, 32000Hz, and 64000Hz.

**Table 15-8. LCD Clock and Frame Frequency**

| Source clock Frequency in Hz | LCD Clock Prescaler | | Divider | LCD Clock Frequency [Hz] | Frame Frequency [Hz] | | | |
|---|---|---|---|---|---|---|---|---|
| | LCLK1 | LCLK0 | | | 1/1 Duty | 1/2 Duty | 1/3 Duty | 1/4 Duty |
| RTCCLK = 32768 | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 64<br>128<br>256<br>512 | 512<br>256<br>128<br>64 | 512<br>256<br>128<br>64 | 256<br>128<br>64<br>32 | 171<br>85<br>43<br>21 | 128<br>64<br>32<br>16 |
| RTCCLK = 32000 | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 64<br>128<br>256<br>512 | 500<br>250<br>125<br>63 | 500<br>250<br>125<br>63 | 250<br>125<br>63<br>31 | 167<br>83<br>42<br>21 | 125<br>63<br>31<br>16 |

**Table 15-8. LCD Clock and Frame Frequency**

| Source clock Frequency in Hz | LCD Clock Prescaler | | Divider | LCD Clock Frequency [Hz] | Frame Frequency [Hz] | | | |
|---|---|---|---|---|---|---|---|---|
| | LCLK1 | LCLK0 | | | 1/1 Duty | 1/2 Duty | 1/3 Duty | 1/4 Duty |
| RTCCLK = 64000 | 0 | 0 | 64 | 1000 | 1000 | 500 | 333 | 250 |
| | 0 | 1 | 128 | 500 | 500250 | 250 | 167 | 125 |
| | 1 | 0 | 256 | 250 | 125 | 125 | 83 | 63 |
| | 1 | 1 | 512 | 125 | | 63 | 42 | 31 |

For other combinations of RTCCLK and divider not shown in Table 15-8, the following formula may be used to calculate the LCD frame frequency for each multiplex mode:

$$\text{LCD Frame Frequency (Hz)} = \left[\frac{(\text{RTCCLK (Hz)})}{\text{Divider}}\right] \cdot \text{Duty}$$

The possible divider values are shown in Table 15-8.

### 15.4.1.3 LCD RAM

For a segment on the LCD to be displayed, data must be written to the LCD RAM which is shown in Section 15.3, “Memory Map and Register Definition”. The 160 bits in the LCD RAM correspond to the 160 segments that are driven by the frontplane and backplane drivers. Writing a 1 to a given location will result in the corresponding display segment being driven with a differential RMS voltage necessary to turn the segment ON when the LCDEN bit is set and the corresponding FP[39:0]EN bit is set. Writing a 0 to a given location will result in the corresponding display segment being driven with a differential RMS voltage necessary to turn the segment OFF. The LCD RAM is a dual port RAM that interfaces with the internal address and data buses of the MCU. It is possible to read from LCD RAM locations for scrolling purposes. When LCDEN = 0, the LCD RAM can be used as on-chip RAM. Writing or reading of the LCDEN bit does not change the contents of the LCD RAM. After a reset, the LCD RAM contents will be indeterminate.

### 15.4.1.4 LCD Driver System Enable and Frontplane Enable Sequencing

If LCDEN = 0 (LCD40F4BV3 driver system disabled) and the frontplane enable bit, FP[39:0]EN, is set, the frontplane driver waveform will not appear on the output until LCDEN is set. If LCDEN = 1 (LCD40F4BV3 driver system enabled), the frontplane driver waveform will appear on the output as soon as the corresponding frontplane enable bit, FP[39:0]EN, in the registers LCDFPENR0–LCDFPENR4 is set.

### 15.4.1.5 LCD Bias and Modes of Operation

The LCD40F4BV3 driver has five modes of operation:

- 1/1 duty (1 backplane), 1/1 bias (2 voltage levels)
- 1/2 duty (2 backplanes), 1/2 bias (3 voltage levels)
- 1/2 duty (2 backplanes), 1/3 bias (4 voltage levels)

- 1/3 duty (3 backplanes), 1/3 bias (4 voltage levels)
- 1/4 duty (4 backplanes), 1/3 bias (4 voltage levels)

The voltage levels required for the different operating modes are generated internally based on VLCD. Changing VLCD alters the differential RMS voltage across the segments in the ON and OFF states, thereby setting the display contrast.

The backplane waveforms are continuous and repetitive every frame. They are fixed within each operating mode and are not affected by the data in the LCD RAM.

The frontplane waveforms generated are dependent on the state (ON or OFF) of the LCD segments as defined in the LCD RAM. The LCD40F4BV3 driver hardware uses the data in the LCD RAM to construct the frontplane waveform to create a differential RMS voltage necessary to turn the segment ON or OFF.

The LCD duty is decided by the DUTY1 and DUTY0 bits in the LCD control register 0 (LCDCR0). The number of bias voltage levels is determined by the BIAS bit in LCDCR0. Table 15-9 summarizes the multiplex modes (duties) and the bias voltage levels that can be selected for each multiplex mode (duty). The backplane pins have their corresponding backplane waveform output BP[3:0] in high impedance state when in the OFF state as indicated in Table 15-9. In the OFF state the corresponding pins BP[3:0]can be used for other functionality, for example as general purpose I/O ports.

**Table 15-9. LCD Duty and Bias**

| Duty | LCDCR0 Register | | Backplanes | | | | Bias (BIAS = 0) | | | Bias (BIAS = 1) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DUTY1 | DUTY0 | BP3 | BP2 | BP1 | BP0 | 1/1 | 1/2 | 1/3 | 1/1 | 1/2 | 1/3 |
| 1/1 | 0 | 1 | OFF | OFF | OFF | BP0 | YES | NA | NA | YES | NA | NA |
| 1/2 | 1 | 0 | OFF | OFF | BP1 | BP0 | NA | YES | NA | NA | NA | YES |
| 1/3 | 1 | 1 | OFF | BP2 | BP1 | BP0 | NA | NA | YES | NA | NA | YES |
| 1/4 | 0 | 0 | BP3 | BP2 | BP1 | BP0 | NA | NA | YES | NA | NA | YES |

## 15.4.2 Operation in Wait Mode

The LCD40F4BV3 driver system operation during wait mode is controlled by the LCD stop in wait (LCDSWAI) bit in the LCD control register 1 (LCDCR1). If LCDSWAI is reset, the LCD40F4BV3 driver system continues to operate during wait mode. If LCDSWAI is set, the LCD40F4BV3 driver system is turned off during wait mode. In this case, the LCD waveform generation clocks are stopped and the LCD40F4BV3 drivers pull down to VSSX those frontplane and backplane pins that were enabled before entering wait mode. The contents of the LCD RAM and the LCD registers retain the values they had prior to entering wait mode.

## 15.4.3 Operation in Stop Mode

All LCD40F4BV3 driver system clocks are stopped, the LCD40F4BV3 driver system pulls down to VSSX those frontplane and backplane pins that were enabled before entering stop mode. Also, during stop mode, the contents of the LCD RAM and the LCD registers retain the values they had prior to entering stop mode. As a result, after exiting from stop mode, the LCD40F4BV3 driver system clocks will run (if LCDEN = 1) and the frontplane and backplane pins retain the functionality they had prior to entering stop mode.

### 15.4.4 LCD Waveform Examples

Figure 15-11 through Figure 15-15 show the timing examples of the LCD output waveforms for the available modes of operation.

### 15.4.4.1 1/1 Duty Multiplexed with 1/1 Bias Mode

Duty = 1/1:DUTY1 = 0, DUTY0 = 1

Bias = 1/1:BIAS = 0 or BIAS = 1

$V_0 = V_1 = VSSX$, $V_2 = V_3 = VLCD$

- BP1, BP2, and BP3 are not used, a maximum of 40 segments are displayed.

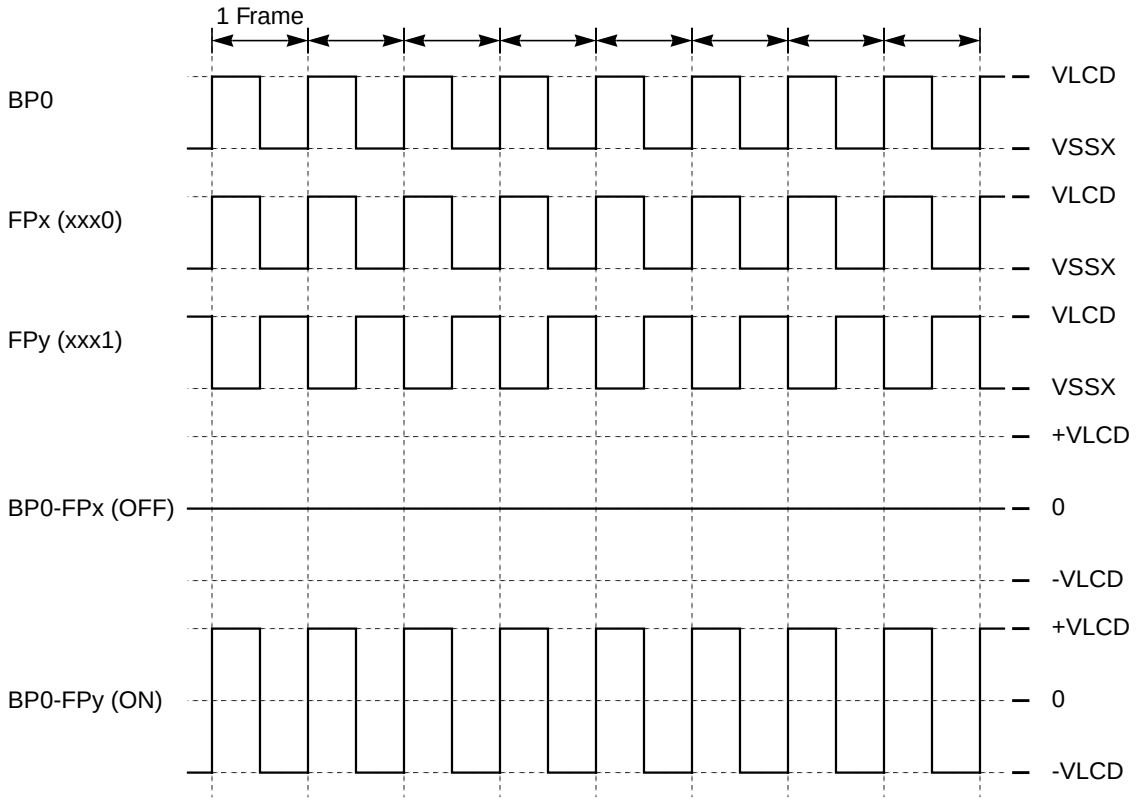



**Figure 15-11. 1/1 Duty and 1/1 Bias**

### 15.4.4.2 1/2 Duty Multiplexed with 1/2 Bias Mode

Duty = 1/2:DUTY1 = 1, DUTY0 = 0

Bias = 1/2:BIAS = 0

$V_0$ = VSSX, $V_1$ = $V_2$ = VLCD * 1/2, $V_3$ = VLCD

- BP2 and BP3 are not used, a maximum of 80 segments are displayed.

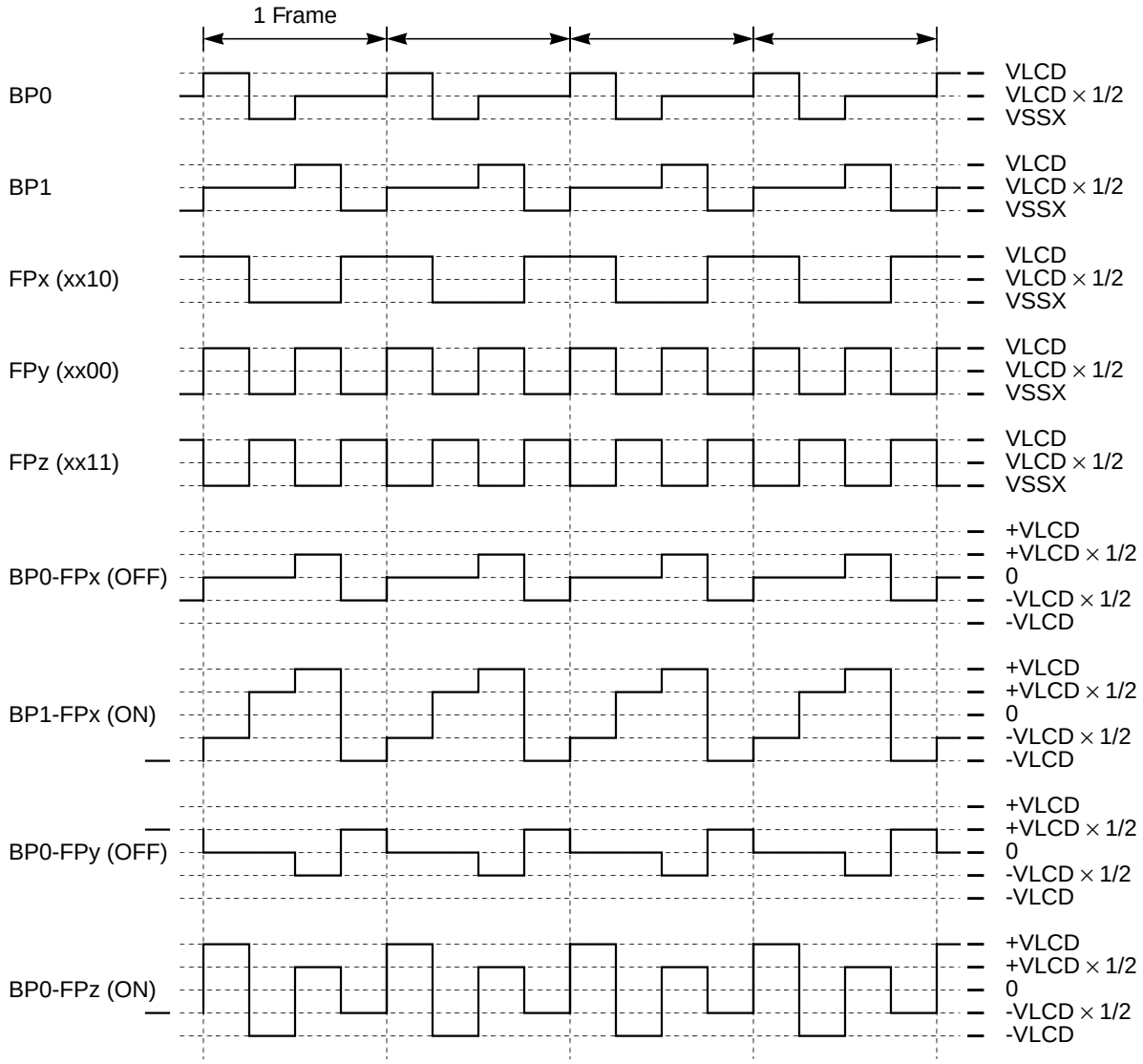



**Figure 15-12. 1/2 Duty and 1/2 Bias**

#### 15.4.4.3 1/2 Duty Multiplexed with 1/3 Bias Mode

Duty = 1/2:DUTY1 = 1, DUTY0 = 0

Bias = 1/3:BIAS = 1

$V_0$ = VSSX, $V_1$ = VLCD * 1/3, $V_2$ = VLCD * 2/3, $V_3$ = VLCD

- BP2 and BP3 are not used, a maximum of 80 segments are displayed.

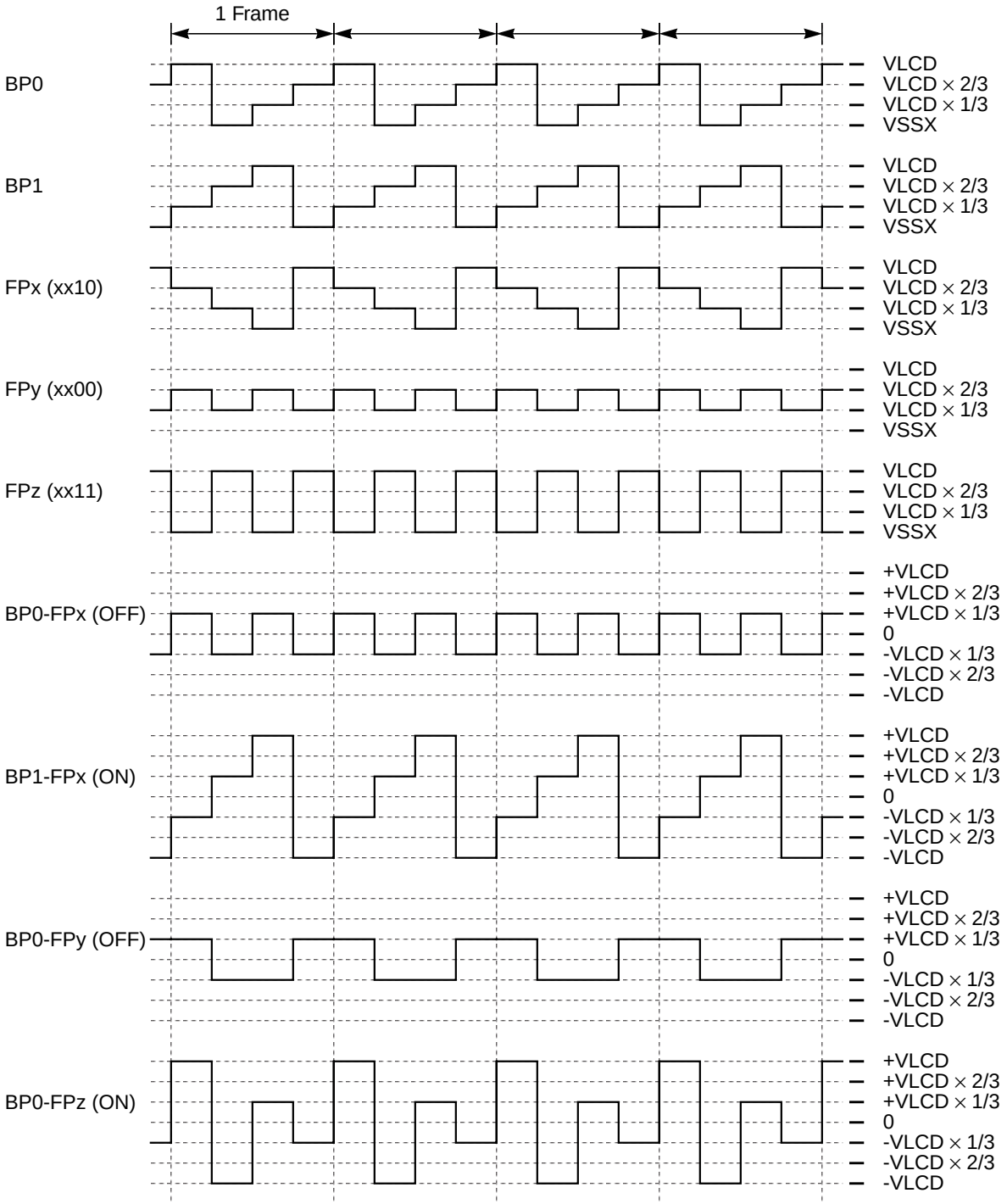



**Figure 15-13. 1/2 Duty and 1/3 Bias**

### 15.4.4.4 1/3 Duty Multiplexed with 1/3 Bias Mode

Duty = 1/3:DUTY1 = 1, DUTY0 = 1

Bias = 1/3:BIAS = 0 or BIAS = 1

$V_0$ = VSSX, $V_1$ = VLCD * 1/3, $V_2$ = VLCD * 2/3, $V_3$ = VLCD

- BP3 is not used, a maximum of 120 segments are displayed.

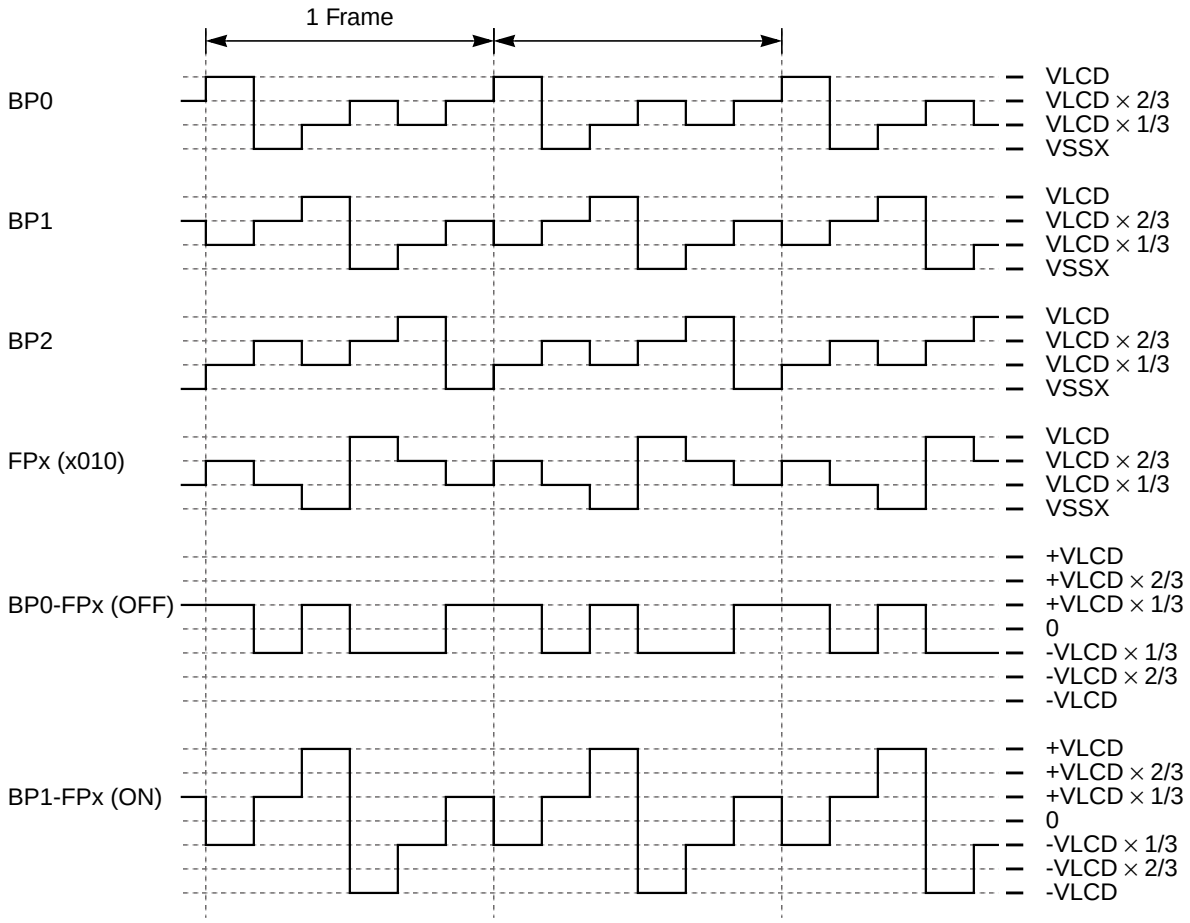



**Figure 15-14. 1/3 Duty and 1/3 Bias**

### 15.4.4.5 1/4 Duty Multiplexed with 1/3 Bias Mode

Duty = 1/4:DUTY1 = 0, DUTY0 = 0

Bias = 1/3:BIAS = 0 or BIAS = 1

$V_0$ = VSSX, $V_1$ = VLCD * 1/3, $V_2$ = VLCD * 2/3, $V_3$ = VLCD

- A maximum of 160 segments are displayed.

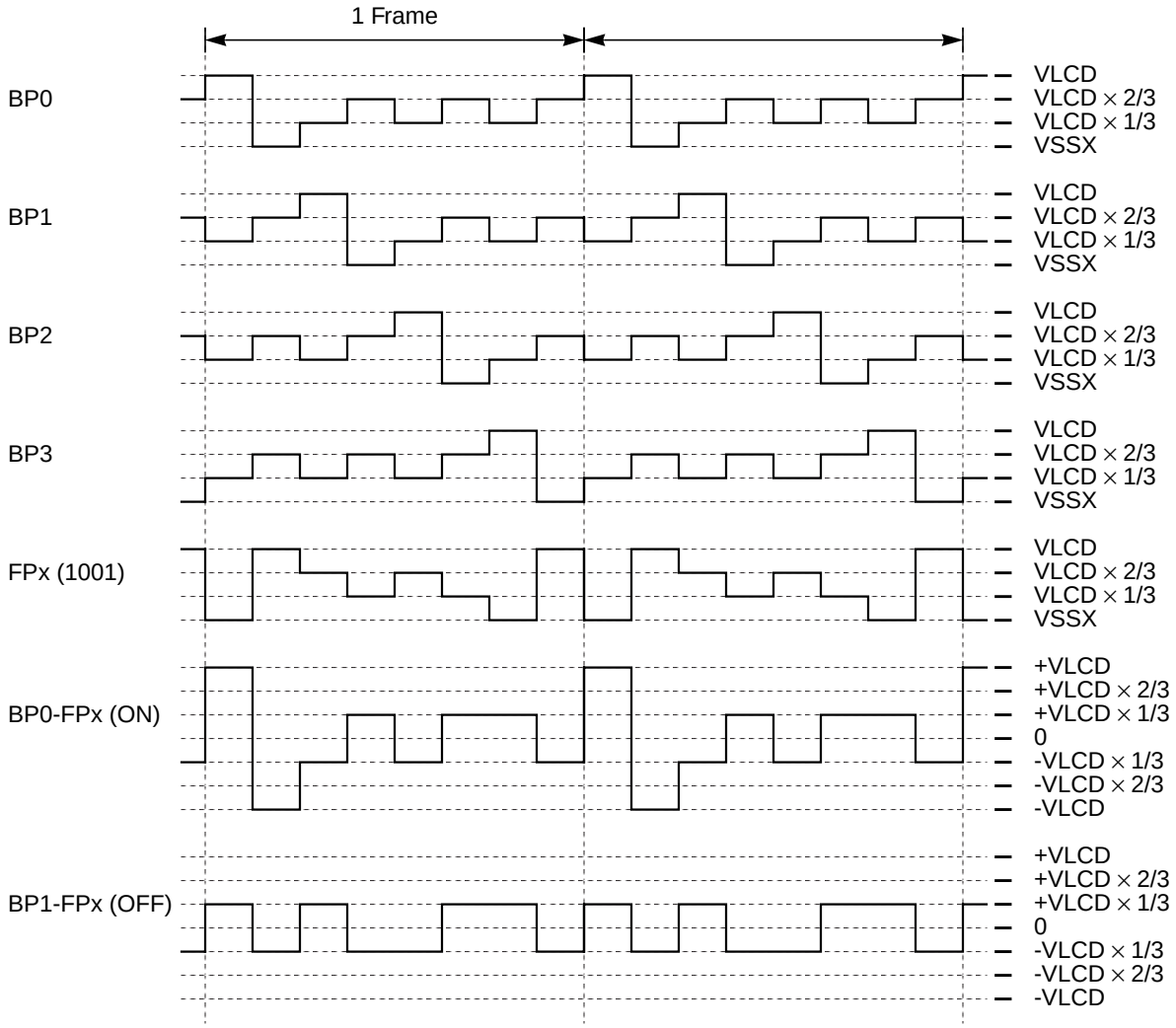



**Figure 15-15. 1/4 Duty and 1/3 Bias**

## 15.4.5 LCD Clock Inputs & Reset Behavior

The LCD clock input can from main OSC (OSCCLK) or 32 kHz OSC (OSCCLK_32K), refer to Reference Manual for detailed information. In case the clock source is from main OSC, after reset, the main OSC will be disabled and must be enabled by setting the OSCE in CPMU as soon as possible in order to have clock available for the LCD. If clock source is OSCCLK_32K, software must wait the OSC startup time before enabling the LCD. If 32K OSC is enabled, it will be always on until a power on reset happens.

## 15.5 Resets

The reset values of registers and signals are described in Section 15.3, “Memory Map and Register Definition”. The behavior of the LCD40F4BV3 system during reset is described in Section 15.4.1, “LCD Driver Description”.

## 15.6 Interrupts

This module does not generate any interrupts.

# Chapter 16
# Motor Controller (MC10B8CV1)

**Table 16-1. Revision History**

| Version Number | Revision Date | Author | Description of Changes |
|---|---|---|---|
| V01.01 | 6-OCT-2009 | | Table 16-12<br>- fixed 2nd content row : MnCyP := $\overline{PWM}$<br>- fixed 4th content row : MnCyP := 0 |

## 16.1 Introduction

The block MC10B8C is a PWM motor controller suitable to drive instruments in a cluster configuration or any other loads requiring a PWM signal. The motor controller has four PWM channels associated with two pins each (8 pins in total).

### 16.1.1 Features

The MC10B8C includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- 7-bit resolution mode (fast mode): duty cycle can be changed by accessing only 1 byte/output
- Left, right, or center aligned PWM
- This module is suited for, but not limited to, driving small stepper and air core motors used in instrumentation applications. This module can be used for other motor control or PWM applications that match the frequency, resolution, and output drive capabilities of the module.

### 16.1.2 Modes of Operation

#### 16.1.2.1 Functional Modes

##### 16.1.2.1.1 PWM Resolution

The motor controller can be configured to either 11- or 7-bits resolution mode by clearing or setting the FAST bit. This bit influences all PWM channels. For details, please refer to Section 16.3.2.5, “Motor Controller Duty Cycle Registers”.

#### 16.1.2.1.2 Dither Function

Dither function can be selected or deselected by setting or clearing the DITH bit. This bit influences all PWM channels. For details, please refer to Section 16.4.1.3.5, “Dither Bit (DITH)”.

### 16.1.2.2 PWM Channel Configuration Modes

The eight PWM channels can operate in three functional modes. Those modes are, with some restrictions, selectable for each channel independently.

#### 16.1.2.2.1 Dual Full H-Bridge Mode

This mode is suitable to drive a stepper motor or a 360$^{o}$ air gauge instrument. For details, please refer to Section 16.4.1.1.1, “Dual Full H-Bridge Mode (MCOM = 11)”. In this mode two adjacent PWM channels are combined, and two PWM channels drive four pins.

#### 16.1.2.2.2 Full H-Bridge Mode

This mode is suitable to drive any load requiring a PWM signal in a H-bridge configuration using two pins. For details please refer to Section 16.4.1.1.2, “Full H-Bridge Mode (MCOM = 10)”.

#### 16.1.2.2.3 Half H-Bridge Mode

This mode is suitable to drive a 90$^{o}$ instrument driven by one pin. For details, please refer to Section 16.4.1.1.3, “Half H-Bridge Mode (MCOM = 00 or 01)”.

### 16.1.2.3 PWM Alignment Modes

Each PWM channel can operate independently in three different alignment modes. For details, please refer to Section 16.4.1.3.1, “PWM Alignment Modes”.

### 16.1.2.4 Low-Power Modes

The behavior of the motor controller in low-power modes is programmable. For details, please refer to Section 16.4.5, “Operation in Wait Mode” and Section 16.4.6, “Operation in Stop and Pseudo-Stop Modes”.

## 16.1.3 Block Diagram

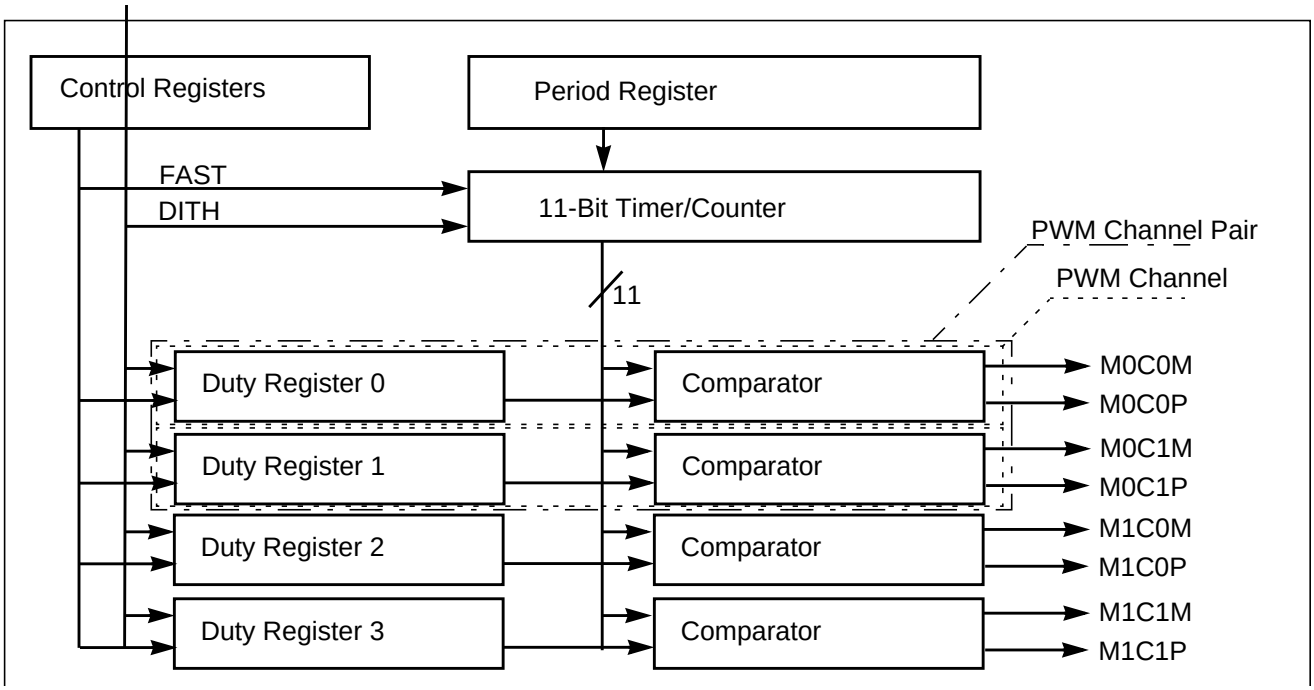



**Figure 16-1. MC10B8C Block Diagram**

## 16.2 External Signal Description

The motor controller is associated with 8 pins. Table 16-2 lists the relationship between the PWM channels and signal pins as well as PWM channel pair (motor number), coils, and nodes they are supposed to drive if all channels are set to dual full H-bridge configuration.

**Table 16-2. PWM Channel and Pin Assignment**

| Pin Name | PWM Channel | PWM Channel Pair[1] | Coil | Node |
|---|---|---|---|---|
| M0C0M | 0 | 0 | 0 | Minus |
| M0C0P | | | | Plus |
| M0C1M | 1 | | 1 | Minus |
| M0C1P | | | | Plus |
| M1C0M | 2 | 1 | 0 | Minus |
| M1C0P | | | | Plus |
| M1C1M | 3 | | 1 | Minus |
| M1C1P | | | | Plus |

1 A PWM Channel Pair always consists of PWM channel x and PWM channel x+1 (x = 2·n). The term “PWM Channel Pair” is equivalent to the term “Motor”. E.g. Channel Pair 0 is equivalent to Motor 0

### 16.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM Output Pins for Motor 0

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 0. PWM output on M0C0M results in a positive current flow through coil 0 when M0C0P is driven to a logic high state. PWM output on M0C1M results in a positive current flow through coil 1 when M0C1P is driven to a logic high state.

### 16.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM Output Pins for Motor 1

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 1. PWM output on M1C0M results in a positive current flow through coil 0 when M1C0P is driven to a logic high state. PWM output on M1C1M results in a positive current flow through coil 1 when M1C1P is driven to a logic high state.

## 16.3 Memory Map and Register Definition

This section provides a detailed description of all registers of the 11-bit 4-channel motor controller module.

### 16.3.1 Module Memory Map

Figure 16-2 shows the memory map of the 11-bit 4-channel motor controller module.

**Figure 16-2. MC10B8C Memory Map**

| Offset | Register | Access |
|---|---|---|
| 0x0000 | Motor Controller Control Register 0 (MCCTL0) | RW |
| 0x0001 | Motor Controller Control Register 1 (MCCTL1) | RW |
| 0x0002 | Motor Controller Period Register (High Byte) | RW |
| 0x0003 | Motor Controller Period Register (Low Byte) | RW |
| 0x0004 | Reserved[1] | — |
| 0x0005 | Reserved | — |
| 0x0006 | Reserved | — |
| 0x0007 | Reserved | — |
| 0x0008 | Reserved | — |
| 0x0009 | Reserved | — |
| 0x000A | Reserved | — |
| 0x000B | Reserved | — |
| 0x000C | Reserved | — |
| 0x000D | Reserved | — |
| 0x000E | Reserved | — |
| 0x000F | Reserved | — |
| 0x0010 | Motor Controller Channel Control Register 0 (MCCC0) | RW |
| 0x0011 | Motor Controller Channel Control Register 1 (MCCC1) | RW |
| 0x0012 | Motor Controller Channel Control Register 2 (MCCC2) | RW |
| 0x0013 | Motor Controller Channel Control Register 3 (MCCC3) | RW |
| 0x0014 | Reserved | — |
| 0x0015 | Reserved | — |
| 0x0016 | Reserved | — |
| 0x0017 | Reserved | — |
| 0x0018 | Reserved | — |
| 0x0019 | Reserved | — |
| 0x001A | Reserved | — |
| 0x001B | Reserved | — |
| 0x001C | Reserved | — |
| 0x001D | Reserved | — |
| 0x001E | Reserved | — |
| 0x001F | Reserved | — |
| 0x0020 | Motor Controller Duty Cycle Register 0 (MCDC0) — High Byte | RW |
| 0x0021 | Motor Controller Duty Cycle Register 0 (MCDC0) — Low Byte | RW |
| 0x0022 | Motor Controller Duty Cycle Register 1 (MCDC1) — High Byte | RW |
| 0x0023 | Motor Controller Duty Cycle Register 1 (MCDC1) — Low Byte | RW |
| 0x0024 | Motor Controller Duty Cycle Register 2 (MCDC2) — High Byte | RW |
| 0x0025 | Motor Controller Duty Cycle Register 2 (MCDC2) — Low Byte | RW |
| 0x0026 | Motor Controller Duty Cycle Register 3 (MCDC3) — High Byte | RW |
| 0x0027 | Motor Controller Duty Cycle Register 3 (MCDC3) — Low Byte | RW |

**Figure 16-2. MC10B8C Memory Map (continued)**

| Offset | Register | Access |
|---|---|---|
| 0x0028 | Reserved | — |
| 0x0029 | Reserved | — |
| 0x002A | Reserved | — |
| 0x002B | Reserved | — |
| 0x002C | Reserved | — |
| 0x002D | Reserved | — |
| 0x002E | Reserved | — |
| 0x002F | Reserved | — |
| 0x0030 | Reserved | — |
| 0x0031 | Reserved | — |
| 0x0032 | Reserved | — |
| 0x0033 | Reserved | — |
| 0x0034 | Reserved | — |
| 0x0035 | Reserved | — |
| 0x0036 | Reserved | — |
| 0x0037 | Reserved | — |
| 0x0038 | Reserved | — |
| 0x0039 | Reserved | — |
| 0x003A | Reserved | — |
| 0x003B | Reserved | — |
| 0x003C | Reserved | — |
| 0x003D | Reserved | — |
| 0x003E | Reserved | — |
| 0x003F | Reserved | — |

[1] Write accesses to “Reserved” addresses have no effect. Read accesses to “Reserved” addresses provide **invalid** data (0x0000).

## 16.3.2 Register Descriptions

### 16.3.2.1 Motor Controller Control Register 0

This register controls the operating mode of the motor controller module.

Offset Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | MCPRE[1:0] | | MCSWAI | FAST | DITH | 0 | MCTOIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-3. Motor Controller Control Register 0 (MCCTL0)**

**Table 16-3. MCCTL0 Field Descriptions**

| Field | Description |
|---|---|
| 6:5<br>MCPRE[1:0] | **Motor Controller Prescaler Select** — MCPRE1 and MCPRE0 determine the prescaler value that sets the motor controller timer counter clock frequency ($f_{TC}$). The clock source for the prescaler is the peripheral bus clock ($f_{BUS}$) as shown in Figure 16-22. Writes to MCPRE1 or MCPRE0 will not affect the timer counter clock frequency $f_{TC}$ until the start of the next PWM period. Table 16-4 shows the prescaler values that result from the possible combinations of MCPRE1 and MCPRE0 |
| 4<br>MCSWAI | **Motor Controller Module Stop in Wait Mode**<br>0 Entering wait mode has no effect on the motor controller module and the associated port pins maintain the functionality they had prior to entering wait mode both during wait mode and after exiting wait mode.<br>1 Entering wait mode will stop the clock of the module and debias the analog circuitry. The module will release the pins. |
| 3<br>FAST | **Motor Controller PWM Resolution Mode**<br>0 PWM operates in 11-bit resolution mode, duty cycle registers of all channels are switched to word mode.<br>1 PWM operates in 7-bit resolution (fast) mode, duty cycle registers of all channels are switched to byte mode. |
| 2<br>DITH | **Motor Control/Driver Dither Feature Enable** (refer to Section 16.4.1.3.5, “Dither Bit (DITH)”)<br>0 Dither feature is disabled.<br>1 Dither feature is enabled. |
| 0<br>MCTOIF | **Motor Controller Timer Counter Overflow Interrupt Flag** — This bit is set when a motor controller timer counter overflow occurs. The bit is cleared by writing a 1 to the bit.<br>0 A motor controller timer counter overflow has not occurred since the last reset or since the bit was cleared.<br>1 A motor controller timer counter overflow has occurred. |

**Table 16-4. Prescaler Values**

| MCPRE[1:0] | $f_{TC}$ |
|---|---|
| 00 | $f_{Bus}$ |
| 01 | $f_{Bus}/2$ |
| 10 | $f_{Bus}/4$ |
| 11 | $f_{Bus}/8$ |

## 16.3.2.2 Motor Controller Control Register 1

This register controls the behavior of the analog section of the motor controller as well as the interrupt enables.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RECIRC | 0 | 0 | 0 | 0 | 0 | 0 | MCTOIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-4. Motor Controller Control Register 1 (MCCTL1)**

**Table 16-5. MCCTL1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RECIRC | **Recirculation in (Dual) Full H-Bridge Mode** (refer to Section 16.4.1.3.3, “RECIRC Bit”)— RECIRC only affects the outputs in (dual) full H-bridge modes. In half H-bridge mode, the PWM output is always active low. RECIRC = 1 will also invert the effect of the S bits (refer to Section 16.4.1.3.2, “Sign Bit (S)”) in (dual) full H-bridge modes. RECIRC must be changed only while no PWM channel is operating in (dual) full H-bridge mode; otherwise, erroneous output pattern may occur.<br>0 Recirculation on the high side transistors. Active state for PWM output is logic low, the static channel will output logic high.<br>1 Recirculation on the low side transistors. Active state for PWM output is logic high, the static channel will output logic low. |
| 0<br>MCTOIE | **Motor Controller Timer Counter Overflow Interrupt Enable**<br>0 Interrupt disabled.<br>1 Interrupt enabled. An interrupt will be generated when the motor controller timer counter overflow interrupt flag (MCTOIF) is set. |

### 16.3.2.3 Motor Controller Period Register

The period register defines PER, the number of motor controller timer counter clocks a PWM period lasts. The motor controller timer counter is clocked with the frequency $f_{TC}$. If dither mode is enabled (DITH = 1, refer to Section 16.4.1.3.5, “Dither Bit (DITH)”), P0 is ignored and reads as a 0. In this case PER = 2 * D[10:1].

Offset Module Base + 0x0002, 0x0003

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | P10 | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-5. Motor Controller Period Register (MCPER) with DITH = 0**

Offset Module Base + 0x0002, 0x0003

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | P10 | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-6. Motor Controller Period Register (MCPER) with DITH = 1**

For example, programming MCPER to 0x0022 (PER = 34 decimal) will result in 34 counts for each complete PWM period. Setting MCPER to 0 will shut off all PWM channels as if MCAM[1:0] is set to 0 in all channel control registers after the next period timer counter overflow. In this case, the motor controller releases all pins.

**NOTE**

Programming MCPER to 0x0001 and setting the DITH bit will be managed as if MCPER is programmed to 0x0000. All PWM channels will be shut off after the next period timer counter overflow.

### 16.3.2.4 Motor Controller Channel Control Registers

Each PWM channel has one associated control register to control output delay, PWM alignment, and output mode. The registers are named MCCC0... MCCCn. In the following, MCCC0 is described as a reference for all other channel registers.

Offset Module Base + 0x0010 . . . 0x0017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MCOM1 | MCOM0 | MCAM1 | MCAM0 | 0 | 0 | CD1 | CD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-7. Motor Controller Control Register Channel 0 (MCCC0)**

**Table 16-6. MCCC0 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>MCOM[1:0] | **Output Mode** — MCOM1, MCOM0 control the PWM channel's output mode. See Table 16-7. |
| 5:4<br>MCAM[1:0] | **PWM Channel Alignment Mode** — MCAM1, MCAM0 control the PWM channel's PWM alignment mode and operation. See Table 16-8.<br>MCAM[1:0] and MCOM[1:0] are double buffered. The values used for the generation of the output waveform will be copied to the working registers either at once (if all PWM channels are disabled or MCPER is set to 0) or if a timer counter overflow occurs. Reads of the register return the most recent written value, which are not necessarily the currently active values. |
| 1:0<br>CD[1:0] | **PWM Channel Delay** — Each PWM channel can be individually delayed by a programmable number of PWM timer counter clocks. The delay will be n/$f_{TC}$. See Table 16-9. |

**Table 16-7. Output Mode**

| MCOM[1:0] | Output Mode |
|---|---|
| 00 | Half H-bridge mode, PWM on MnCxM, MnCxP is released |
| 01 | Half H-bridge mode, PWM on MnCxP, MnCxM is released |
| 10 | Full H-bridge mode |
| 11 | Dual full H-bridge mode |

**Table 16-8. PWM Alignment Mode**

| MCAM[1:0] | PWM Alignment Mode |
|---|---|
| 00 | Channel disabled |
| 01 | Left aligned |
| 10 | Right aligned |
| 11 | Center aligned |

**Table 16-9. Channel Delay**

| CD[1:0] | n [# of PWM Clocks] |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 2 |
| 11 | 3 |

**NOTE**

The PWM motor controller will release the pins after the next PWM timer counter overflow without accommodating any channel delay if a single channel has been disabled or if the period register has been cleared or all channels have been disabled. Program one or more inactive PWM frames (duty cycle = 0) before writing a configuration that disables a single channel or the entire PWM motor controller.

### 16.3.2.5 Motor Controller Duty Cycle Registers

Each duty cycle register sets the sign and duty functionality for the respective PWM channel.

The contents of the duty cycle registers define DUTY, the number of motor controller timer counter clocks the corresponding output is driven low (RECIRC = 0) or is driven high (RECIRC = 1). Setting all bits to 0 will give a static high output in case of RECIRC = 0; otherwise, a static low output. Values greater than or equal to the contents of the period register will generate a static low output in case of RECIRC = 0, or a static high output if RECIRC = 1. The layout of the duty cycle registers differ dependent upon the state of the FAST bit in the control register 0.

Offset Module Base + 0x0020 . . . 0x002F — Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | S | S | S | S | S | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-8. Motor Controller Duty Cycle Register x (MCDCx) with FAST = 0**

Offset Module Base + 0x0020 . . . 0x002F — Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | S | D8 | D7 | D6 | D5 | D4 | D3 | D2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-9. Motor Controller Duty Cycle Register x (MCDCx) with FAST = 1**

**Table 16-10. MCDCx Field Descriptions**

| Field | Description |
|---|---|
| 0<br>S | SIGN — The SIGN bit is used to define which output will drive the PWM signal in (dual) full-H-bridge modes. The SIGN bit has no effect in half-bridge modes. See Section 16.4.1.3.2, “Sign Bit (S)”, and table Table 16-12 for detailed information about the impact of RECIRC and SIGN bit on the PWM output. |

Whenever FAST = 1, the bits D10, D9, D1, and D0 will be set to 0 if the duty cycle register is written.

For example setting MCDCx = 0x0158 with FAST = 0 gives the same output waveform as setting MCDCx = 0x5600 with FAST = 1 (with FAST = 1, the low byte of MCDCx needs not to be written).

The state of the FAST bit has impact only during write and read operations. A change of the FAST bit (set or clear) without writing a new value does not impact the internal interpretation of the duty cycle values.

To prevent the output from inconsistent signals, the duty cycle registers are double buffered. The motor controller module will use working registers to generate the output signals. The working registers are copied from the bus accessible registers at the following conditions:

- MCPER is set to 0 (all channels are disabled in this case)
- MCAM[1:0] of the respective channel is set to 0 (channel is disabled)
- A PWM timer counter overflow occurs while in half H-bridge or full H-bridge mode
- A PWM channel pair is configured to work in Dual Full H-Bridge mode and a PWM timer counter overflow occurs after the odd[1] duty cycle register of the channel pair has been written.

In this way, the output of the PWM will always be either the old PWM waveform or the new PWM waveform, not some variation in between.

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active sign, duty cycle, and dither functionality due to the double buffering scheme.

1. Odd duty cycle register: MCDCx+1, x = 2·n

# 16.4 Functional Description

## 16.4.1 Modes of Operation

### 16.4.1.1 PWM Output Modes

The motor controller is configurable between three output modes.

- Dual full H-bridge mode can be used to control either a stepper motor or a 360° air core instrument. In this case two PWM channels are combined.
- In full H-bridge mode, each PWM channel is updated independently.
- In half H-bridge mode, one pin of the PWM channel can generate a PWM signal to control a 90° air core instrument (or other load requiring a PWM signal) and the other pin is unused.

The mode of operation for each PWM channel is determined by the corresponding MCOM[1:0] bits in channel control registers. After a reset occurs, each PWM channel will be disabled, the corresponding pins are released.

Each PWM channel consists of two pins. One output pin will generate a PWM signal. The other will operate as logic high or low output depending on the state of the RECIRC bit (refer to Section 16.4.1.3.3, “RECIRC Bit”), while in (dual) full H-bridge mode, or will be released, while in half H-bridge mode. The state of the S bit in the duty cycle register determines the pin where the PWM signal is driven in full H-bridge mode. While in half H-bridge mode, the state of the released pin is determined by other modules associated with this pin.

Associated with each PWM channel pair n are two PWM channels, x and x + 1, where x = 2 * n and n (0, 1) is the PWM channel pair number. Duty cycle register x controls the sign of the PWM signal (which pin drives the PWM signal) and the duty cycle of the PWM signal for motor controller channel x. The pins associated with PWM channel x are MnC0P and MnC0M. Similarly, duty cycle register x + 1 controls the sign of the PWM signal and the duty cycle of the PWM signal for channel x + 1. The pins associated with PWM channel x + 1 are MnC1P and MnC1M. This is summarized in Table 16-11.

**Table 16-11. Corresponding Registers and Pin Names for Each PWM Channel Pair**

| PWM Channel Pair Number | PWM Channel Control Register | Duty Cycle Register | Channel Number | Pin Names |
|---|---|---|---|---|
| n | MCMCx | MCDCx | PWM Channel x, x = 2·n | MnC0M |
| | | | | MnC0P |
| | MCMCx + 1 | MCDCx + 1 | PWM Channel x + 1, x = 2·n | MnC1M |
| | | | | MnC1P |
| 0 | MCMC0 | MCDC0 | PWM Channel 0 | M0C0M |
| | | | | M0C0P |
| | MCMC1 | MCDC1 | PWM Channel 1 | M0C1M |
| | | | | M0C1P |

**Table 16-11. Corresponding Registers and Pin Names for Each PWM Channel Pair (continued)**

| PWM Channel Pair Number | PWM Channel Control Register | Duty Cycle Register | Channel Number | Pin Names |
|---|---|---|---|---|
| 1 | MCMC2 | MCDC2 | PWM Channel 2 | M1C0M |
| | | | | M1C0P |
| | MCMC3 | MCDC3 | PWM Channel 3 | M1C1M |
| | | | | M1C1P |

#### 16.4.1.1.1 Dual Full H-Bridge Mode (MCOM = 11)

PWM channel pairs x and x + 1 operate in dual full H-bridge mode if both channels have been enabled (MCAM[1:0]=01, 10, or 11) and both of the corresponding output mode bits MCOM[1:0] in both PWM channel control registers are set.

A typical configuration in dual full H-bridge mode is shown in Figure 16-10. PWM channel x drives the PWM output signal on either MnC0P or MnC0M. If MnC0P drives the PWM signal, MnC0M will be output either high or low depending on the RECIRC bit. If MnC0M drives the PWM signal, MnC0P will be an output high or low. PWM channel x + 1 drives the PWM output signal on either MnC1P or MnC1M. If MnC1P drives the PWM signal, MnC1M will be an output high or low. If MnC1M drives the PWM signal, MnC1P will be an output high or low. This results in motor recirculation currents on the high side drivers (RECIRC = 0) while the PWM signal is at a logic high level, or motor recirculation currents on the low side drivers (RECIRC = 1) while the PWM signal is at a logic low level. The pin driving the PWM signal is determined by the S (sign) bit in the corresponding duty cycle register and the state of the RECIRC bit. The value of the PWM duty cycle is determined by the value of the D[10:0] or D[8:2] bits respectively in the duty cycle register depending on the state of the FAST bit.

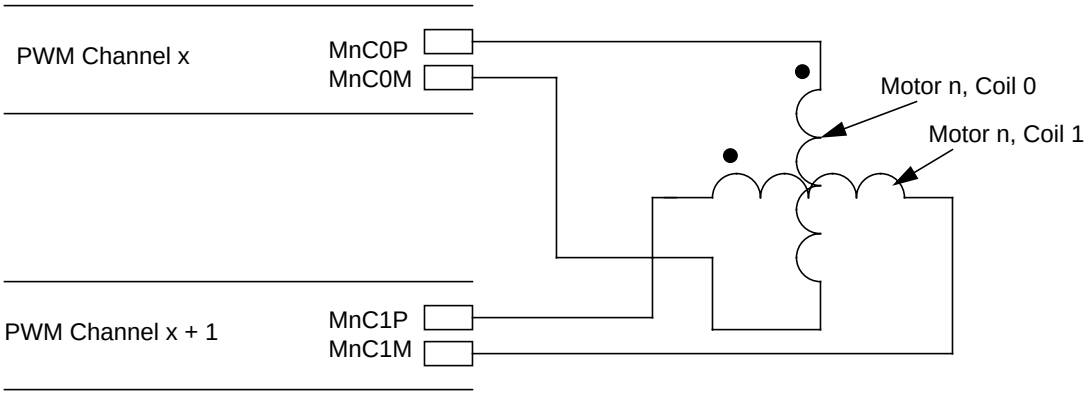



**Figure 16-10. Typical Dual Full H-Bridge Mode Configuration**

Whenever FAST = 0 only 16-bit write accesses to the duty cycle registers are allowed, 8-bit write accesses can lead to unpredictable duty cycles.

While fast mode is enabled (FAST = 1), 8-bit write accesses to the high byte of the duty cycle registers are allowed, because only the high byte of the duty cycle register is used to determine the duty cycle.

The following sequence should be used to update the current magnitude and direction for coil 0 and coil 1 of the motor to achieve consistent PWM output:

1. Write to duty cycle register x
2. Write to duty cycle register x + 1.

At the next timer counter overflow, the duty cycle registers will be copied to the working duty cycle registers. Sequential writes to the duty cycle register x will result in the previous data being overwritten.

#### 16.4.1.1.2 Full H-Bridge Mode (MCOM = 10)

In full H-bridge mode, the PWM channels x and x + 1 operate independently. The duty cycle working registers are updated whenever a timer counter overflow occurs.

#### 16.4.1.1.3 Half H-Bridge Mode (MCOM = 00 or 01)

In half H-bridge mode, the PWM channels x and x + 1 operate independently. In this mode, each PWM channel can be configured such that one pin is released and the other pin is a PWM output. Figure 16-11 shows a typical configuration in half H-bridge mode.

The two pins associated with each channel are switchable between released mode and PWM output dependent upon the state of the MCOM[1:0] bits in the MCCCx (channel control) register. See register description in Section 16.3.2.4, “Motor Controller Channel Control Registers”. In half H-bridge mode, the state of the S bit has no effect.

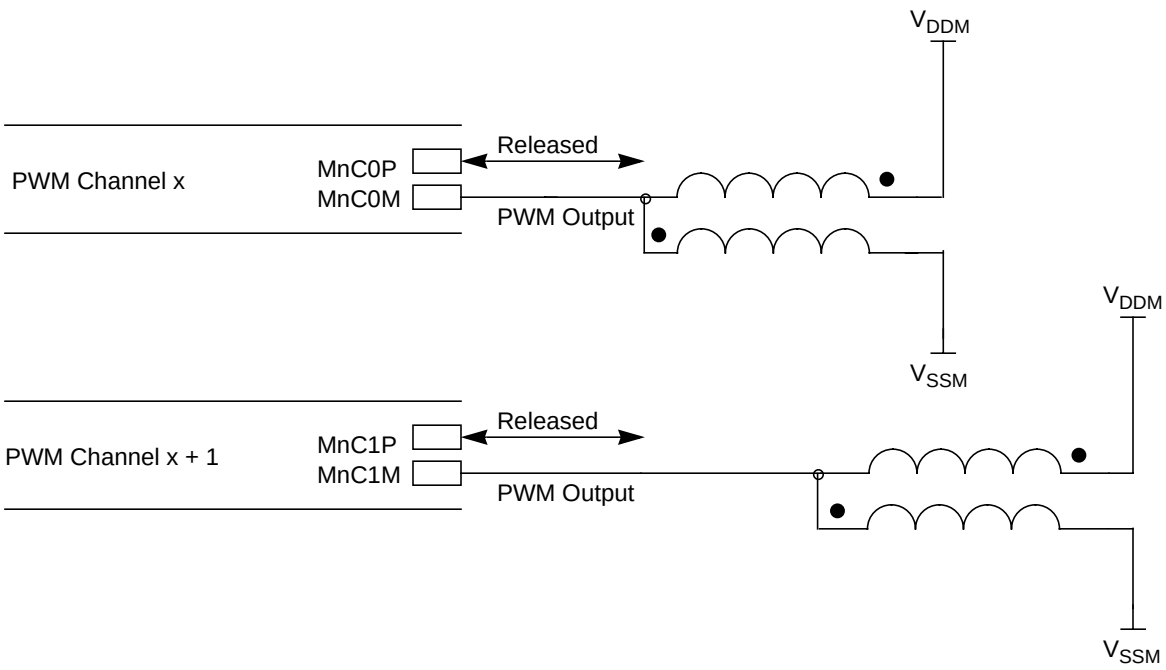



**Figure 16-11. Typical Quad Half H-Bridge Mode Configuration**

### 16.4.1.2 Relationship Between PWM Mode and PWM Channel Enable

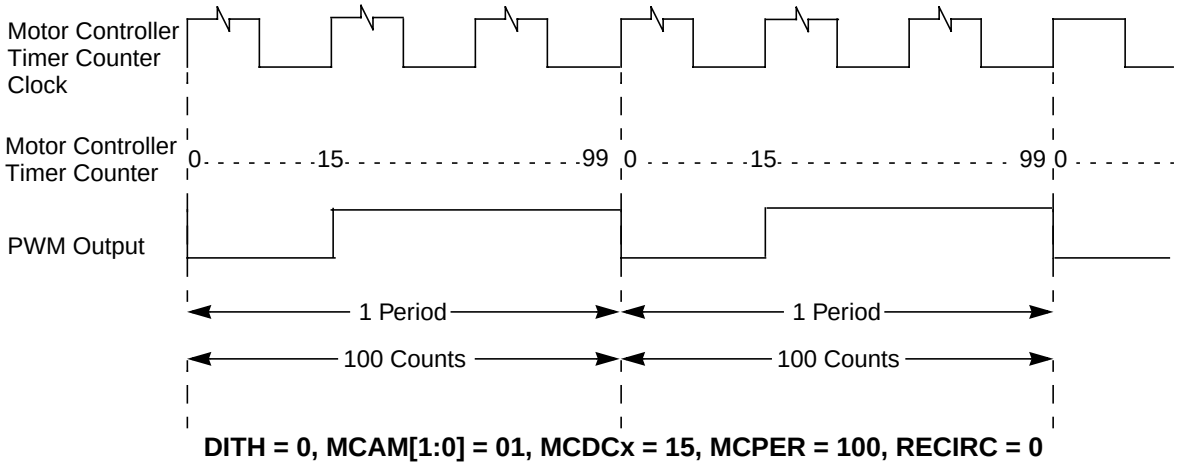
The pair of motor controller channels cannot be placed into dual full H-bridge mode unless both motor controller channels have been enabled (MCAM[1:0] not equal to 00) and dual full H-bridge mode is selected for both PWM channels (MCOM[1:0] = 11). If only one channel is set to dual full H-bridge mode, this channel will operate in full H-bridge mode, the other as programmed.

### 16.4.1.3 Relationship Between Sign, Duty, Dither, RECIRC, Period, and PWM Mode Functions

#### 16.4.1.3.1 PWM Alignment Modes

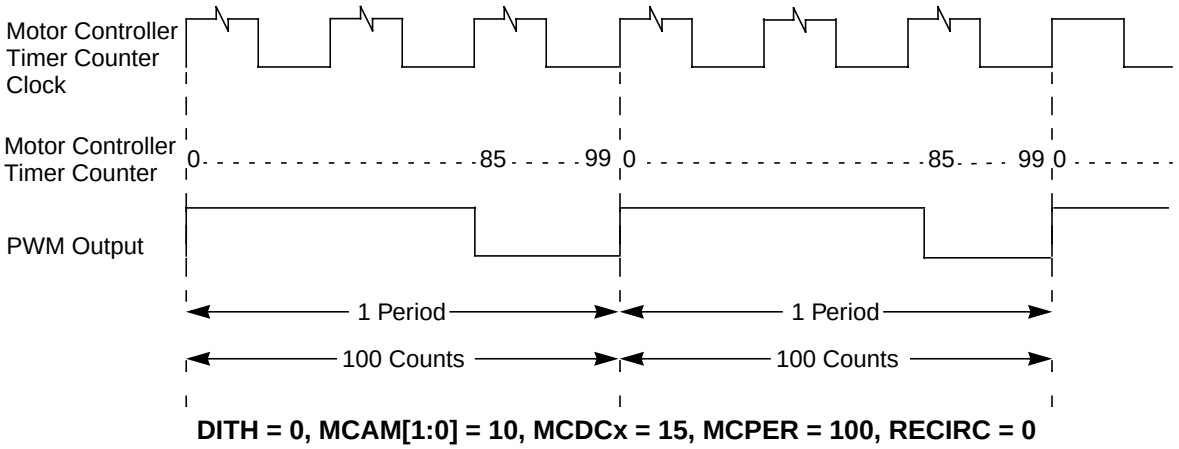
Each PWM channel can be programmed individually to three different alignment modes. The mode is determined by the MCAM[1:0] bits in the corresponding channel control register.

Left aligned (MCAM[1:0] = 01): The output will start active (low if RECIRC = 0 or high if RECIRC = 1) and will turn inactive (high if RECIRC = 0 or low if RECIRC = 1) after the number of counts specified by the corresponding duty cycle register.



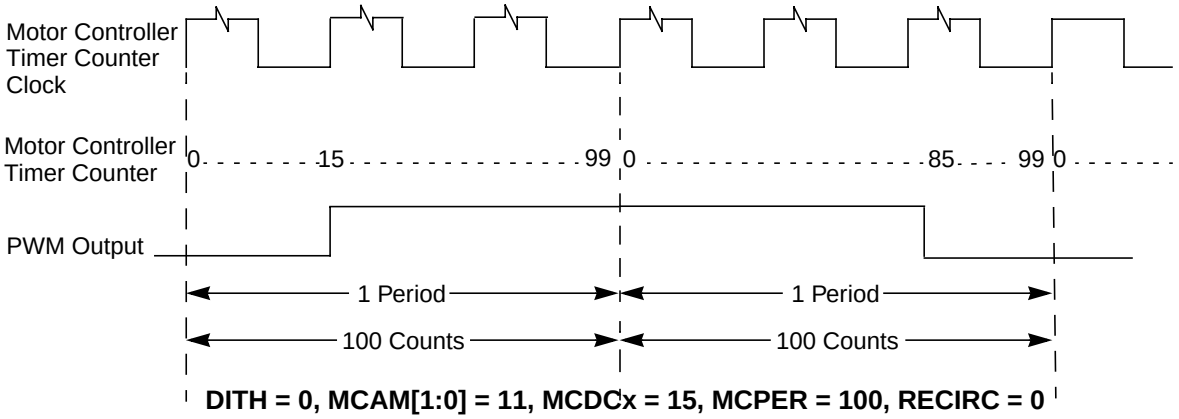

**DITH = 0, MCAM[1:0] = 01, MCDCx = 15, MCPER = 100, RECIRC = 0**

Right aligned (MCAM[1:0] = 10): The output will start inactive (high if RECIRC = 0 and low if RECIRC = 1) and will turn active after the number of counts specified by the difference of the contents of period register and the corresponding duty cycle register.




**DITH = 0, MCAM[1:0] = 10, MCDCx = 15, MCPER = 100, RECIRC = 0**

Center aligned (MCAM[1:0] = 11): Even periods will be output left aligned, odd periods will be output right aligned. PWM operation starts with the even period after the channel has been enabled. PWM operation in center aligned mode might start with the odd period if the channel has not been disabled before changing the alignment mode to center aligned.




**DITH = 0, MCAM[1:0] = 11, MCDCx = 15, MCPER = 100, RECIRC = 0**

### 16.4.1.3.2 Sign Bit (S)

Assuming RECIRC = 0 (the active state of the PWM signal is low), when the S bit for the corresponding channel is cleared, MnC0P (if the PWM channel number is even, n = 0, 1, 2, 3, see Table 16-11) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2, 3, see Table 16-11), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the S bit has no effect. The PWM output signal is generated on MnC0M (if the PWM channel number is even, n = 0, 1, 2, 3, see Table 16-11) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2, 3).

Assuming RECIRC = 0 (the active state of the PWM signal is low), when the S bit for the corresponding channel is set, MnC0M (if the PWM channel number is even, n = 0, 1, 2, 3, see Table 16-11) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2, 3, see Table 16-11), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the S bit has no effect. The PWM output signal is generated on MnC0P (if the PWM channel number is even, n = 0, 1, 2, 3, see Table 16-11) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2, 3).

Setting RECIRC = 1 will also invert the effect of the S bit such that while S = 0, MnC0P or MnC1P will generate the PWM signal and MnC0M or MnC1M will be a static low output. While S = 1, MnC0M or MnC1M will generate the PWM signal and MnC0P or MnC1P will be a static low output. In this case the active state of the PWM signal will be high.

See Table 16-12 for detailed information about the impact of SIGN and RECIRC bit on the PWM output.

**Table 16-12. Impact of RECIRC and SIGN Bit on the PWM Output**

| Output Mode | RECIRC | SIGN | MnCyM | MnCyP |
|---|---|---|---|---|
| (Dual) Full H-Bridge | 0 | 0 | $\overline{PWM}$[1] | 1 |
| (Dual) Full H-Bridge | 0 | 1 | 1 | $\overline{PWM}$ |
| (Dual) Full H-Bridge | 1 | 0 | 0 | PWM[2] |
| (Dual) Full H-Bridge | 1 | 1 | PWM | 0 |
| Half H-Bridge: PWM on MnCyM | Don't care | Don't care | PWM | —[3] |
| Half H-Bridge: PWM on MnCyP | Don't care | Don't care | — | PWM |

1 $\overline{PWM}$: The PWM signal is low active. e.g., the waveform starts with 0 in left aligned mode. Output M generates the PWM signal. Output P is static high.
2 PWM: The PWM signal is high active. e.g., the waveform starts with 1 in left aligned mode. output P generates the PWM signal. Output M is static low.
3 The state of the output transistors is not controlled by the motor controller.

#### 16.4.1.3.3 RECIRC Bit

The RECIRC bit controls the flow of the recirculation current of the load. Setting RECIRC = 0 will cause recirculation current to flow through the high side transistors, and RECIRC = 1 will cause the recirculation current to flow through the low side transistors. The RECIRC bit is only active in (dual) full H-bridge modes.

Effectively, RECIRC = 0 will cause a static high output on the output terminal not driven by the PWM, RECIRC = 1 will cause a static low output on the output terminals not driven by the PWM. To achieve the same current direction, the S bit behavior is inverted if RECIRC = 1. Figure 16-12, Figure 16-13, Figure 16-14, and Figure 16-15 illustrate the effect of the RECIRC bit in (dual) full H-bridge modes.

RECIRC bit must be changed only while no PWM channel is operated in (dual) full H-bridge mode.

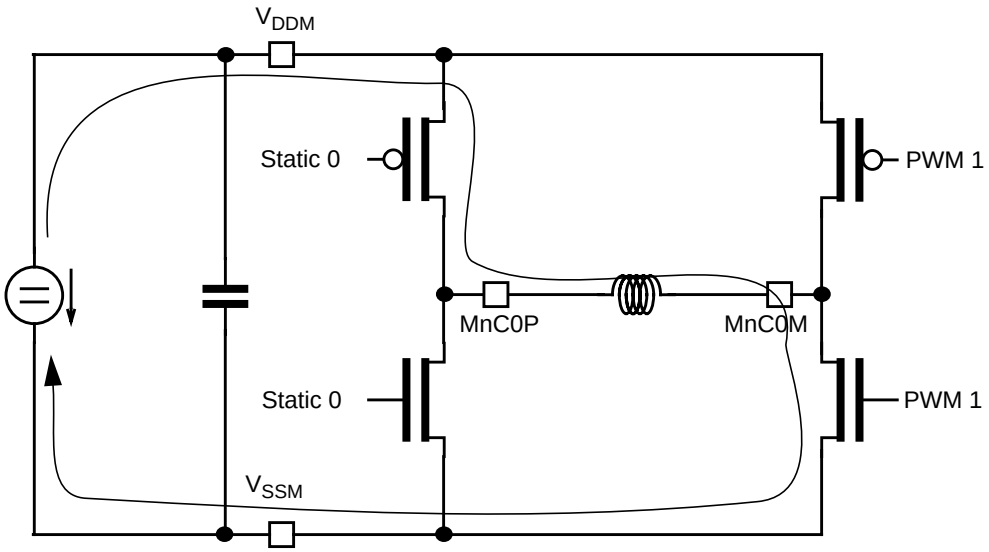



**Figure 16-12. PWM Active Phase, RECIRC = 0, S = 0**

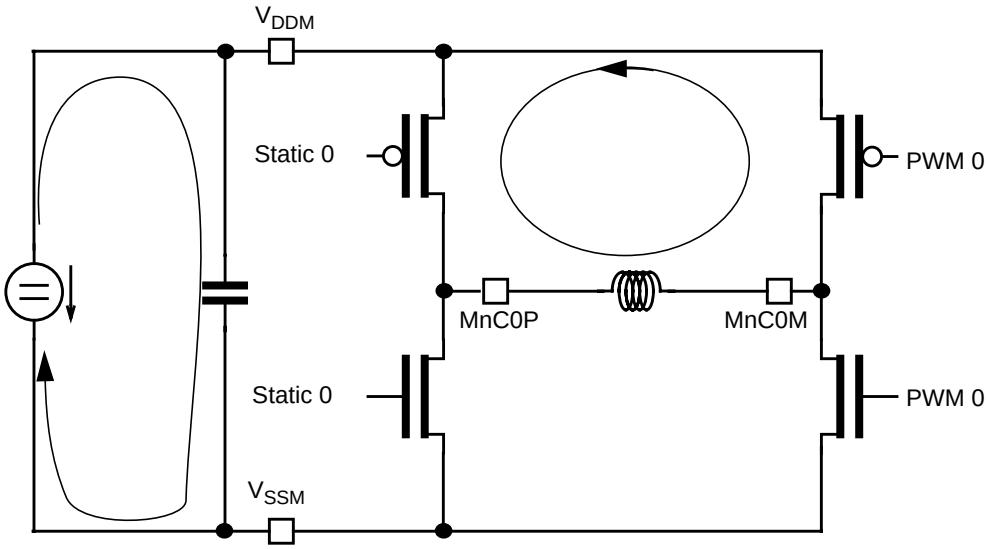



**Figure 16-13. PWM Passive Phase, RECIRC = 0, S = 0**

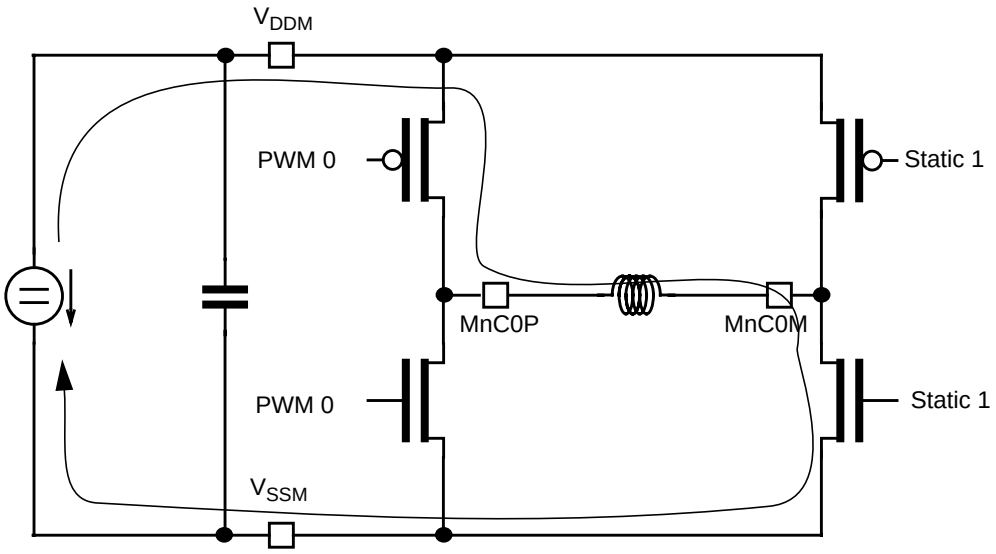



**Figure 16-14. PWM Active Phase, RECIRC = 1, S = 0**

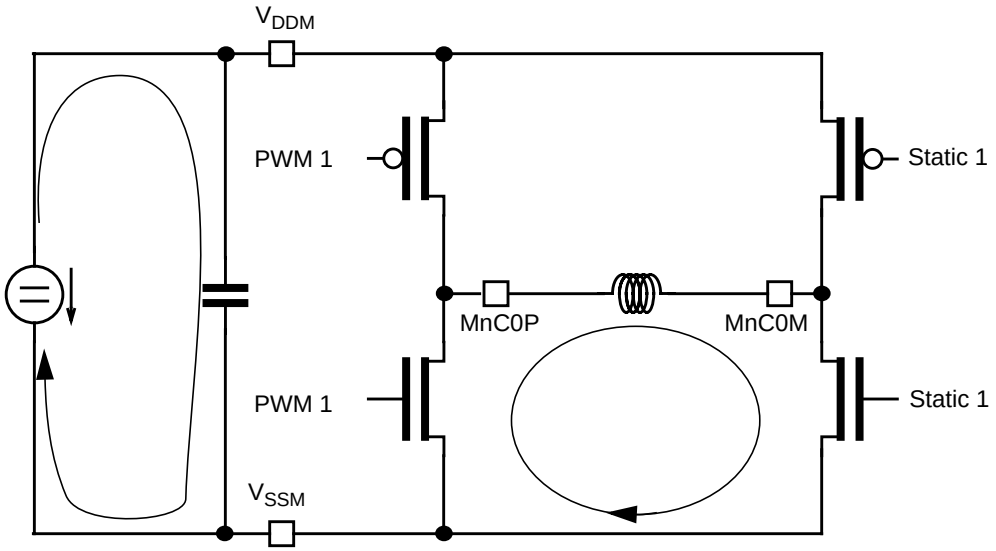



**Figure 16-15. PWM Passive Phase, RECIRC = 1, S = 0**

#### 16.4.1.3.4 Relationship Between RECIRC Bit, S Bit, MCOM Bits, PWM State, and Output Transistors

Please refer to Figure 16-16 for the output transistor assignment.

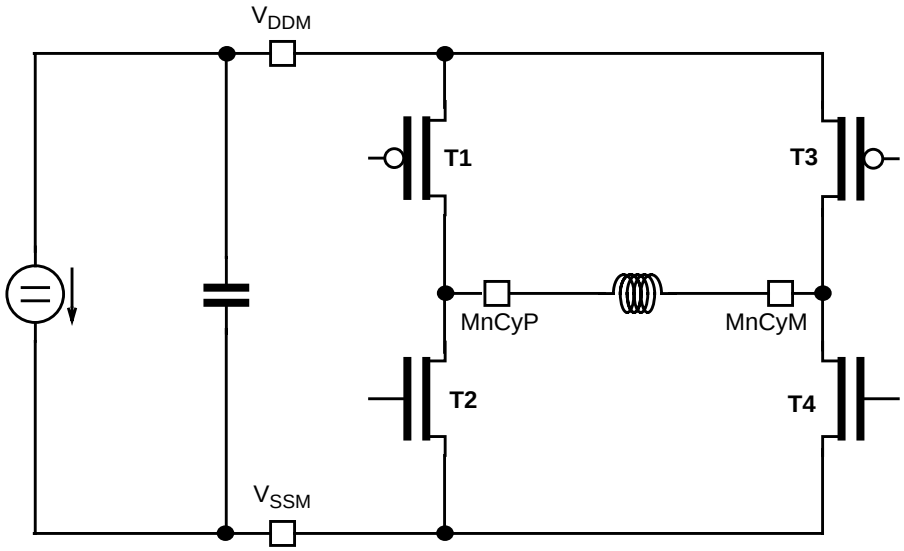



**Figure 16-16. Output Transistor Assignment**

Table 16-13 illustrates the state of the output transistors in different states of the PWM motor controller module. '—' means that the state of the output transistor is not controlled by the motor controller.

**Table 16-13. State of Output Transistors in Various Modes**

| Mode | MCOM[1:0] | PWM Duty | RECIRC | S | T1 | T2 | T3 | T4 |
|---|---|---|---|---|---|---|---|---|
| Off | Don't care | — | Don't care | Don't care | — | — | — | — |
| Half H-Bridge | 00 | Active | Don't care | Don't care | — | — | OFF | ON |
| Half H-Bridge | 00 | Passive | Don't care | Don't care | — | — | ON | OFF |
| Half H-Bridge | 01 | Active | Don't care | Don't care | OFF | ON | — | — |
| Half H-Bridge | 01 | Passive | Don't care | Don't care | ON | OFF | — | — |
| (Dual) Full | 10 or 11 | Active | 0 | 0 | ON | OFF | OFF | ON |
| (Dual) Full | 10 or 11 | Passive | 0 | 0 | ON | OFF | ON | OFF |
| (Dual) Full | 10 or 11 | Active | 0 | 1 | OFF | ON | ON | OFF |
| (Dual) Full | 10 or 11 | Passive | 0 | 1 | ON | OFF | ON | OFF |
| (Dual) Full | 10 or 11 | Active | 1 | 0 | ON | OFF | OFF | ON |
| (Dual) Full | 10 or 11 | Passive | 1 | 0 | OFF | ON | OFF | ON |
| (Dual) Full | 10 or 11 | Active | 1 | 1 | OFF | ON | ON | OFF |
| (Dual) Full | 10 or 11 | Passive | 1 | 1 | OFF | ON | OFF | ON |

#### 16.4.1.3.5 Dither Bit (DITH)

The purpose of the dither mode is to increase the minimum length of output pulses without decreasing the PWM resolution, in order to limit the pulse distortion introduced by the slew rate control of the outputs. If dither mode is selected the output pattern will repeat after two timer counter overflows. For the same output frequency, the shortest output pulse will have twice the length while dither feature is selected. To achieve the same output frame frequency, the prescaler of the MC10B8C module has to be set to twice the division rate if dither mode is selected; e.g., with the same prescaler division rate the repeat rate of the output pattern is the same as well as the shortest output pulse with or without dither mode selected.

The DITH bit in control register 0 enables or disables the dither function.

DITH = 0: dither function is disabled.

When DITH is cleared and assuming left aligned operation and RECIRC = 0, the PWM output will start at a logic low level at the beginning of the PWM period (motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 11-bit PWM duty cycle value, DUTY, contained in D[10:0] in MCDCx. When a match (output compare between motor controller timer counter and DUTY) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the contents of MCPER – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes one PWM period. The PWM period repeats every P counts (as defined by the bits P[10:0] in the motor controller period register) of the motor controller timer counter. If DUTY >= P, the output will be static low. If DUTY = 0x0000, the output will be continuously at a logic high level. The relationship between the motor controller timer counter clock, motor controller timer counter value, and PWM output while DITH = 0 is shown in Figure 16-17.

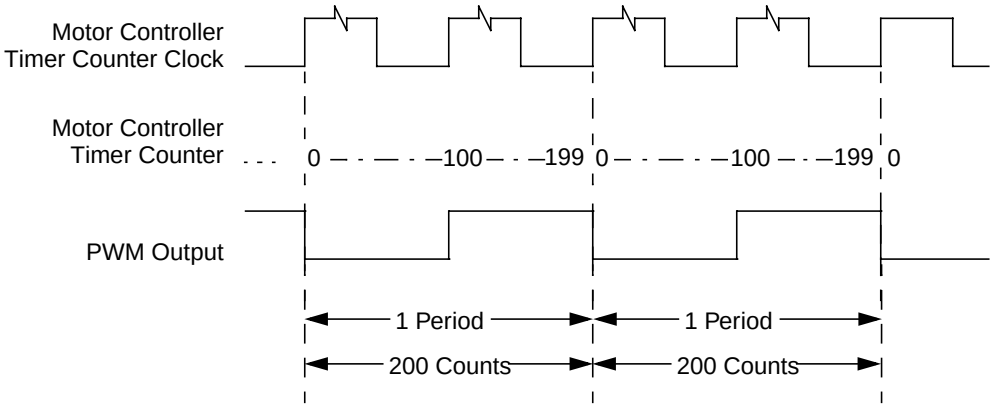



**Figure 16-17. PWM Output: DITH = 0, MCAM[1:0] = 01, MCDC = 100, MCPER = 200, RECIRC = 0**

DITH = 1: dither function is enabled

Please note if DITH = 1, the bit P0 in the motor controller period register will be internally forced to 0 and read always as 0.

When DITH is set and assuming left aligned operation and RECIRC = 0, the PWM output will start at a logic low level at the beginning of the PWM period (when the motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 10-bit PWM duty cycle

value, DUTY, contained in D[10:1] in MCDCx. When a match (output compare between motor controller timer counter and DUTY) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by P[10:1] – 1 in MCPER). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes the first half of the PWM period. During the second half of the PWM period, the PWM output will remain at a logic low level until either the motor controller timer counter matches the 10-bit PWM duty cycle value, DUTY, contained in D[10:1] in MCDCx if D0 = 0, or the motor controller timer counter matches the 10-bit PWM duty cycle value + 1 (the value of D[10:1] in MCDCx is increment by 1 and is compared with the motor controller timer counter value) if D0 = 1 in the corresponding duty cycle register. When a match occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by P[10:1] – 1 in MCPER). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level.

This process will repeat every number of counts of the motor controller timer counter defined by the period register contents (P[10:0]). If the output is neither set to 0% nor to 100% there will be four edges on the PWM output per PWM period in this case. Therefore, the PWM output compare function will alternate between DUTY and DUTY + 1 every half PWM period if D0 in the corresponding duty cycle register is set to 1. The relationship between the motor controller timer counter clock ($f_{TC}$), motor controller timer counter value, and left aligned PWM output if DITH = 1 is shown in Figure 16-18 and Figure 16-19. Figure 16-20 and Figure 16-21 show right aligned and center aligned PWM operation respectively, with dither feature enabled and D0 = 1. Please note: In the following examples, the MCPER value is defined by the bits P[10:0], which is, if DITH = 1, always an even number.

**NOTE**

The DITH bit must be changed only if the motor controller is disabled (all channels disabled or period register cleared) to avoid erroneous waveforms.

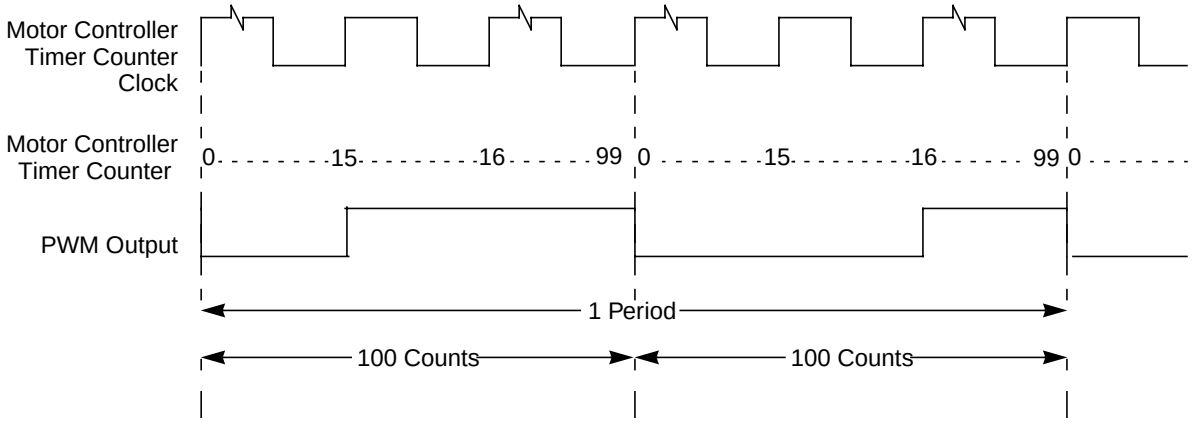



**Figure 16-18. PWM Output: DITH = 1, MCAM[1:0] = 01, MCDC = 31, MCPER = 200, RECIRC = 0**

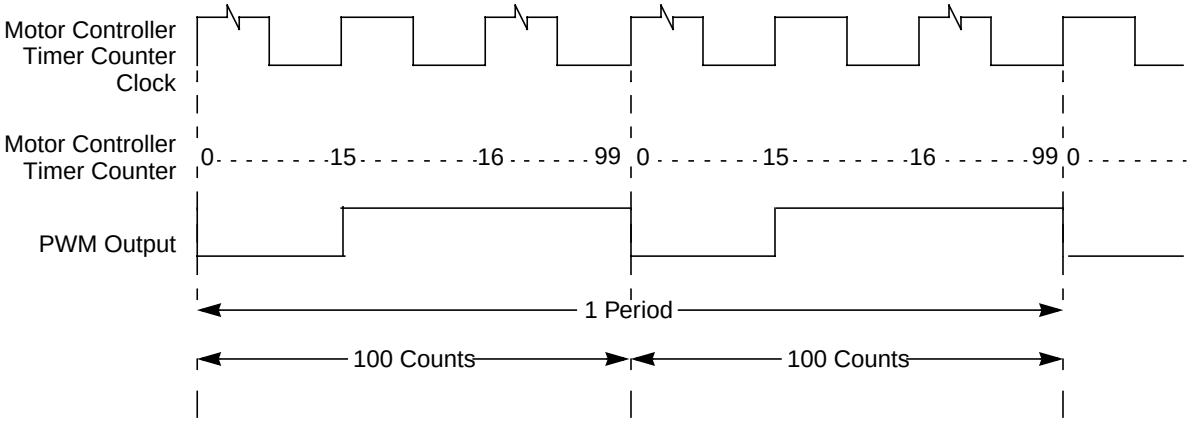



**Figure 16-19. PWM Output: DITH = 1, MCAM[1:0] = 01, MCDC = 30, MCPER = 200, RECIRC = 0**

.

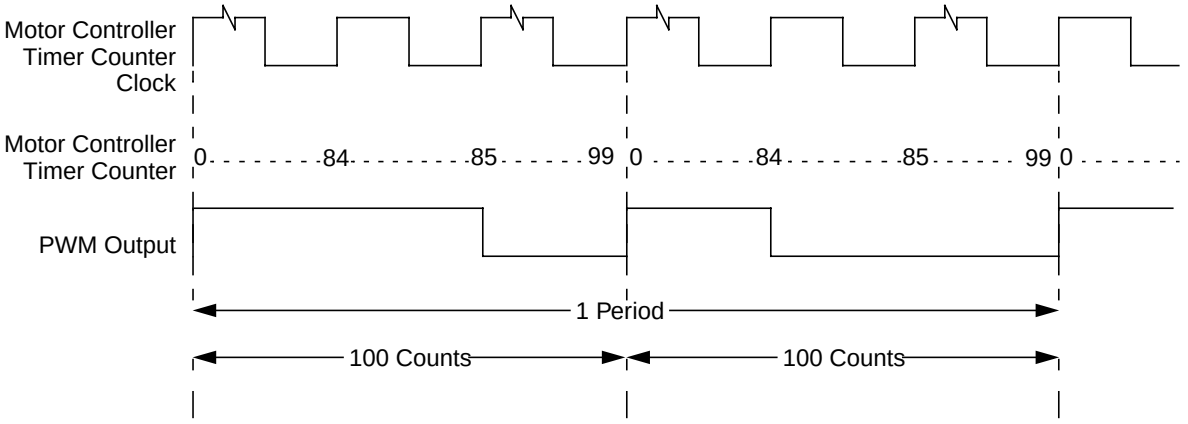



**Figure 16-20. PWM Output: DITH = 1, MCAM[1:0] = 10, MCDC = 31, MCPER = 200, RECIRC = 0**

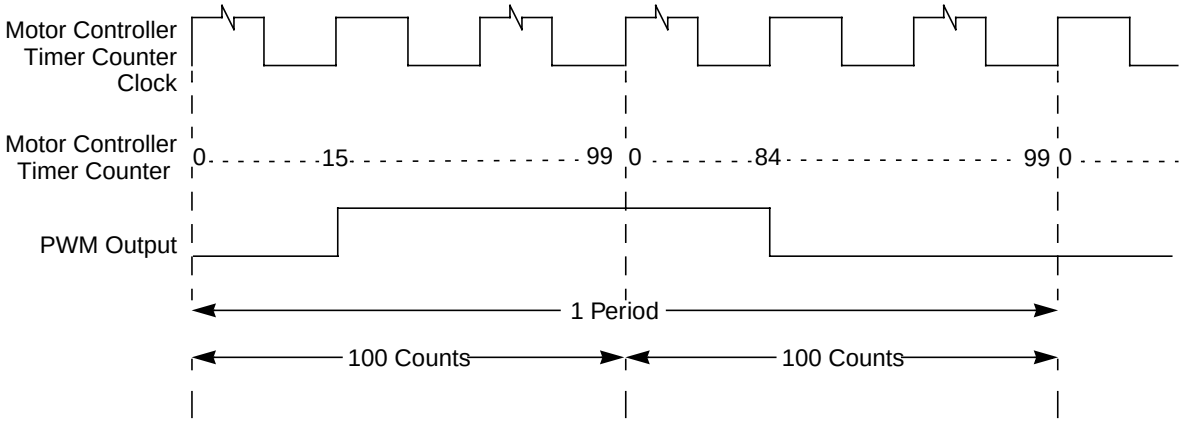



**Figure 16-21. PWM Output: DITH = 1, MCAM[1:0] = 11, MCDC = 31, MCPER = 200, RECIRC = 0**

## 16.4.2 PWM Duty Cycle

The PWM duty cycle for the motor controller channel x can be determined by dividing the decimal representation of bits D[10:0] in MCDCx by the decimal representation of the bits P[10:0] in MCPER and multiplying the result by 100% as shown in the equation below:

$$\text{Effective PWM Channel X \% Duty Cycle} = \frac{\text{DUTY}}{\text{MCPER}} \cdot 100\%$$

**NOTE**

x = PWM Channel Number = 0, 1, 2, 3 ... 8. This equation is only valid if DUTY <= MCPER and MCPER is not equal to 0.

Whenever D[10:0] >= P[10:0], a constant low level (RECIRC = 0) or high level (RECIRC = 1) will be output.

## 16.4.3 Motor Controller Counter Clock Source

Figure 16-22 shows how the PWM motor controller timer counter clock source is selected.

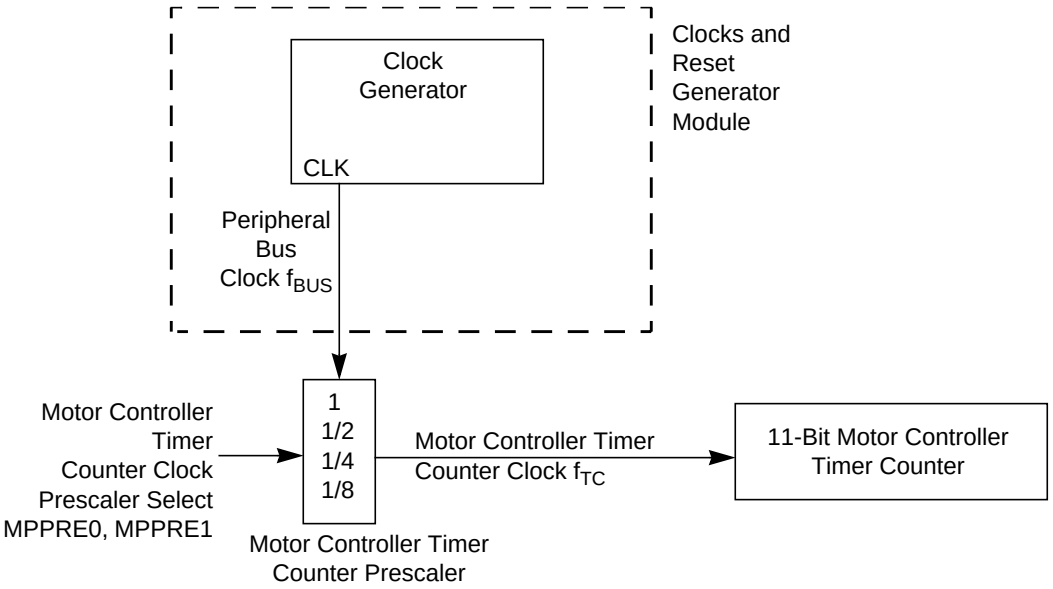



**Figure 16-22. Motor Controller Counter Clock Selection**

The peripheral bus clock is the source for the motor controller counter prescaler. The motor controller counter clock rate, $f_{TC}$, is set by selecting the appropriate prescaler value. The prescaler is selected with the MCPRE[1:0] bits in motor controller control register 0 (MCCTL0). The motor controller channel frequency of operation can be calculated using the following formula if DITH = 0:

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot \text{M}}$$

The motor controller channel frequency of operation can be calculated using the following formula if DITH = 1:

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{MCPER \cdot M/2}$$

**NOTE**

Both equations are only valid if MCPER is not equal to 0. M = 1 for left or right aligned mode, M = 2 for center aligned mode.

Table 16-14 shows examples of the motor controller channel frequencies that can be generated based on different peripheral bus clock frequencies and the prescaler value.

**Table 16-14. Motor Controller Channel Frequencies (Hz), MCPER = 256, DITH = 0, MCAM = 10, 01**

| Prescaler | Peripheral Bus Clock Frequency | | | | |
|---|---|---|---|---|---|
| | 16 MHz | 10 MHz | 8 MHz | 5 MHz | 4 MHz |
| 1 | 62500 | 39063 | 31250 | 19531 | 15625 |
| 1/2 | 31250 | 19531 | 15625 | 9766 | 7813 |
| 1/4 | 15625 | 9766 | 7813 | 4883 | 3906 |
| 1/8 | 7813 | 4883 | 3906 | 2441 | 1953 |

**NOTE**

Due to the selectable slew rate control of the outputs, clipping may occur on short output pulses.

### 16.4.4 Output Switching Delay

In order to prevent large peak current draw from the motor power supply, selectable delays can be used to stagger the high logic level to low logic level transitions on the motor controller outputs. The timing delay, $t_d$, is determined by the CD[1:0] bits in the corresponding channel control register (MCMCx) and is selectable between 0, 1, 2, or 3 motor controller timer counter clock cycles.

**NOTE**

A PWM channel gets disabled at the next timer counter overflow without notice of the switching delay.

### 16.4.5 Operation in Wait Mode

During wait mode, the operation of the motor controller pins are selectable between the following two options:

1. MCSWAI = 1: All module clocks are stopped and the associated port pins are set to their inactive state, which is defined by the state of the RECIRC bit during wait mode. The motor controller module registers stay the same as they were prior to entering wait mode. Therefore, after exiting from wait mode, the associated port pins will resume to the same functionality they had prior to entering wait mode.
2. MCSWAI = 0: The PWM clocks continue to run and the associated port pins maintain the functionality they had prior to entering wait mode both during wait mode and after exiting wait mode.

### 16.4.6 Operation in Stop and Pseudo-Stop Modes

All module clocks are stopped and the associated port pins are set to their inactive state, which is defined by the state of the RECIRC bit. The motor controller module registers stay the same as they were prior to entering stop or pseudo-stop modes. Therefore, after exiting from stop or pseudo-stop modes, the associated port pins will resume to the same functionality they had prior to entering stop or pseudo-stop modes.

## 16.5 Reset

The motor controller is reset by system reset. All associated ports are released, all registers of the motor controller module will switch to their reset state as defined in Section 16.3.2, “Register Descriptions”.

## 16.6 Interrupts

The motor controller has one interrupt source.

### 16.6.1 Timer Counter Overflow Interrupt

An interrupt will be requested when the MCTOIE bit in the motor controller control register 1 is set and the running PWM frame is finished. The interrupt is cleared by either setting the MCTOIE bit to 0 or to write a 1 to the MCTOIF bit in the motor controller control register 0.

## 16.7 Initialization/Application Information

This section provides an example of how the PWM motor controller can be initialized and used by application software. The configuration parameters (e.g., timer settings, duty cycle values, etc.) are not guaranteed to be adequate for any real application.

The example software is implemented in assembly language.

### 16.7.1 Code Example

One way to use the motor controller is:

1. Perform global initialization
   a) Set the motor controller control registers MCCTL0 and MCCTL1 to appropriate values.
      i) Prescaler disabled (MCPRE1 = 0, MCPRE0 = 0).
      ii) Fast mode and dither disabled (FAST = 0, DITH = 0).
      iii) Recirculation feature in dual full H-bridge mode disabled (RECIRC = 0).
      All other bits in MCCTL0 and MCCTL1 are set to 0.
   b) Configure the channel control registers for the desired mode.
      i) Dual full H-bridge mode (MCOM[1:0] = 11).
      ii) Left aligned PWM (MCAM[1:0] = 01).
      iii) No channel delay (MCCD[1:0] = 00).
2. Perform the startup phase
   a) Clear the duty cycle registers MCDC0 and MCDC1
   b) Initialize the period register MCPER, which is equivalent to enabling the motor controller.
   c) Enable the timer which generates the timebase for the updates of the duty cycle registers.
3. Main program
   a) Check if pin PB0 is set to “1” and execute the sub program if a timer interrupt is pending.
   b) Initiate the shutdown procedure if pin PB0 is set to “0”.
4. Sub program
   a) Update the duty cycle registers
      Load the duty cycle registers MCDC0 and MCDC1 with new values from the table and clear the timer interrupt flag.
      The sub program will initiate the shutdown procedure if pin PB0 is set to “0”.
   b) Shutdown procedure

The timer is disabled and the duty cycle registers are cleared to drive an inactive value on the PWM output as long as the motor controller is enabled. The period register is cleared after a certain time, which disables the motor controller. The table address is restored and the timer interrupt flag is cleared.

# Chapter 17
# Stepper Stall Detector (SSDV2) Block Description

**Table 17-1. SSDV2 Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V02.00 | 1st Aug 2011 | | -remove the SMS bit in RTZCTL register |

## 17.1 Introduction

The stepper stall detector (SSD) block provides a circuit to measure and integrate the induced voltage on the non-driven coil of a stepper motor using full steps when the gauge pointer is returning to zero (RTZ). During the RTZ event, the pointer is returned to zero using full steps in either clockwise or counter clockwise direction, where only one coil is driven at any point in time. The back electromotive force (EMF) signal present on the non-driven coil is integrated after a blanking time, and its results stored in a 16-bit accumulator. The 16-bit modulus down counter can be used to monitor the blanking time and the integration time. The value in the accumulator represents the change in linked flux (magnetic flux times the number of turns in the coil) and can be compared to a stored threshold. Values above the threshold indicate a moving motor, in which case the pointer can be advanced another full step in the same direction and integration be repeated. Values below the threshold indicate a stalled motor, thereby marking the cessation of the RTZ event.

### 17.1.1 Modes of Operation

- Return to zero modes
  - — Blanking with no drive
  - — Blanking with drive
  - — Conversion
  - — Integration
- Low-power modes

### 17.1.2 Features

- Programmable full step state
- Programmable integration polarity
- Blanking (recirculation) state
- 16-bit integration accumulator register

- 16-bit modulus down counter with interrupt

## 17.1.3 Block Diagram

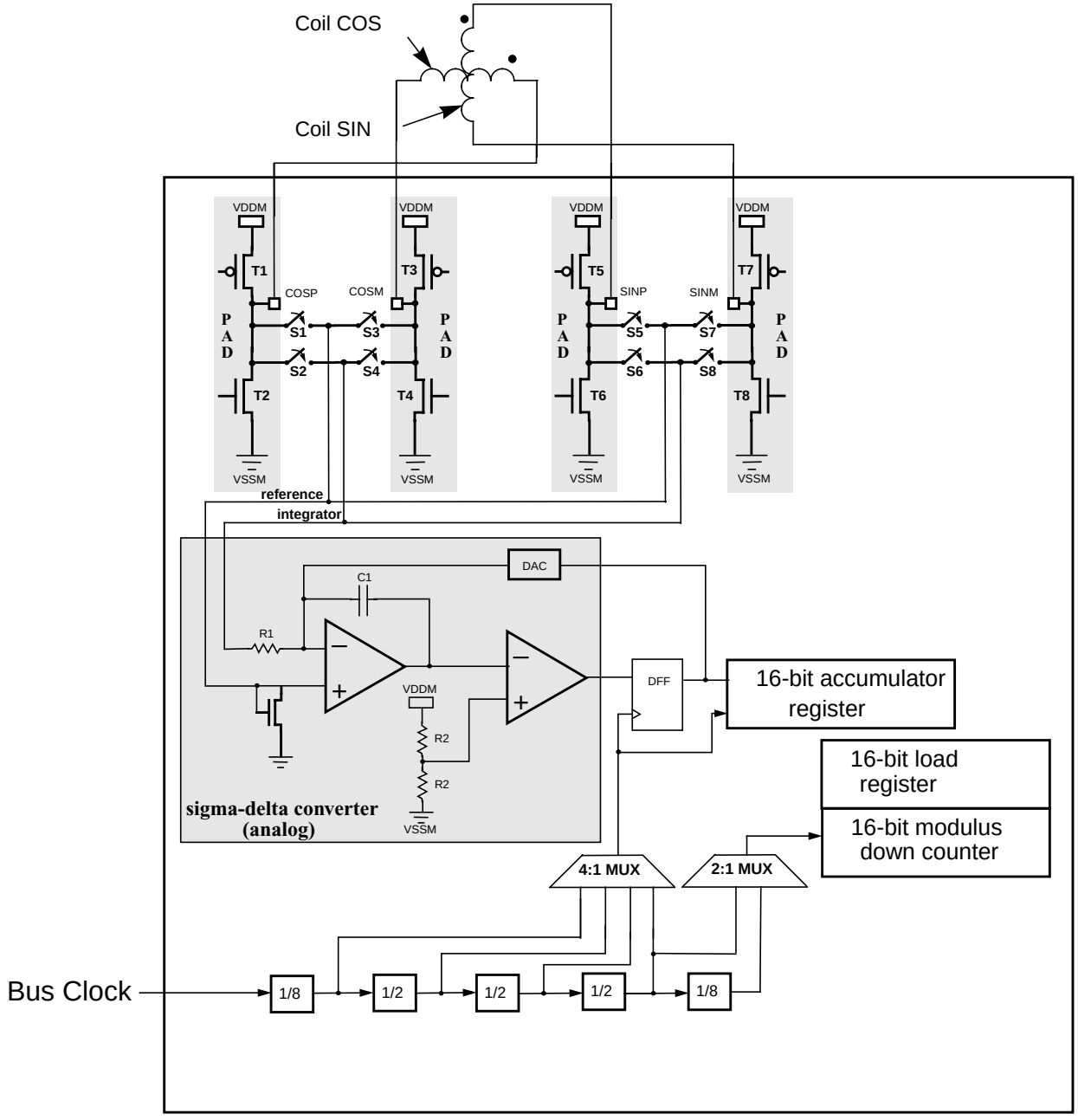



**Figure 17-1. SSD Block Diagram**

## 17.2 External Signal Description

Each SSD signal is the output pin of a half bridge, designed to source or sink current. The H-bridge pins drive the sine and cosine coils of a stepper motor to provide four-quadrant operation.

**Table 17-2. Pin Table**

| Pin Name | Node | Coil |
|---|---|---|
| COSM | Minus | COS |
| COSP | Plus | |
| SINM | Minus | SIN |
| SINP | Plus | |

### 17.2.1 COSM/COSP — Cosine Coil Pins for Motor

These pins interface to the cosine coils of a stepper motor to measure the back EMF for calibration of the pointer reset position.

### 17.2.2 SINM/SINP — Sine Coil Pins for Motor

These pins interface to the sine coils of a stepper motor to measure the back EMF for calibration of the pointer reset position.

## 17.3 Memory Map and Register Definition

This section provides a detailed description of all registers of the stepper stall detector (SSD) block.

### 17.3.1 Module Memory Map

Table 17-3 gives an overview of all registers in the SSDV2 memory map. The SSDV2 occupies eight bytes in the memory space. The register address results from the addition of *base address* and *address offset.* The *base address* is determined at the MCU level and is given in the Device Overview chapter. The *address offset* is defined at the block level and is given here.

**Table 17-3. SSDV2 Memory Map**

| Address Offset | Use | Access |
|---|---|---|
| 0x0000 | RTZCTL | R/W |
| 0x0001 | MDCCTL | R/W |
| 0x0002 | SSDCTL | R/W |
| 0x0003 | SSDFLG | R/W |
| 0x0004 | MDCCNT (High) | R/W |
| 0x0005 | MDCCNT (Low) | R/W |
| 0x0006 | ITGACC (High) | R |
| 0x0007 | ITGACC (Low) | R |

### 17.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the SSDV2 block. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

### 17.3.2.1 Return-to-Zero Control Register (RTZCTL)

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ITG | DCOIL | RCIR | POL | 0 | 0 | STEP | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-2. Return-to-Zero Control Register (RTZCTL)**

Read: anytime

Write: anytime

**Table 17-4. RTZCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ITG | **Integration** — During return to zero (RTZE = 1), one of the coils must be recirculated or non-driven determined by the STEP field. If the ITG bit is set, the coil is non-driven, and if the ITG bit is clear, the coil is being recirculated. Table 17-5 shows the condition state of each transistor from Figure 17-1 based on the STEP, ITG, DCOIL and RCIR bits.<br>Regardless of the RTZE bit value, if the ITG bit is set, one end of the non-driven coil connects to the (non-zero) reference input and the other end connects to the integrator input of the sigma-delta converter. Regardless of the RTZE bit value, if the ITG bit is clear, the non-driven coil is in a blanking state, the converter is in a reset state, and the accumulator is initialized to zero. Table 17-6 shows the condition state of each switch from Figure 17-1 based on the ITG, STEP and POL bits.<br>0 Blanking<br>1 Integration |
| 6<br>DCOIL | **Drive Coil** — During return to zero (RTZE=1), one of the coils must be driven determined by the STEP field. If the DCOIL bit is set, this coil is driven. If the DCOIL bit is clear, this coil is disconnected or drivers turned off. Table 17-5 shows the condition state of each transistor from Figure 17-1 based on the STEP, ITG, DCOIL and RCIR bits.<br>0 Disconnect Coil<br>1 Drive Coil |
| 5<br>RCIR | **Recirculation in Blanking Mode** — During return to zero (RTZE = 1), one of the coils is recirculated prior to integration during the blanking period. This bit determines if the coil is recirculated via VDDM or via VSSM. Table 17-5 shows the condition state of each transistor from Figure 17-1 based on the STEP, ITG, DCOIL and RCIR bits.<br>0 Recirculation on the high side transistors<br>1 Recirculation on the low side transistors |
| 4<br>POL | **Polarity** — This bit determines which end of the non-driven coil is routed to the sigma-delta converter during conversion or integration mode. Table 17-6 shows the condition state of each switch from Figure 17-1 based on the ITG, STEP and POL bits. |
| 1:0<br>STEP | **Full Step State** — This field indicates one of the four possible full step states. Step 0 is considered the east pole or 0° angle, step 1 is the north Pole or 90° angle, step 2 is the west pole or 180° angle, and step 3 is the south pole or 270° angle. For each full step state, Table 17-7 shows the current through each of the two coils, and the coil nodes that are multiplexed to the sigma-delta converter during conversion or integration mode. |

**Table 17-5. Transistor Condition States (RTZE = 1)**

| STEP | ITG | DCOIL | RCIR | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| xx | 1 | 0 | x | OFF | OFF | OFF | OFF | OFF | OFF | OFF | OFF |
| 00 | 0 | 0 | 0 | OFF | OFF | OFF | OFF | ON | OFF | ON | OFF |
| 00 | 0 | 0 | 1 | OFF | OFF | OFF | OFF | OFF | ON | OFF | ON |
| 00 | 0 | 1 | 0 | ON | OFF | OFF | ON | ON | OFF | ON | OFF |
| 00 | 0 | 1 | 1 | ON | OFF | OFF | ON | OFF | ON | OFF | ON |
| 00 | 1 | 1 | x | ON | OFF | OFF | ON | OFF | OFF | OFF | OFF |
| 01 | 0 | 0 | 0 | ON | OFF | ON | OFF | OFF | OFF | OFF | OFF |
| 01 | 0 | 0 | 1 | OFF | ON | OFF | ON | OFF | OFF | OFF | OFF |
| 01 | 0 | 1 | 0 | ON | OFF | ON | OFF | ON | OFF | OFF | ON |
| 01 | 0 | 1 | 1 | OFF | ON | OFF | ON | ON | OFF | OFF | ON |
| 01 | 1 | 1 | x | OFF | OFF | OFF | OFF | ON | OFF | OFF | ON |
| 10 | 0 | 0 | 0 | OFF | OFF | OFF | OFF | ON | OFF | ON | OFF |
| 10 | 0 | 0 | 1 | OFF | OFF | OFF | OFF | OFF | ON | OFF | ON |
| 10 | 0 | 1 | 0 | OFF | ON | ON | OFF | ON | OFF | ON | OFF |
| 10 | 0 | 1 | 1 | OFF | ON | ON | OFF | OFF | ON | OFF | ON |
| 10 | 1 | 1 | x | OFF | ON | ON | OFF | OFF | OFF | OFF | OFF |
| 11 | 0 | 0 | 0 | ON | OFF | ON | OFF | OFF | OFF | OFF | OFF |
| 11 | 0 | 0 | 1 | OFF | ON | OFF | ON | OFF | OFF | OFF | OFF |
| 11 | 0 | 1 | 0 | ON | OFF | ON | OFF | OFF | ON | ON | OFF |
| 11 | 0 | 1 | 1 | OFF | ON | OFF | ON | OFF | ON | ON | OFF |
| 11 | 1 | 1 | x | OFF | OFF | OFF | OFF | OFF | ON | ON | OFF |

**Table 17-6. Switch Condition States (RTZE = 1 or 0)**

| ITG | STEP | POL | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | xx | x | Open | Open | Open | Open | Open | Open | Open | Open |
| 1 | 00 | 0 | Open | Open | Open | Open | Close | Open | Open | Close |
| 1 | 00 | 1 | Open | Open | Open | Open | Open | Close | Close | Open |
| 1 | 01 | 0 | Open | Close | Close | Open | Open | Open | Open | Open |
| 1 | 01 | 1 | Close | Open | Open | Close | Open | Open | Open | Open |
| 1 | 10 | 0 | Open | Open | Open | Open | Open | Close | Close | Open |
| 1 | 10 | 1 | Open | Open | Open | Open | Close | Open | Open | Close |
| 1 | 11 | 0 | Close | Open | Open | Close | Open | Open | Open | Open |
| 1 | 11 | 1 | Open | Close | Close | Open | Open | Open | Open | Open |

**Table 17-7. Full Step States**

| STEP | Pole | Angle | COSINE Coil Current | | SINE Coil Current | | Coil Node to Integrator input (Close Switch) | | Coil Node to Reference input (Close Switch) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | DCOIL = 0 | DCOIL = 1 | DCOIL = 0 | DCOIL = 1 | ITG = 1 POL = 0 | ITG = 1 POL = 1 | ITG = 1 POL = 0 | ITG = 1 POL = 1 |
| 0 | East | 0° | 0 | + I max | 0 | 0 | SINM (S8) | SINP (S6) | SINP (S5) | SINM (S7) |
| 1 | North | 90° | 0 | 0 | 0 | + I max | COSP (S2) | COSM (S4) | COSM (S3) | COSP (S1) |
| 2 | West | 180° | 0 | – I max | 0 | 0 | SINP (S6) | SINM (S8) | SINM (S7) | SINP (S5) |
| 3 | South | 270° | 0 | 0 | 0 | – I max | COSM (S4) | COSP (S2) | COSP (S1) | COSM (S3) |

## 17.3.2.2 Modulus Down Counter Control Register (MDCCTL)

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MCZIE | MODMC | RDMCL | PRE | 0 | MCEN | 0 | AOVIE |
| W | | | | | FLMC | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-3. Modulus Down Counter Control Register (MDCCTL)**

Read: anytime

Write: anytime.

**Table 17-8. MDCCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>MCZIE | **Modulus Counter Underflow Interrupt Enable**<br>0 Interrupt disabled.<br>1 Interrupt enabled. An interrupt will be generated when the modulus counter underflow interrupt flag (MCZIF) is set. |
| 6<br>MODMC | **Modulus Mode Enable**<br>0 The modulus counter counts down from the value in the counter register and will stop at 0x0000.<br>1 Modulus mode is enabled. When the counter reaches 0x0000, the counter is loaded with the latest value written to the modulus counter register.<br>**Note:** For proper operation, the MCEN bit should be cleared before modifying the MODMC bit in order to reset the modulus counter to 0xFFFF. |
| 5<br>RDMCL | **Read Modulus Down-Counter Load**<br>0 Reads of the modulus count register (MDCCNT) will return the present value of the count register.<br>1 Reads of the modulus count register (MDCCNT) will return the contents of the load register. |

**Table 17-8. MDCCTL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>PRE | **Prescaler**<br>0 The modulus down counter clock frequency is the bus frequency divided by 64.<br>1 The modulus down counter clock frequency is the bus frequency divided by 512.<br>**Note:** A change in the prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs. |
| 3<br>FLMC | **Force Load Register into the Modulus Counter Count Register** — This bit always reads zero.<br>0 Write zero to this bit has no effect.<br>1 Write one into this bit loads the load register into the modulus counter count register. |
| 2<br>MCEN | **Modulus Down-Counter Enable**<br>0 Modulus down-counter is disabled. The modulus counter (MDCCNT) is preset to 0xFFFF. This will prevent an early interrupt flag when the modulus down-counter is enabled.<br>1 Modulus down-counter is enabled. |
| 0<br>AOVIE | **Accumulator Overflow Interrupt Enable**<br>0 Interrupt disabled.<br>1 Interrupt enabled. An interrupt will be generated when the accumulator overflow interrupt flag (AOVIF) is set. |

## 17.3.2.3 Stepper Stall Detector Control Register (SSDCTL)

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTZE | SDCPU | SSDWAI | FTST | 0 | 0 | ACLKS | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-4. Stepper Stall Detector Control Register (SSDCTL)**

Read: anytime

Write: anytime

**Table 17-9. SSDCTL Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTZE | **Return to Zero Enable** — If this bit is set, the coils are controlled by the SSD and are configured into one of the four full step states as shown in Table 17-7. If this bit is cleared, the coils are not controlled by the SSD.<br>0 RTZ is disabled.<br>1 RTZ is enabled. |
| 6<br>SDCPU | **Sigma-Delta Converter Power Up** — This bit provides on/off control for the sigma-delta converter allowing reduced MCU power consumption. Because the analog circuit is turned off when powered down, the sigma-delta converter requires a recovery time after it is powered up.<br>0 Sigma-delta converter is powered down.<br>1 Sigma-delta converter is powered up. |

**Table 17-9. SSDCTL Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>SSDWAI | **SSD Disabled during Wait Mode** — When entering Wait Mode, this bit provides on/off control over the SSD allowing reduced MCU power consumption. Because the analog circuit is turned off when powered down, the sigma-delta converter requires a recovery time after exit from Wait Mode.<br>0 SSD continues to run in WAIT mode.<br>1 Entering WAIT mode freezes the clock to the prescaler divider, powers down the sigma-delta converter, and if RTZE bit is set, the sine and cosine coils are recirculated via VSSM. |
| 4<br>FTST | **Factory Test** — This bit is reserved for factory test and reads zero in user mode. |
| 1:0<br>ACLKS | **Accumulator Sample Frequency Select** — This field sets the accumulator sample frequency by pre-scaling the bus frequency by a factor of 8, 16, 32, or 64. A faster sample frequency can provide more accurate results but cause the accumulator to overflow. Best results are achieved with a frequency between 500 kHz and 2 MHz.<br>Accumulator Sample Frequency = $f_{BUS}$ / (8 x $2^{ACLKS}$) |

**Table 17-10. Accumulator Sample Frequency**

| ACLKS | Frequency | $f_{BUS}$ = 40 MHz | $f_{BUS}$ = 25 MHz | $f_{BUS}$ = 16 MHz |
|---|---|---|---|---|
| 0 | $f_{BUS}$ / 8 | 5.00 MHz | 3.12 MHz | 2.00 MHz |
| 1 | $f_{BUS}$ / 16 | 2.50 MHz | 1.56 MHz | 1.00 MHz |
| 2 | $f_{BUS}$ / 32 | 1.25 MHz | 781 kHz | 500 kHz |
| 3 | $f_{BUS}$ / 64 | 625 kHz | 391 kHz | 250 kHz |

**NOTE**

A change in the accumulator sample frequency will not be effective until the ITG bit is cleared.

### 17.3.2.4 Stepper Stall Detector Flag Register (SSDFLG)

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MCZIF | 0 | 0 | 0 | 0 | 0 | 0 | AOVIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-5. Stepper Stall Detector Flag Register (SSDFLG)**

Read: anytime

Write: anytime.

**Table 17-11. SSDFLG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>MCZIF | **Modulus Counter Underflow Interrupt Flag** — This flag is set when the modulus down-counter reaches 0x0000. If not masked (MCZIE = 1), a modulus counter underflow interrupt is pending while this flag is set. This flag is cleared by writing a ‘1’ to the bit. A write of ‘0’ has no effect. |
| 0<br>AOVIF | **Accumulator Overflow Interrupt Flag** — This flag is set when the Integration Accumulator has a positive or negative overflow. If not masked (AOVIE = 1), an accumulator overflow interrupt is pending while this flag is set. This flag is cleared by writing a ‘1’ to the bit. A write of ‘0’ has no effect. |

### 17.3.2.5 Modulus Down-Counter Count Register (MDCCNT)

Module Base + 0x0004

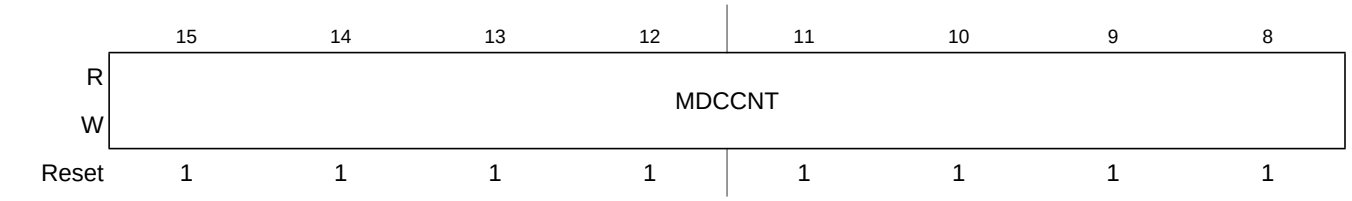



**Figure 17-6. Modulus Down-Counter Count Register High (MDCCNT)**

Module Base + 0x0005

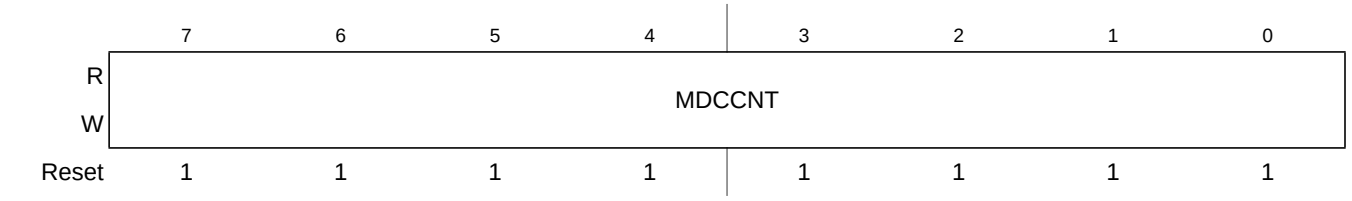



**Figure 17-7. Modulus Down-Counter Count Register Low (MDCCNT)**

Read: anytime

Write: anytime.

**NOTE**

A separate read/write for high byte and low byte gives a different result than accessing the register as a word.

If the RDMCL bit in the MDCCTL register is cleared, reads of the MDCCNT register will return the present value of the count register. If the RDMCL bit is set, reads of the MDCCNT register will return the contents of the load register.

With a 0x0000 write to the MDCCNT register, the modulus counter stays at zero and does not set the MCZIF flag in the SSDFLG register.

If modulus mode is not enabled (MODMC = 0), a write to the MDCCNT register immediately updates the load register and the counter register with the value written to it. The modulus counter will count down from this value and will stop at 0x0000.

If modulus mode is enabled (MODMC = 1), a write to the MDCCNT register updates the load register with the value written to it. The count register will not be updated with the new value until the next counter

underflow. The FLMC bit in the MDCCTL register can be used to immediately update the count register with the new value if an immediate load is desired.

The modulus down counter clock frequency is the bus frequency divided by 64 or 512.

### 17.3.2.6 Integration Accumulator Register (ITGACC)

Module Base + 0x0006

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | ITGACC | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-8. Integration Accumulator Register High (ITGACC)**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ITGACC | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-9. Integration Accumulator Register Low (ITGACC)**

Read: anytime.

Write: Never.

**NOTE**

A separate read for high byte and low byte gives a different result than accessing the register as a word.

This 16-bit field is signed and is represented in two's complement. It indicates the change in flux while integrating the back EMF present in the non-driven coil during a return to zero event.

When ITG is zero, the accumulator is initialized to 0x0000 and the sigma-delta converter is in a reset state.

When ITG is one, the accumulator increments or decrements depending on the sigma-delta conversion sample. The accumulator sample frequency is determined by the ACLKS field. The accumulator freezes at 0x7FFF on a positive overflow and freezes at 0x8000 on a negative overflow.

## 17.4 Functional Description

The stepper stall detector (SSD) has a simple control block to configure the H-bridge drivers of a stepper motor in four different full step states with four available modes during a return to zero event. The SSD has a detect circuit using a sigma-delta converter to measure and integrate changes in flux of the de-energized winding in the stepping motor and the conversion result is accumulated in a 16-bit signed register. The SSD also has a 16-bit modulus down counter to monitor blanking and integration times. DC offset compensation is implemented when using the modulus down counter to monitor integration times.

### 17.4.1 Return to Zero Modes

There are four return to zero modes as shown in Table 17-12.

**Table 17-12. Return to Zero Modes**

| ITG | DCOIL | Mode |
|---|---|---|
| 0 | 0 | Blanking with no drive |
| 0 | 1 | Blanking with drive |
| 1 | 0 | Conversion |
| 1 | 1 | Integration |

#### 17.4.1.1 Blanking with No Drive

In blanking mode with no drive, one of the coils is masked from the sigma-delta converter, and if RTZ is enabled (RTZE = 1), it is set up to recirculate its current. If RTZ is enabled (RTZE = 1), the other coil is disconnected to prevent any loss of flux change that would occur when the motor starts moving before the end of recirculation and start of integration. In blanking mode with no drive, the accumulator is initialized to 0x0000 and the converter is in a reset state.

#### 17.4.1.2 Blanking with Drive

In blanking mode with drive, one of the coils is masked from the sigma-delta converter, and if RTZ is enabled (RTZE = 1), it is set up to recirculate its current. If RTZ is enabled (RTZE = 1), the other coil is driven. In blanking mode with drive, the accumulator is initialized to 0x0000 and the converter is in a reset state.

#### 17.4.1.3 Conversion

In conversion mode, one of the coils is routed for integration with one end connected to the (non-zero) reference input and the other end connected to the integrator input of the sigma-delta converter. If RTZ is enabled (RTZE=1), both coils are disconnected. This mode is not useful for stall detection.

#### 17.4.1.4 Integration

In integration mode, one of the coils is routed for integration with one end connected to the (non-zero) reference input and the other end connected to the integrator input of the sigma-delta converter. If RTZ is enabled (RTZE = 1), the other coil is driven. This mode is used to rectify and integrate the back EMF produced by the coils to detect stepped rotary motion.

DC offset compensation is implemented when using the modulus down counter to monitor integration time.

### 17.4.2 Full Step States

During a return to zero (RTZ) event, the stepper motor pointer requires a 90° full motor electrical step with full amplitude pulses applied to each phase in turn. For counter clockwise rotation (CCW), the STEP value is incremented 0, 1, 2, 3, 0 and so on, and for a clockwise rotation the STEP value is decremented 3, 2, 1,

0 and so on. Figure 17-10 shows the current level through each coil for each full step in CCW rotation when DCOIL is set.

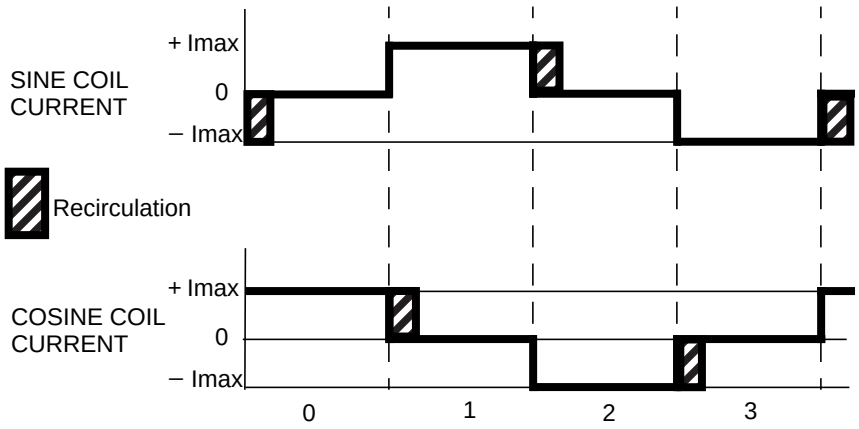



**Figure 17-10. Full Steps (CCW)**

Figure 17-11 shows the current flow in the SIN and COS H-bridges when STEP = 0, DCOIL = 1, ITG = 0 and RCIR = 0.

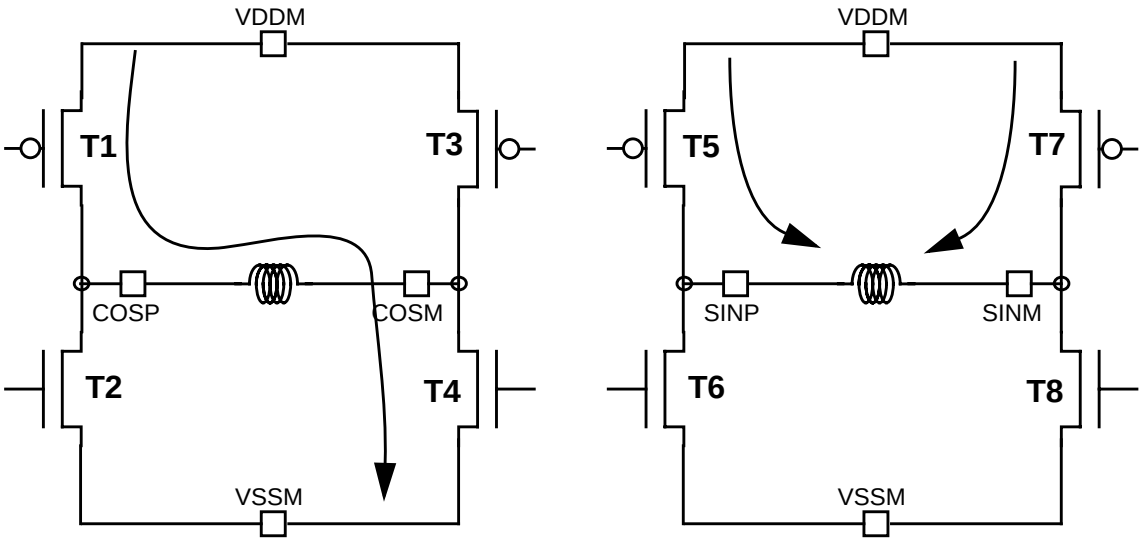



**Figure 17-11. Current Flow when STEP = 0, DCOIL = 1, ITG = 0, RCIR = 0**

Figure 17-12 shows the current flow in the SIN and COS H-bridges when STEP = 1, DCOIL = 1, ITG = 0 and RCIR = 1.

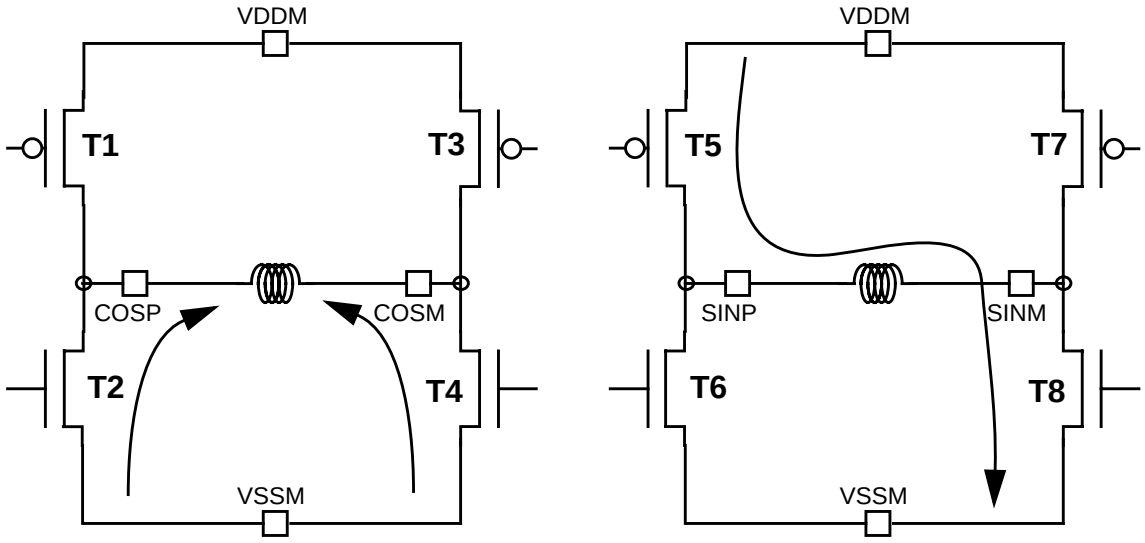



**Figure 17-12. Current Flow when STEP = 1, DCOIL = 1, ITG = 0, RCIR = 1**

Figure 17-13 shows the current flow in the SIN and COS H-bridges when STEP = 2, DCOIL = 1 and ITG = 1.

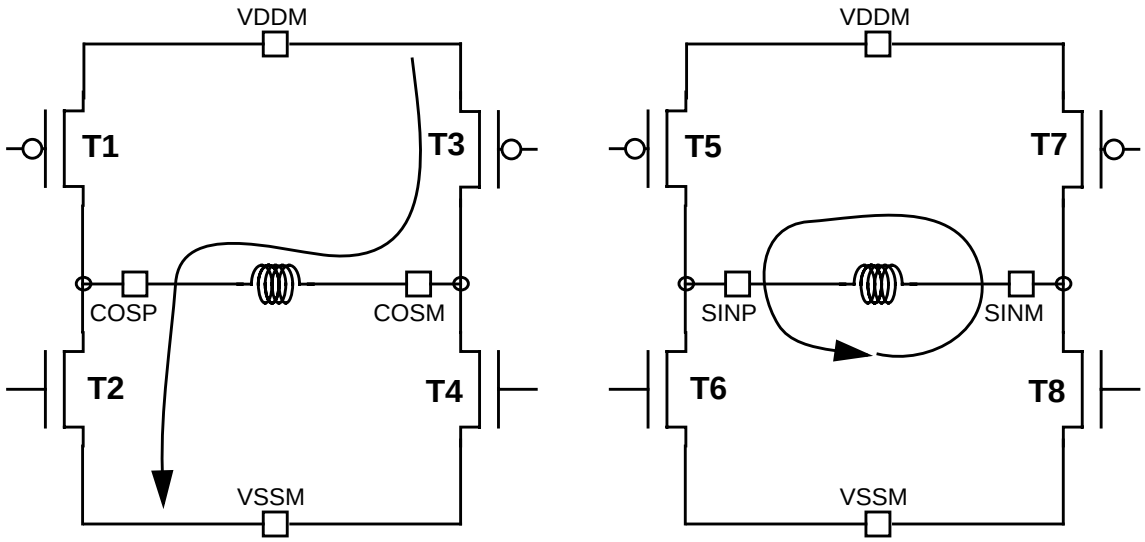



**Figure 17-13. Current flow when STEP = 2, DCOIL = 1, ITG = 1**

Figure 17-14 shows the current flow in the SIN and COS H-bridges when STEP = 3, DCOIL = 1 and ITG = 1.

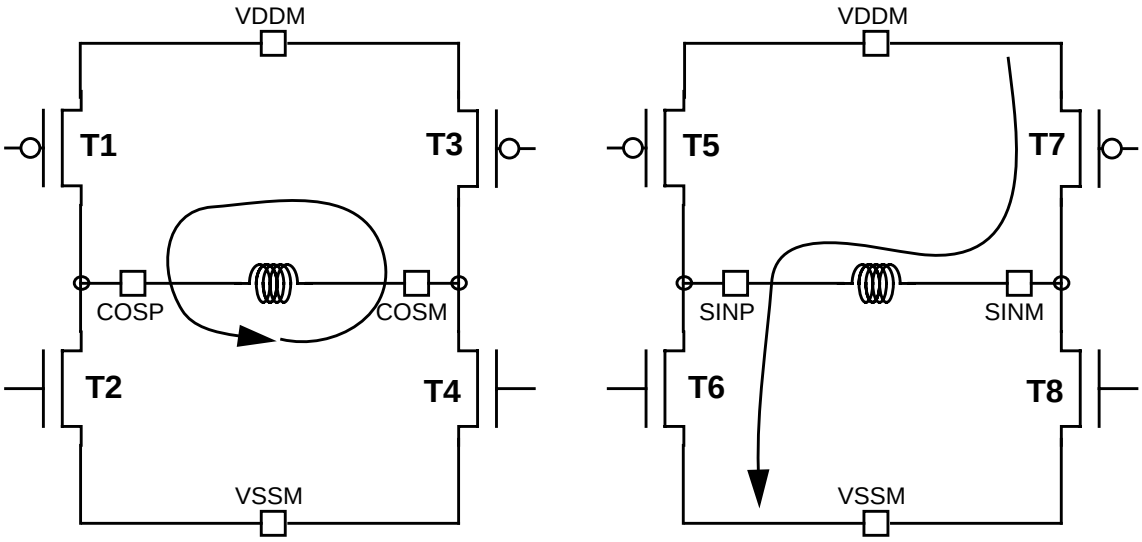



**Figure 17-14. Current flow when STEP = 3, DCOIL = 1, ITG = 1**

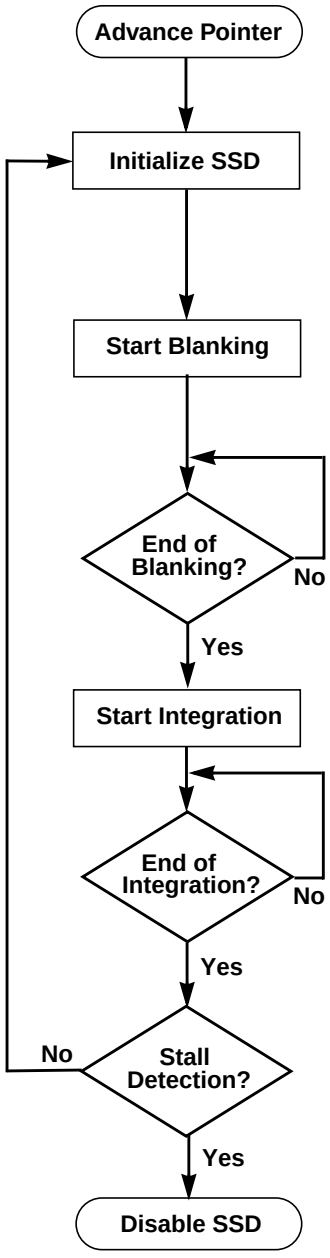
### 17.4.3 Operation in Low Power Modes

The SSD block can be configured for lower MCU power consumption in three different ways.

- Stop mode powers down the sigma-delta converter and halts clock to the modulus counter. Exit from Stop enables the sigma-delta converter and the clock to the modulus counter but due to the converter recovery time, the integration result should be ignored.
- Wait mode with SSDWAI bit set powers down the sigma-delta converter and halts the clock to the modulus counter. Exit from Wait enables the sigma-delta converter and clock to the modulus counter but due to the converter recovery time, the integration result should be ignored.
- Clearing SDCPU bit powers down the sigma-delta converter.

### 17.4.4 Stall Detection Flow

Figure 17-15 shows a flowchart and software setup for stall detection of a stepper motor.




| |
|---|
| Using Motor Control module, drive pointer to within 3 full steps of calibrated zero position. |
| 1. Clear (or set) RCIR; clear (or set) POL;<br>2. Set MCZIE; clear MODMC; clear (or set) PRE; set MCEN.<br>3. Set RTZE; set SDCPU; write ACLKS (select sample frequency).<br>4. Store threshold value in RAM. |
| 1. Clear MCZIF.<br>2. Write MDCCNT with blanking time value.<br>3. Clear ITG; clear (or set) DCOIL; increment (or decrement) STEP for CCW (or CW) motion. |
| MDCCNT = 0x0000? or MCZIF = 1? |
| 1. Clear MCZIF.<br>2. Write MDCCNT with integration time value.<br>3. Set ITG; set DCOIL. |
| MDCCNT = 0x0000? or MCZIF = 1? |
| ITGACC < Threshold (RAM value)? |
| 1. Clear MCZIF.<br>2. Clear MCEN.<br>3. Clear ITG.<br>4. Clear RTZE; clear SDCPU. |

**Figure 17-15. Return-to-Zero Flowchart**

# Chapter 18
# Real-Time Counter With Calendar (RTCV2)

**Table 18-1. Revision History**

| Version Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| 0.0 | April 2011 | All | • inital Draft |
| 0.7 | March 2011 | All | • fix typo on 18.1,18.2,18.3.4,18.5.3,18.5.6,18.6.1.1 |
| 0.8 | March 2011 | All | • V2 spec,add clock source of IRCCLK, Figure 18-1./18-635, 18.3.3/18-636, Table 18-2./18-636, Figure 18-3./18-638, Table 18-4./18-638 |
| 0.9 | March 2011 | All | • fix typo according review |

## 18.1 Introduction

The real-time counter (RTC) supports the calendar functions. It can also serve as a cyclical wake up from low power modes, STOP and WAITs, without the need of external components. A frequency compensation mechanism is built into this RTC module to allow adjustments made to the RTC clock when a less accurate crystal is used or to correct frequency shifts due to changing temperature. The 1-Hz clock that drives the clock and calendar could make use of the built-in compensation mechanism for crystal frequency error compensation so that the 1-Hz clock could be made more accurate than the frequency accuracy of the crystal that drive the module.

## 18.2 Features

Features of the RTC module include:

- 16-bit up-counter
- Support both 4-16MHz OSC(OSCCLCK) and 32.768KHz OSC(OSCCLK_32K) input
- Calendar Counter registers for:
  - — Second
  - — Minute
  - — Hour
- Five periodic interrupts
- Built-in compensation mechanism for frequency error compensation

### 18.2.1 Modes of Operation

This section defines the RTC operation in stop, wait and background debug modes.

#### 18.2.1.1 Wait Mode

The RTC continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the RTC can be used to bring the MCU out of wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during wait mode.

#### 18.2.1.2 Stop Modes

The RTC continues to run in STOP mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can be used to bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

#### 18.2.1.3 Freeze Mode

In freeze mode(BDM active), depend on the bit FRZ in RTCCTL3, RTC will be stop or run continue.

### 18.2.2 Block Diagram

The block diagram for the RTC module is shown in Figure 18-1.

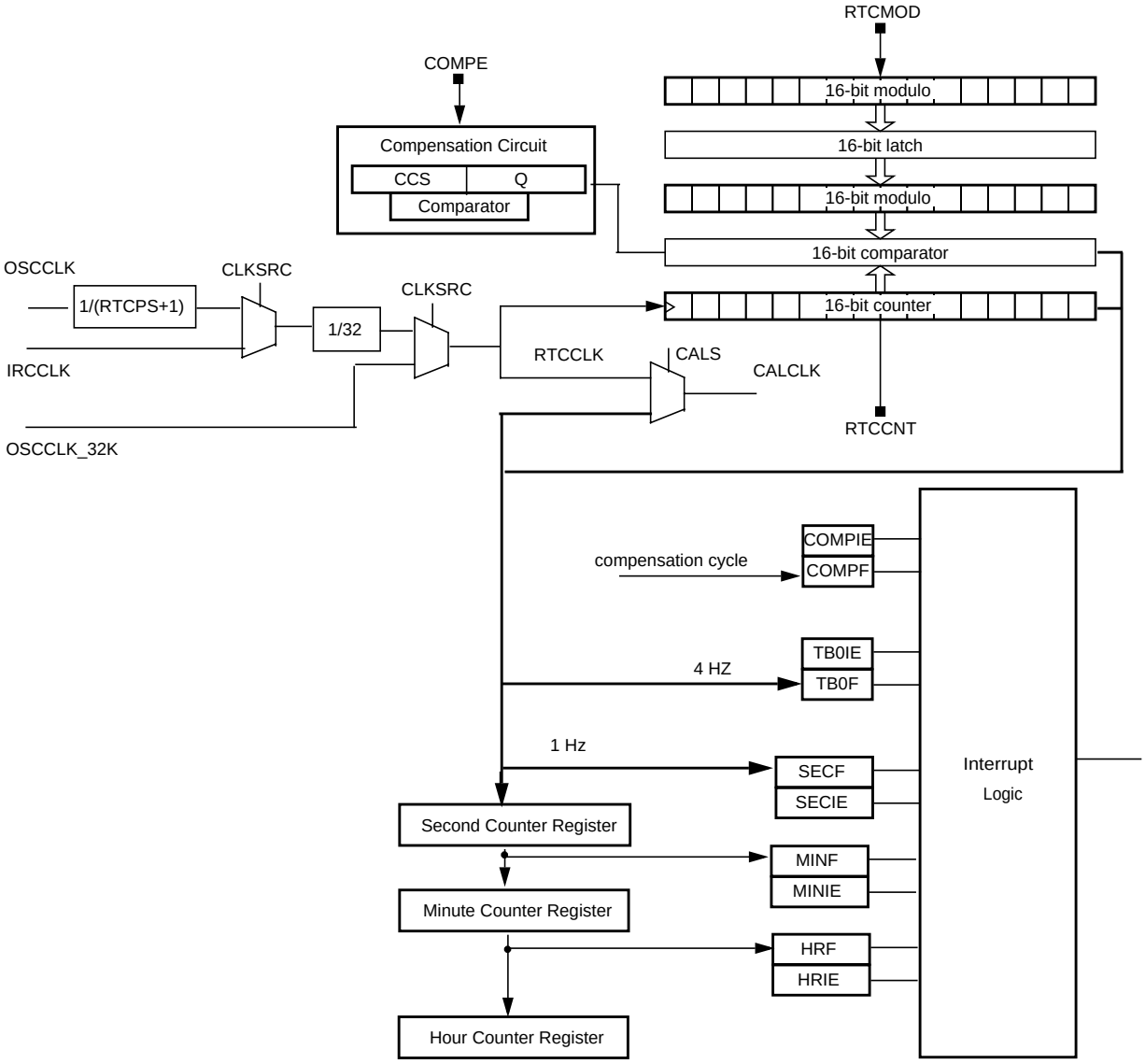



**Figure 18-1. Real-Time Counter with Calendar (RTC) Block Diagram**

## 18.3 External Signal Description

### 18.3.1 OSCCLK

The OSCCLK is 4-16MHZ main OSC output.

### 18.3.2 OSCCLK_32K

The OSCCLK_32K is 32.768K OSC output.

### 18.3.3 IRCCLK

The IRCCLK is 1MHz CPMU output.

### 18.3.4 RTCCLK

The RTCCLK can run from OSCCLK or OSCCLK_32K. It depends on the CLKSRC setting . If derived from OSCCLK, it depends on the RTCPS settting also. User need to set the correct RTCPS in order to get the right RTCCLK generated. This clock will be used as LCD clock input source also.

### 18.3.5 CALCLK

The CALCLK is calibration clock output. It is internal routed to the timer channel for the on chip calibration. Refer the PIM section for detailed information. And it is also routed to the pin, refer to the device specification for avaiablity and connectivity of signal.

## 18.4 Register Definition

The RTC includes status and control registers, a 16-bit counter register, and a 16-bit modulo register, calendar function counter registers. Except the RTCCTL3, RTCCTL4, RTCS1 registers which are normal reset, all the other RTC registers are POR reset only.

Table 18-2 is a summary of RTC registers.

**Table 18-2. RTC Register Summary**

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 RTCCTL1 | R | RTCEN | 0 | 0 | 0 | 0 | 0 | COMPE | 0 |
| | W | | | | | | | | |
| 0x0001 RTCCTL2 | R | CLKSRC 1 | CLKSRC 0 | 0 | 0 | RTCPS3 | RTCPS2 | RTCPS1 | RTCPS0 |
| | W | | | | | | | | |
| 0x0002 RTCCTL3 | R | | | 0 | 0 | FRZ | 0 | CALS | 0 |
| | W | RTCWE1 | RTCWE 0 | | | | | | |
| 0x0003 RTCCTL4 | R | 0 | 0 | HRIE | MINIE | SECIE | COMPIE | 0 | TB0IE |
| | W | | | | | | | | |
| 0x0004 RTCS1 | R | CDLC | 0 | HRF | MINF | SECF | COMPF | 0 | TB0F |
| | W | | | | | | | | |
| 0x0005 RTCCCR | R | CCS | | Q | | | | | |
| | W | | | | | | | | |

**Table 18-2. RTC Register Summary (continued)**

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0006 RTCMODH | R | RTCMODH | | | | | | | |
| | W | | | | | | | | |
| 0x0007 RTCMODL | R | RTCMODL | | | | | | | |
| | W | | | | | | | | |
| 0x0008 RTCCNTH | R | RTCCNTH | | | | | | | |
| | W | | | | | | | | |
| 0x0009 RTCCNTL | R | RTCCNTL | | | | | | | |
| | W | | | | | | | | |
| 0x000A - 0x000C RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000D RTCSECR | R | 0 | 0 | SEC5 | SEC4 | SEC3 | SEC2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x000E RTCMINR | R | 0 | 0 | MIN5 | MIN4 | MIN3 | MIN2 | MIN1 | MIN0 |
| | W | | | | | | | | |
| 0x000F RTCHRR | R | 0 | 0 | 0 | HR4 | HR3 | HR2 | HR1 | HR0 |
| | W | | | | | | | | |

## 18.4.1 RTC Control Register 1(RTCCTL1)

The RTCCTL1 contains the enable bit for RTC compensation function and RTC function enable bit. This register is write-protected, writing this register should follows write-protect mechanism section.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTCEN | 0 | 0 | 0 | 0 | 0 | COMPE | 0 |
| W | | | | | | | | |
| POR: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-2. RTC Control Register 1 (RTCCTL1)**

**Table 18-3. RTCCTL1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTCEN | **RTC Enable** — This read/write bit enables the RTC.<br>0 RTC function is disabled.<br>1 RTC function is enabled. |
| 1<br>COMPE | **RTC Compensation Enable**— The read/write bit enables the clock compensation mechanism for clock frequency errors.<br>0 Compensation mechanism is disabled.<br>1 Compensation mechanism is enabled. |

## 18.4.2 RTC Control Register 2 (RTCCTL2)

The RTCCTL2 contains the RTC clock source selection bit and clock divider control bits. This register is write one time only in normal mode, anytime in special mode.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CLKSRC1 | CLKSRC0 | 0 | 0 | RTCPS3 | RTCPS2 | RTCPS1 | RTCPS0 |
| W | | | | | | | | |
| POR: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-3. RTC Control Register 2(RTCCTL2)**

**Table 18-4. RTCCTL2 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>CLKSRC | **RTC Clock Source Selection**— The read/write bit select the RTCCLK source.<br>00 RTCCLK is from OSCCLK<br>01 RTCCLK is from OSCCLK_32K<br>10 RTCCLK is from IRCCLK<br>11 reserved |
| 3:0<br>RTCPS | **RTC main OSC prescaler** — These read/write bits define the OSCCLK prescaler value. The input OSCCLK will be divided by (RTCPS+1) if RTCLK source is from OSCCLK. User should set the RTCPS value to let prescaler clock frequency to be 1MHZ. The 1MHZ clock will be divide by 32 to generate 31.25KHz RTC counter clock. |

## 18.4.3 RTC Control Register 3 (RTCCTL3)

The RTCCTL3 contains the write protection configure bits (RTCWE1, RTCW0), the compensation data setting ready bit.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | | | 0 | 0 | FRZ | 0 | CALS | 0 |
| W | RTCWE1 | RTCWE0 | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-4. RTC Control Register 3(RTCCTL3)**

**Table 18-5. RTCCTL3 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>RTCWE[1:0] | **RTC Module Write Enable** — These two write-only bits control the write-protect function of several RTC registers and bits. After a reset, write-protect is disabled, allowing full write access to RTC registers and bits. These two bits always read as 0.<br>To enable write-protect, perform the following sequence:<br>1. Write %10 to RTCWE[1:0] bits<br>2. Write %10 to RTCWE[1:0] bits<br>To disable write-protect, perform the following sequence:<br>1. Write %00 to RTCWE[1:0] bits<br>2. Write %01 to RTCWE[1:0] bits<br>3. Write %11 to RTCWE[1:0] bits<br>4. Write %10 to RTCWE[1:0] bits<br>To disable write-protect from an unsure protection state, first perform the enable write-protect sequence, followed by the disable write protect sequence. |
| 3<br>FRZ | **RTC Stop in Freeze Mode**— The read/write bit decide the RTC behavioral during Freeze mode.<br>0 RTC run during freeze mode.<br>1 RTC stop during freeze mode. |
| 1<br>CALS | **RTC Calibration Clock Selection**— The read/write bit enables the clock calibration source selection.<br>0 RTCCLK is output to CALCLK.<br>1 1HZ compensation clock is output to CALCLK. |

## 18.4.4 RTC Control Register 4 (RTCCTL4)

The RTCCTL4 contains the interrupt enable bits for RTC interrupt functions.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | HRIE | MINIE | SECIE | COMPIE | 0 | TB0IE |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-5. RTC Control Register 4 (RTCCTL4)**

**Table 18-6. RTCCTL4 Field Descriptions**

| Field | Description |
|---|---|
| 5<br>HRIE | **Hour Interrupt Enable** — This read/write bit enables hour interrupts. If HRIE is set, then an interrupt is generated when HRF is set.<br>0 Hour interrupt request is disabled.<br>1 Hour interrupt request is enabled. |
| 4<br>MINIE | **Minute Interrupt Enable** — This read/write bit enables minute interrupts. If MINIE is set, then an interrupt is generated when MINF is set.<br>0 Minute interrupt request is disabled.<br>1 Minute interrupt request is enabled. |
| 3<br>SECIE | **Second Interrupt Enable** — This read/write bit enables second interrupts. If SECIE is set, then an interrupt is generated when SECF is set.<br>0 Second interrupt request is disabled.<br>1 Second interrupt request is enabled. |
| 2<br>COMPIE | **Compensation cycle Interrupt Enable** — This read/write bit enables compensation cycle interrupts. If COMPIE is set, then an interrupt is generated when COMPF is set.<br>0 Compensation cycle interrupt request is disabled.<br>1 Compensation cycle interrupt request is enabled. |
| 0<br>TB0IE | **4 HZ Time Tick interrupt Enable** — This read/write bit enables 4 Hz time ticket interrupt. If TB0IE is set, then an interrupt is generated when TB0IF is set.<br>0 4 Hz time ticket interrupt request is disabled.<br>1 4 Hz time ticket interrupt request is enabled. |

## 18.4.5 RTC Status Register 1 (RTCS1)

The RTCS1 status register contains six interrupt status flags. When a flag is set and the corresponding interrupt enable bit is also set, a CPU interrupt request is generated.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CDLC | 0 | HRF | MINF | SECF | COMPF | 0 | TB0F |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-6. RTC Status and Control Register1 (RTCSC1)**

**Table 18-7. RTCS1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CDLC | **Compensation Data Load Cycle** — This status bit is set automatically when compensation circuit start load RTCMOD and RTCCCR to internal buffered register and cleared automatically when finished. If this bit is set, write to RTCMOD and RTCCCR is blocked. Write to this bit has no effect.<br>0 write to RTCMOD or RTCCCR is allowed.<br>1 write to RTCMOD and RTCCR is blocked. |
| 5<br>HRF | **Hour Flag** — This status bit is set on every increment of the hour counter. When the HRIE bit is set, HRF generates a CPU interrupt request. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the hour interrupt request. Reset clears HRF to 0.<br>0 No hour counter increment.<br>1 Hour counter increment. |
| 4<br>MINF | **Minute Flag** — This status bit is set on every increment of the minute counter. When the MINIE bit is set, MINF generates a CPU interrupt request. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the minute interrupt request. Reset clears MINF to 0.<br>0 No minute counter increment.<br>1 Minute counter increment. |
| 3<br>SECF | **Second Flag** — This status bit is set on every increment of the second counter. When the SECIE bit is set, SECF generates a CPU interrupt request. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the second interrupt request. Reset clears SECF to 0.<br>0 No second counter increment.<br>1 Second counter increment. |
| 2<br>COMPF | **Compensation cycle Flag** — This status bit is set on every last second of compensation cycle. When the COMPIE bit is set, COMPF generates a CPU interrupt request. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the time base interrupt request. Reset clears COMPF to 0.<br>0 No Compensation cycle has occurred.<br>1 A compensation cycle has occurred. |
| 0<br>TB0F | **4 Hz time tick Flag** — This status bit is set on every 4 Hz time ticket. When the TB0IE bit is set, TB0F generates a CPU interrupt request. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the time base interrupt request. Reset clears TB0F to 0.<br>0 No 4 Hz time tick has occurred.<br>1 A 4 Hz time tick has occurred. |

## 18.4.6 RTC Compensation Configure Register (RTCCCR)

This register includes the CCS and Q value for free run 16-bit counter match times with M and M+1 modulo value during compensation.See compensation function section for detail.

**Figure 18-7. RTC Compensation Configure Register (RTCCCR)**

**Table 18-8. RTCCCR Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>CCS | **Compensation cycle selection** — When the clock compensation mechanism enabled, It will decide the compensation cycle.<br>00 -- 5 second compensation cycle<br>01 -- 15 second compensation cycle<br>10 -- 30 second compensation cycle<br>11 -- 60 second compensation cycle |
| 5:0<br>Q | **16-bit Counter Matches with M+1 Modulo Value Times** — When the clock compensation mechanism enabled, the free run timer counter matches with M+1 modulo value which set in RTC modulo register (RTCMOD) Q times first, then matches with M modulo value with reset compensation cycle time. When Q is 0, free run timer counter matches with M+1 modulo value do not happen. Refer to Section Table 18-14., “the CCS, precision and Q value” |

## 18.4.7 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 16-bit counter.

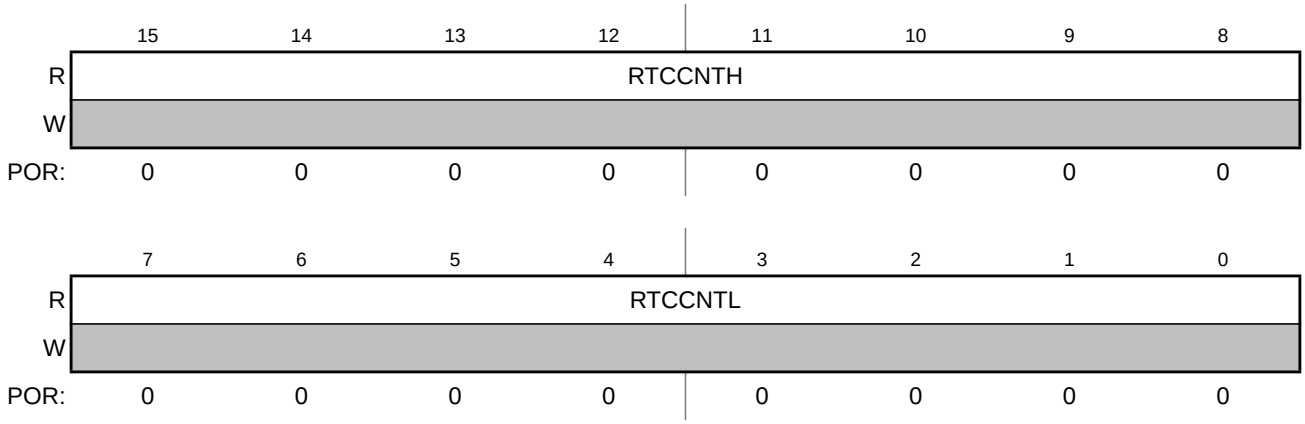



**Figure 18-8. RTC Counter Register (RTCCNT)**

**Table 18-9. RTCCNT Field Description**

| Field | Description |
|---|---|
| 15:0<br>RTCCNT | **RTC Count**— These sixteen read-only bits contain the current value of the 16-bit counter. Writes have no effect to this register. Reset will clear the count to 0x0000. Note, the RTCCNT is not synced, so user need to read register two times to justify if value is correct, two read data same means the correct read data. |

## 18.4.8 RTC Modulo Register (RTCMOD)

RTCMOD is the 16-bit modulo value.

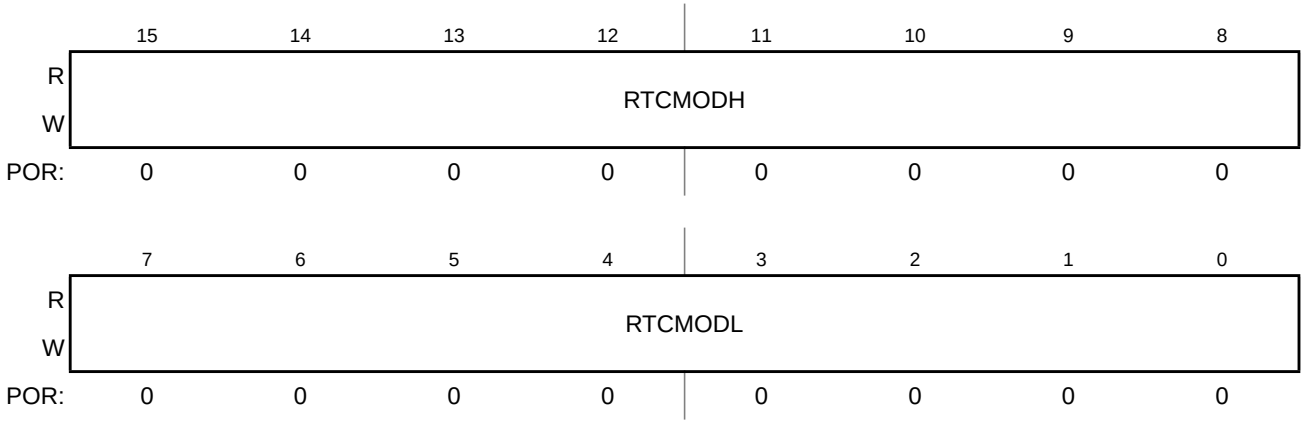



**Figure 18-9. RTC Modulo Register (RTCMOD)**

**Table 18-10. RTCMOD Field Descriptions**

| Field | Description |
|---|---|
| 15:0 RTCMOD | **RTC Modulo** — These sixteen read/write bits contain the modulo value used to reset the RTCCNT count to 0x0000. Reset sets the modulo to 0x0000. |

## 18.4.9 RTC Second Register (RTCSECR)

This read/write register contains the current value of the second counter. This register can be read at any time without affecting the counter count. Writing to this register loads the value to the second counter and the counter continues to count from this new value.Writing values other than 0 to 59 to this register has no effect. This register is write-protected, writing this register should follow write-protect mechanism section.

The second counter rolls over to 0 after reaching 59.

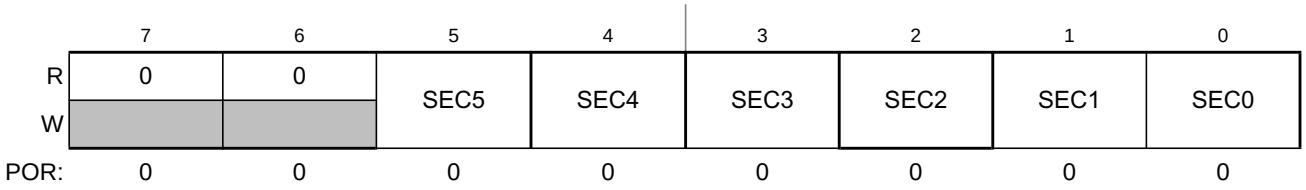



**Figure 18-10. Second Register (SECR)**

**Table 18-11. SECR Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>SEC5~SEC0 | **Second Counter Value** — These read/write bits contains the current value of the second. 0 to 59 is valid. Wirting the value other than 0 to 59 has no effect. |

## 18.4.10 RTC Minute Register (RTCMINR)

This read/write register contains the current value of the minute counter. This register can be read at any time without affecting the counter count. Writing to this register loads the value to the minute counter and the counter continues to count from this new value. Writing values other than 0 to 59 to this register has no effect. This register is write-protected, writing this register should follow write-protect mechanism section.

The second counter rolls over to 0 after reaching 59.

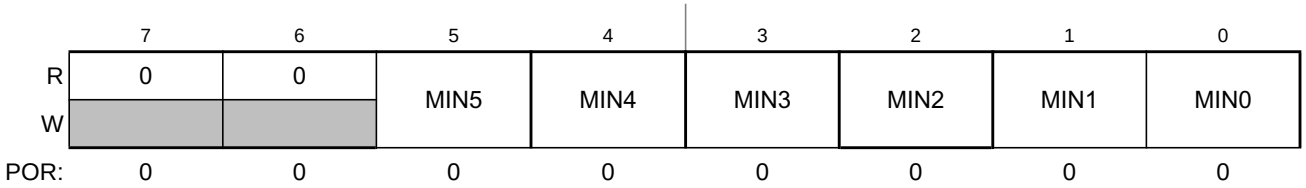



**Figure 18-11. Minute Register (MINR)**

**Table 18-12. MINR Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>MIN5~MIN0 | **Minute Counter Value** — These read/write bits contains the current value of the minute. 0 to 59 is valid.Wirting the value other than 0 to 59 has no effect. |

## 18.4.11 RTC Hour Register (RTCHRR)

This read/write register contains the current value of the hour counter. This register can be read at any time without affecting the counter count. Writing to this register loads the value to the hour counter and the counter continues to count from this new value.Writing values other than 0 to 23 to this register has no effect. This register is write-protected, writing this register should follow write-protect mechanism section.

The second counter rolls over to 0 after reaching 23.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | HR4 | HR3 | HR2 | HR1 | HR0 |
| W | | | | | | | | |
| POR: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-12. Hour Register (HRR)**

**Table 18-13. HRR Field Descriptions**

| Field | Description |
|---|---|
| 4:0<br>HR4~HR0 | **Hour Counter Value** — These read/write bits contains the current value of the hour. 0 to 23 is valid.Wirting the value other than 0 to 23 has no effect. |

## 18.5 Functional Description

The RTC module support calendar functions. It includes a main 16-bit up-counter with a 16-bit modulo register. The module also contains software selectable interrupt logic and clock output for compensation calibration.

The RTC module provides clock indications in seconds, minutes, hours. Reading the calendar registers return the current time. Writing to these registers set the time, and the counters will continue to count from the new settings.

### 18.5.1 RTC clock and reset behavior

Depending on the use case, the RTC clock can derived from OSCCLK or OSCCLK_32K. After power on reset, the RTC clock is derived from OSCCLK. User should set the RTCPS and CLKSRC value before enabling the RTCEN. If RTCCLK is from OSCCLK, the RTCPS value should be set to meet that the prescaler clock frequency will be 1MHz. The 1MHz clock will follow by a 1/32 divider to generate a 31.25KHz clock, which will be used as RTC counter clock(RTCCNT clock input). In case the clock source is OSCCLK, after reset, the main OSC will be disabled and must be enabled by setting the OSCE in CPMU as soon as possible in order to have clock available for the RTC. If clock source is OSCCLK_32K, software must wait the OSC startup time after power on reset before enabling the RTC. If 32K OSC is enabled, it will be always on until a power on reset happens.

### 18.5.2 Calendar Functions

Calendar functions are provided by the second, minute and hour count registers. The second, minute and hour counters are able to generate interrupts on every count increment, providing periodic interrupts for the second (RTCSECF), minute (RTCMINF) and hour (RTCHRF). A CPU interrupt request is generated if the corresponding enable bit (SECIE, MINIE and HRIE) is also set.

### 18.5.3 Interrupts

In addition to the second, minute and hour periodic interrupts generated by the clock functions, the divider circuits generates compensation cycle periodic interrupt and 4 Hz periodic interrupt. Depend on

the CCS setting, the cycle period can be 5, 15, 30, 60 seconds. The COMPIE bit in the RTC control register 4(RTCCTL4) can enable compensation cycle interrupt. When the compensation logic start at the last second compensation cycle, it will generate the interrupt. And user can calculate the next compensation value and update these registers.

When a 4 Hz time base tick happens, it indicated by TB0F flags. And a CPU interrupt request is generated if the corresponding interrupt enable is also set.

### 18.5.4 RTC Clock Compensation

The RTC module is built with compensation circuit. The circuit is used to compensate frequency errors of the input RTC clock, so that a more accurate 1-Hz RTC clock output can be achieved. Using this method, it is possible to use a less expensive crystal. If the crystal temperature profile and the current temperature is known, the compensation circuit can also be used to compensate crystal frequency errors due to ambient temperature change and crystal aging.

The compensation logic use the RTCMOD, CCS and Q of RTCCCR to decide the free run timer counter. User first need to decide the CCS, which determines the compensation precision. The larger CCS value, higher the precision, longer the compensation period. During each compensation cycle, the free run timer counter will first count RTCMOD+1 for Q times , then count RTCMOD for the “compensation peirod-Q” times. Note if Q >= compensation period, it means that free run timer counter will count RTCMOD+1 for “compensation period” times

The RTCMOD is the integral portion of actual frequency. Q value can be calculated base on Section Table 18-14., “the CCS, precision and Q value”.

**Table 18-14. the CCS, precision and Q value**

| CCS | Compensation period (second) | Compensation Precision | Q |
|---|---|---|---|
| 0 | 5 | 1/(2 * 5 * RTCMOD)[1] | integer of ((fraction value+1/(2*5)) * 5) |
| 1 | 15 | 1/(2 * 15 * RTCMOD)[1] | integer of ((fraction value+1/(2*15)) * 15) |
| 2 | 30 | 1/(2 * 30 * RTCMOD)[1] | integer of ((fraction value+1/(2*30)) * 30) |
| 3 | 60 | 1/(2 * 60 * RTCMOD)[1] | integer of ((fraction value+1/(2*60)) * 60) |

[1] the RTCCLK has only two possible input frequency, 31.25KHz or 32.768KHz. If RTCMOD value is set as requirement, user can use 31250 or 32768 to replace the RTCMOD value here to easy the compensation precision calculation.

First example, if the RTC clock is from OSCCLK_32K and it has 2000PPM frequence drift due to the crystal and temperature etc factors, the RTCCLK frequency will be 32833.536HZ. We set RTCMOD value to 32833 first. The fraction value will be 0.536, and we can get CCS=0, Q=3 or CCS=1, Q=8 or CCS=2, Q=16 or CCS=3, Q=32.

Second example, if the RTC clock is from OSCLK and it has -2000PPM frequence drift due to the crystal and temperature etc factors, the RTCPS is set to 15, the RTCCLK frequency will be 31187.5HZ. We set RTCMOD value to 31187 first. The fraction value will be 0.5, and we can get CCS=0, Q=3 or CCS=1, Q=8 or CCS=2, Q=15 or CCS=3, Q=30.

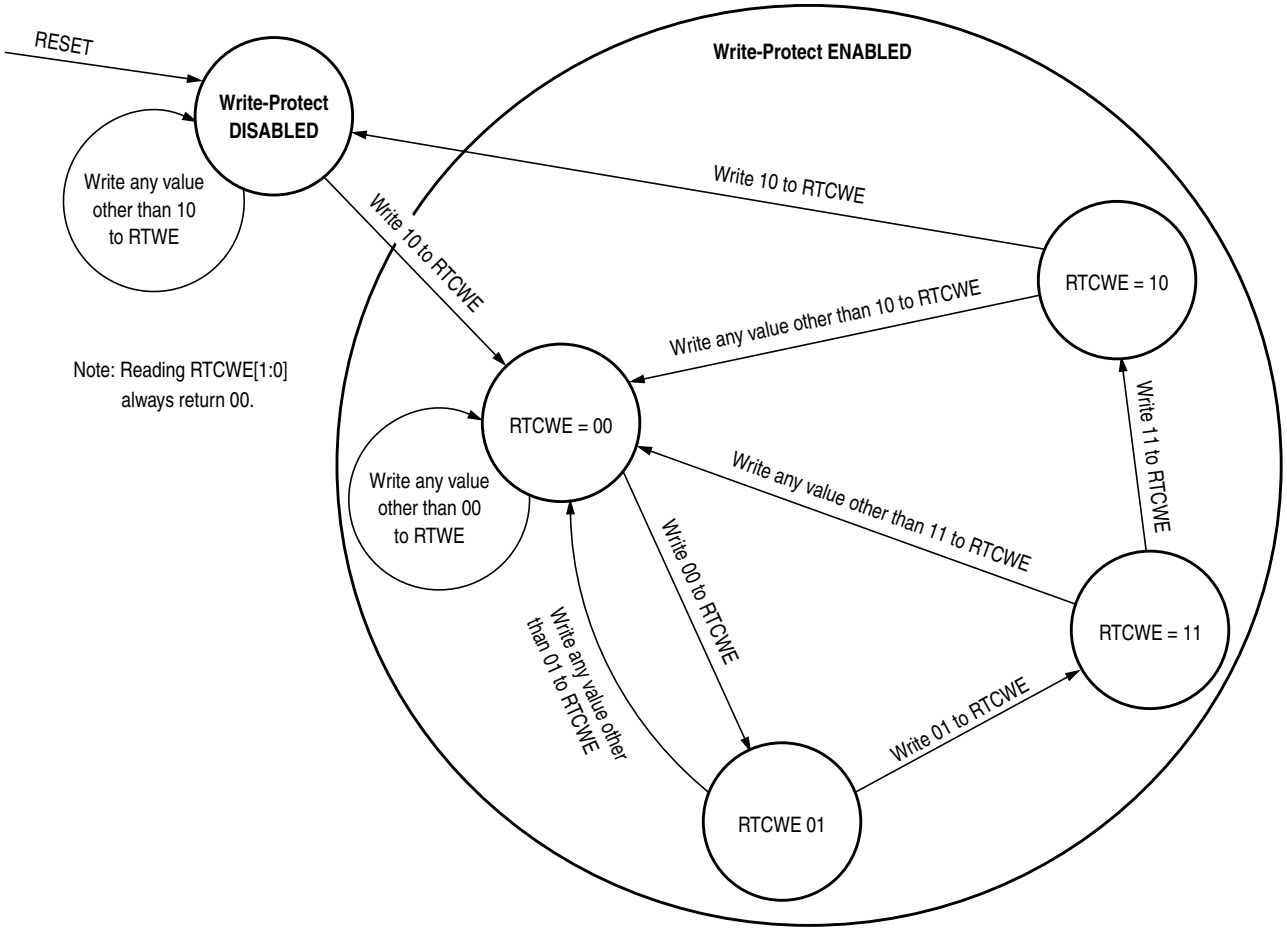
## 18.5.5 Calendar Register and Bit Write Protection

A write-protect mechanism is implemented to prevent accidental writes to the RTC clock registers, calendar registers, and other control bits. The protected RTC registers and bits are listed in below table

**Table 18-15. Write-Protected RTC Registers and Bits**

| Register | Bit |
|---|---|
| RTC Control Register 1(RTCCTL1) | All Bits |
| Second Register(RTCSECR) | All Bits |
| Minute Register(RTCMINR) | All Bits |
| Hour Register(RTCHRR) | All Bits |

The mechanism uses the RTCWE[1:0] bits in the RTC control register 3(RTCCTL3) in a state machine, which requires a bit-write sequence to disable the write-protection. A block diagram of the state machine is shown in below Figure.




After a reset, the write-protect mechanism is disabled, allowing the user set the time and date in the calendar registers.

### 18.5.6 Load buffer register

Compensation circuit and modulo circiuit using buffered register, those two register are loaded from RTCCCR and RTCMOD only when compensation completed.Write to RTCMOD and RTCCCR is blocked when CDLC=1, to protect data is stable when load occurs.

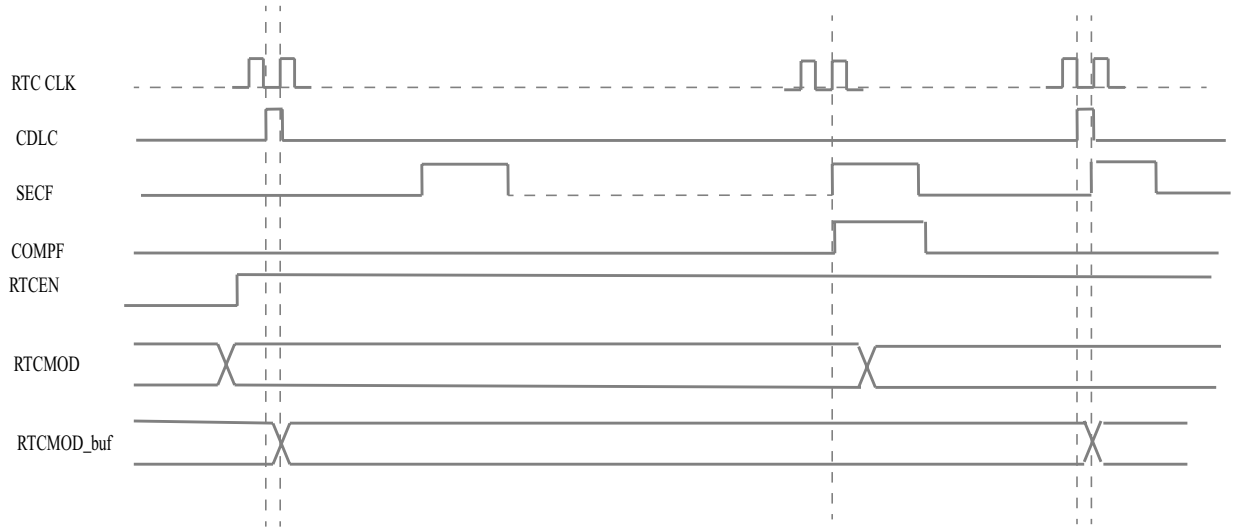



**Figure 18-13. Buffered register load sequence and CDLC**

## 18.6 Initialization/Application Information

### 18.6.1 RTC Calibration

#### 18.6.1.1 ThRev. 1.00e off chip calibration

The RTC output clock (CALCLK) can be output on the RTC_CAL pin. User can measure the clcok frequency directly. Refer to Device spec on how to enable the CALCLK on RTC_CAL pin.

#### 18.6.1.2 The on chip calibration

The CALCLK signal can be routed to an internal timer channel for on-chip measurement. At same time, user need feed one high precision 1 Hz clock to the other timer input pin. The calibration software can then take simultaneous measurements of both signals using the input capture function of the timer to calculate the variation in the RTC clock output and determine the correct calibration point. Refer to Device spec for more information on timer channel connection.

### 18.6.2 RTC compensation

To reach the high precision RTC clock, the compensation need following steps.

- Characterization of the OSCCLK or OSCCLK_32K crystal, get the lookup table for the crystal frequency versus to temperature. This step is just required one time per crystal type.
- Perform the RTC calibration at production for each device, get the base frequency offset.
- Perform the RTC compensation periodically after enabling the RTC.
  - Measure the external temperature
  - Lookup compensation value in lookup table
  - Load the compensation value into RTC

# Chapter 19
# Simple Sound Generator (SSGV1)

**Table 19-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V1.0 | Apr. 20, 2011 | All | Initial revision of Sound Generator Module. |

## 19.1 Introduction

This document describes the Simple Sound Generator module.

The SSG module generates audio frequency tone with autonomous amplitude control.

Refer to Figure 19-1 for the detailed block diagram of the module.

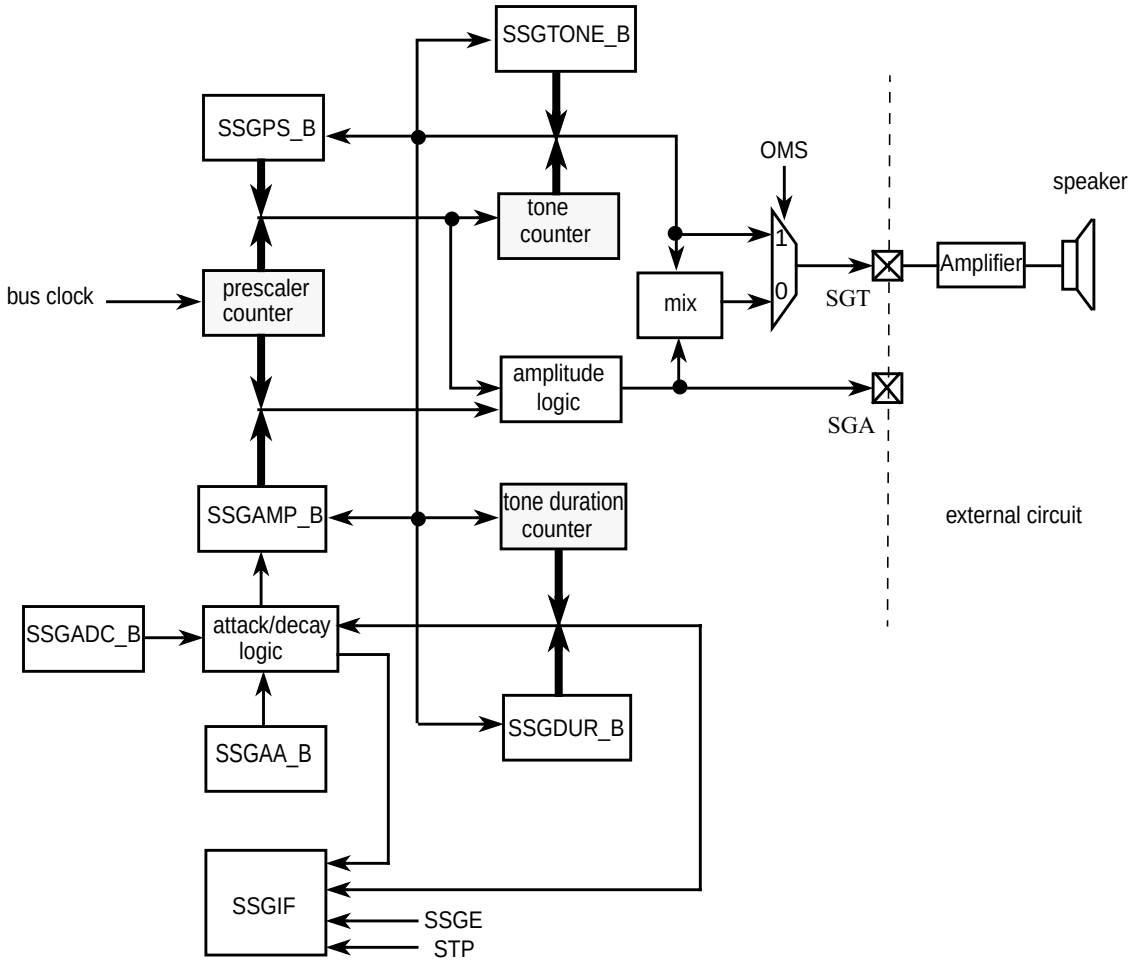
### 19.1.1 Features

The SSG block includes these distinctive features:

- Programmable amplitude level encoded with 11 bit resolution from zero amplitude to max amplitude
- Sound STOP function to stop sound generation immediately
- Registers double-buffered synchronously reload at edge of tone to avoid distortion of output tone
- Interrupt generates when SSG is ready to configure new sound data
- Input clock prescaler with 11 bit resolution
- Module disable for power saving when not in use
- Separate or mixed frequency and amplitude outputs for flexibility in external hardware variations
- Attack/decay function which can increase/decrease sound amplitude automatically without CPU interaction, including linear, gong and exponential attack/decay operation

### 19.1.2 Block Diagram

Figure 19-1 shows the block diagram for SSG block.




NOTE: XXX_B is the buffer register of relevant XXX register

The external circuit diagram is just for the OMS=0 case

External amplifier is required due to the limited driving capability of SGT pad

**Figure 19-1. SSG Block Diagram**

## 19.2 External Signal Description

### 19.2.1 SGT

Tone output signal of SSG, which can be a single output tone signal or a tone combined with amplitude encoding.

### 19.2.2 SGA

Encoded amplitude output signal of SSG.

## 19.3 Memory Map and Register Definition

### 19.3.1 Module Memory Map

This section describes the content of the registers in the SSG module. The base address of the SSG module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset. The figure below shows the registers associated with the SSG and their relative offset from the base address. The register detail description follows the order they appear in the register map.

Reserved bits within a register will always read as 0 and the write will be unimplemented. Unimplemented functions are indicated by shading the bit.

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

### 19.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the SSG module.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 SSGCR | R | SSGE | 0 | 0 | 0 | 0 | OMS | RDR | STP |
| | W | | | | | | | | |
| 0x0001 SSGADC | R | ADE | 0 | 0 | 0 | 0 | ADM[1:0] | | ADS |
| | W | | | | | | | | |
| 0x0002 SSGPSH | R | 0 | 0 | 0 | 0 | 0 | PS10 | PS9 | PS8 |
| | W | | | | | | | | |
| 0x0003 SSGPSL | R | PS7 | PS6 | PS5 | PS4 | PS3 | PS2 | PS1 | PS0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 19-2. The SSG Register Summary (Sheet 1 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0004 SSGTONEH | R | 0 | 0 | 0 | 0 | 0 | 0 | TONE9 | TONE8 |
| | W | | | | | | | | |
| 0x0005 SSGTONEL | R | TONE7 | TONE6 | TONE5 | TONE4 | TONE3 | TONE2 | TONE1 | TONE0 |
| | W | | | | | | | | |
| 0x0006 SSGAMPH | R | 0 | 0 | 0 | 0 | 0 | AMP10 | AMP9 | AMP8 |
| | W | | | | | | | | |
| 0x0007 SSGAMPL | R | AMP7 | AMP6 | AMP5 | AMP4 | AMP3 | AMP2 | AMP1 | AMP0 |
| | W | | | | | | | | |
| 0x0008 SSGAAH | R | 0 | 0 | 0 | 0 | 0 | AA10 | AA9 | AA8 |
| | W | | | | | | | | |
| 0x0009 SSGAAL | R | AA7 | AA6 | AA5 | AA4 | AA3 | AA2 | AA1 | AA0 |
| | W | | | | | | | | |
| 0x000A SSGATH | R | 0 | 0 | 0 | 0 | 0 | AT10 | AT9 | AT8 |
| | W | | | | | | | | |
| 0x000B SSGATL | R | AT7 | AT6 | AT5 | AT4 | AT3 | AT2 | AT1 | AT0 |
| | W | | | | | | | | |
| 0x000C SSGDUR | R | DUR7 | DUR6 | DUR5 | DUR4 | DUR3 | DUR2 | DUR1 | DUR0 |
| | W | | | | | | | | |
| 0x000D SSGIE | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RNDIE |
| | W | | | | | | | | |
| 0x000E SSGIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RNDI |
| | W | | | | | | | | |
| 0x000F RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0010 SSGAMPBH | R | 0 | 0 | 0 | 0 | 0 | AMPB10 | AMPB9 | AMPB8 |
| | W | | | | | | | | |
| 0x0011 SSGAMPBL | R | AMPB7 | AMPB6 | AMPB5 | AMPB4 | AMPB3 | AMPB2 | AMP1 | AMPB0 |
| | W | | | | | | | | |
| 0x0012 SSGDCNT | R | DCNT7 | DCNT6 | DCNT5 | DCNT4 | DCNT3 | DCNT2 | DCNT1 | DCNT0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 19-2. The SSG Register Summary (Sheet 2 of 3)**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0013 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0014 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0015 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0016 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0017 RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 19-2. The SSG Register Summary (Sheet 3 of 3)**

### 19.3.2.1 SSG Control Register (SSGCR)

The control register SSGCR contains bits to control module enable, SSG stop control, register data ready status and the SGT output mode selection.

Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SSGE | 0 | 0 | 0 | 0 | OMS | RDR | STP |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-3. SSG Control Register (SSGCR)**

Read: Anytime

Write: Anytime

**Table 19-2. SSGCR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SSGE | **SSG Enable**<br>0 SSG is disabled. All counters will be reset to 0, input clock will be gated.<br>1 SSG is enabled. |
| 2<br>OMS | **SSG Output Mode Selection**<br>0 SGT output tone mixed with amplitude.<br>1 SGT output tone solely. |

**Table 19-2. SSGCR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>RDR | **SSG Register Data Ready —**<br>This bit indicates whether the registers data are ready to reload into the relative buffer registers. Writing any of SSGPS/SSGTONE/SSGAMP/SSGDUR/SSGAA/SSGAT/SSGADC will clear this bit. User should write 1 to this bit if registers setting has been done. Writing 0 to this bit will stop the SSG when the synchronous reload event occurs.<br>0 Register data is not ready to load into buffer register.<br>1 Register data is ready to load into buffer register. |
| 0<br>STP | **SSG STOP**<br>0 SSG runs normally.<br>1 Immediately stop SSG. |

### 19.3.2.2 SSG attack/decay control register (SSGADC)

SSGADC control the attack/decay enable and attack/decay function selection.

Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ADE | 0 | 0 | 0 | 0 | ADM[1:0] | | ADS |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-4. SSG attack/decay control register (SSGADC)**

Read: Anytime

Write: Anytime

**Table 19-3. SSGADC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADE | **SSG attack/decay enable**<br>0 SSG attack/decay function disabled.<br>1 SSG attack/decay function enabled. |
| 2 - 1<br>ADM[1:0] | **SSG Linear, Gong or Exponential attack/decay Selection.**<br>refer to Section 19.4.3, “SSG Attack and Decay function for detail.<br>00 Select linear attack/decay operation.<br>01 Select gong attack/decay operation.<br>10 Select exponential attack/decay operation.<br>11 Reserved. |
| 0<br>ADS | **SSG Decay or attack Selection**<br>0 Select amplitude attack function.<br>1 Select amplitude decay function. |

### 19.3.2.3 SSG Clock Prescaler Register (SSGPS)

SSGPS is a 11 bit prescaler register. For the 32MHz input bus clock source, the frequency range of encoding amplitude waveform is from 15.625KHz to 125KHz, so the divided clock frequency is as Table 19-5.

Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | PS10 | PS9 | PS8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-5. SSG Clock Prescaler Register (SSGPSH)**

Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PS7 | PS6 | PS5 | PS4 | PS3 | PS2 | PS1 | PS0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-6. SSG Clock Prescaler Register (SSGPSL)**

Read: Anytime

Write: Anytime

**Table 19-4. SSGPS Field Descriptions**

| Field | Description |
|---|---|
| 10–0<br>PS[10:0] | **SSG Clock Prescaler Register Bits —**<br>The input clock will be divided by SSGPS+1. This divided signal will be used as tone counter clock. Also it will decide the amplitude of the PWM cycle period. |

**Table 19-5. Prescaler Clock Divider**

| Divided Clock Frequency From 32MHz Clock Source | Divide Ratio | SSGPS[10:0] |
|---|---|---|
| 125KHz | 256 | 0xFF |
| 124.514KHz | 257 | 0x100 |
| 124.031KHz | 258 | 0x101 |
| 123.552KHz~15.633KHz | 259~2047 | 0x102~0x7FE |
| 15.625KHz | 2048 | 0x7FF |

### 19.3.2.4 SSG Tone Register (SSGTONE)

SSGTONE is SSG tone register. The tone frequency range is between 100Hz to 8KHz. Refer to Table 19-7 for the tone frequency divide ratio and possible tone frequency.

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | TONE9 | TONE8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-7. SSG Tone Register (SSGTONEH)**

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TONE7 | TONE6 | TONE5 | TONE4 | TONE3 | TONE2 | TONE1 | TONE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-8. SSG Tone Register (SSGTONEL)**

Read: Anytime

Write: Anytime

**Table 19-6. SSGPS Field Descriptions**

| Field | Description |
|---|---|
| 9–0<br>TONE[9:0] | **SSG Tone Register Bits —**<br>The tone frequency is equal to prescaler clock / (2*(SSGTONE+1)). |

**Table 19-7.** Tone Frequency configuration

| Prescaler Clock Frequency | Tone Frequency Range | SSGTONE[9:0] | Adjacent Tone Frequency Gap Range |
|---|---|---|---|
| 15KHz | 100Hz~7.5KHz | 0x4A~0x0 | 1.351Hz~3750Hz |
| 30KHz | 100Hz~7.5KHz | 0x95~0x1 | 0.671Hz~2500Hz |
| 60KHz | 100Hz~7.5KHz | 0x12B~0x3 | 0.334Hz~1500Hz |
| 90KHz | 100Hz~7.5KHz | 0x1C1~0x5 | 0.223~1072Hz |
| 125KHz | 100Hz~7.813KHz | 0x270~0x7 | 0.16Hz~1115Hz |

### 19.3.2.5 SSG Amplitude Register (SSGAMP)

SSGAMP register contains amplitude duty.

Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | AMP10 | AMP9 | AMP8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-9. SSG amplitude Register (SSGAMPH)**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AMP7 | AMP6 | AMP5 | AMP4 | AMP3 | AMP2 | AMP1 | AMP0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-10. SSG amplitude Register (SSGAMPL)**

Read: Anytime

Write: Anytime

**Table 19-8. SSGAMP Field Descriptions**

| Field | Description |
|---|---|
| 10–0<br>AMP[10:0] | **SSG Amplitude Register Bits —**<br>Amplitude duty cycle is SSGAMP/(SSGPS+1). When SSGAMP >= SSGPS+1, duty cycle will be 100%. |

**NOTE**

due to algorithm reason, in gong attack operation mode, amplitude should not be set to 0, otherwise amplitude will always be 0 and gong attack operation will never end.

### 19.3.2.6 SSG Amplitude Adjustment (SSGAA)

The SSGAA is amplitude adjustment register.

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | AA10 | AA9 | AA8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-11. SSG Amplitude Adjustment (SSGAAH)**

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AA7 | AA6 | AA5 | AA4 | AA3 | AA2 | AA1 | AA0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-12. SSG Amplitude Adjustment (SSGAAL)**

Read: Anytime

Write: Anytime

**Table 19-9. SSGAA Field Descriptions**

| Field | Description |
|---|---|
| 10–0<br>AA[10:0] | **SSG Amplitude Adjustment Register Bits —**<br>The bits define the amplitude adjustment value in each linear attack/decay operation. The register is only used when linear attack/decay is selected. |

## 19.3.2.7 SSG Amplitude Threshold Register (SSGAT)

SSGAT contains the amplitude threshold.

Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | AT10 | AT9 | AT8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-13. SSG Amplitude Threshold (SSGATH)**

Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AT7 | AT6 | AT5 | AT4 | AT3 | AT2 | AT1 | AT0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-14. SSG Amplitude Threshold (SSGATL)**

Read: Anytime

Write: Anytime

**Table 19-10. SSGAT Field Descriptions**

| Field | Description |
|---|---|
| 10–0<br>AT[10:0] | **SSG Amplitude Threshold Register Bits —**<br>When the amplitude >= SSGAT (attack) or amplitude <=SSGAT (decay), an interrupt will be triggered. |

### 19.3.2.8 SSG Tone Duration register (SSGDUR)

The SSGDUR register defines the number of tone cycle in one tone duration. The tone cycle number in one duration is SSGDUR + 1.

Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DUR7 | DUR6 | DUR5 | DUR4 | DUR3 | DUR2 | DUR1 | DUR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-15. SSG Tone Duration Register (SSGDUR)**

Read: Anytime

Write: Anytime

**Table 19-11. SSGDUR Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>DUR[7:0] | **SSG Tone Duration Register bits —**<br>These bits defines the number of tone cycle. The tone number in one duration is SSGDUR + 1.<br>In attack/decay mode, when tone duration counter reach the SSGDUR + 1, amplitude will increase/decrease by SSGAA.<br>In non-attack/decay mode, when tone duration counter reach the SSGDUR + 1, config registers will reload if RDR of SSGCR is 1, or SSG will stop if RDR of SSGCR is 0. |

### 19.3.2.9 SSG Interrupt Enable (SSGIE)

The SSGIE controls the enable of the interrupt.

Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RNDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-16. SSG Interrupt Enable Register (SSGIE)**

**Table 19-12. SSGIE Field Descriptions**

| Field | Description |
|---|---|
| 0<br>RNDIE | **SSG Ready for Next Sound Data Interrupt Enable —**<br>Make the interrupt flag bit RNDI can be set when next sound data is ready to config registers.<br>0 The interrupt signal will not be set when SSGIF is set.<br>1 The interrupt signal will be set when SSGIF is set. |

## 19.3.2.10 SSG Interrupt Flag Registers (SSGIF)

The SSGIF is the Interrupt flag.

Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RNDI |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-17. SSG Interrupt Flag Register (SSGIF)**

**Table 19-13. SSGIF Field Descriptions**

| Field | Description |
|---|---|
| 0<br>RNDI | **SSG Ready for Next Sound Data Interrupt —**<br>The interrupt flag will be set in two cases: one is when the current sound configuration data is loaded into internal buffer registers, after the flag is set, user can configure the registers for the next sound data. RNDI will get set as well when the reload time is reached and RDR is cleared. Writing 1 to this bit will clear the flag bit, writing 0 has no effect on this bit. The details of internal registers loading refer to Section 19.4.5, “Register Reload.<br>0 The SSG is not ready to config the next sound data.<br>1 The SSG is ready to config the next sound data. |

## 19.3.2.11 Buffer Register of SSGAMP (SSGAMPB)

The SSGAMPB is the buffer register of SSGAMP. It determines amplitude directly.

Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | AMPB10 | AMPB9 | AMPB8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-18. Buffer Register of SSGAMP (SSGAMPBH)**

Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AMPB7 | AMPB6 | AMPB5 | AMPB4 | AMPB3 | AMPB2 | AMPB1 | AMPB0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-19. Buffer Register of SSGAMP (SSGAMPBL)**

Read: Anytime

Write: Anytime

**Table 19-14. SSGAMPB Field Descriptions**

| Field | Description |
|---|---|
| 10–0<br>AMPB[10:0] | **SSGAMPB Bits —**<br>The register is read only. In reload time the SSGAMP will be copied to the SSGAMPB. If attack/decay function is enabled, the SSGAMPB will do decay or attack operation automatically leaving the SSGAMP unchanged. The SSGAMPB determines the amplitude directly and finally. |

### 19.3.2.12 SSG Tone Duration Counter (SSGDCNT)

The SSGDCNT is tone duration counter which contains tone cycle number. It is triggered by the negedge of tone signal.

Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DCNT7 | DCNT6 | DCNT5 | DCNT4 | DCNT3 | DCNT2 | DCNT1 | DCNT0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 19-20. SSG Tone Duration Counter (SSGDCNT)**

Read: Anytime

Write: Anytime

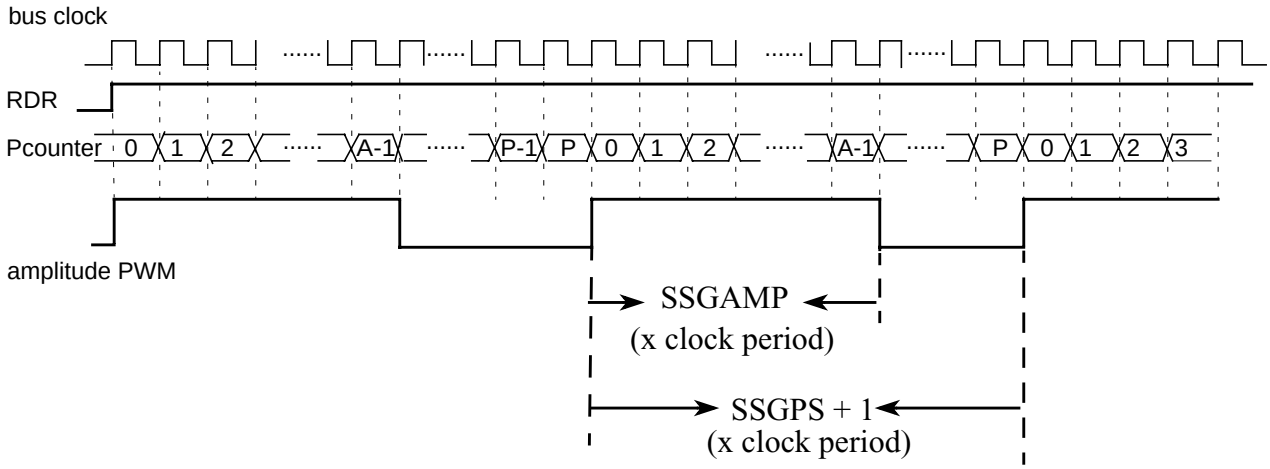
**Table 19-15. SSGDCNT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>DCNT[7:0] | **SSG Tone Duration Counter bits —**<br>The counter register is read only, It contains the tone cycle number. The counter will be reset to 0 when it reaches the SSGDUR value. In any SSG stop case, the counter will also be reset to 0. |

## 19.4 Functional Description

### 19.4.1 SSG Amplitude Generation

The SSG sound amplitude is presented as the duty cycle of PWM signal. The prescaler signal period is the period of the PWM and the amplitude register SSGAMP contains the duty, the amplitude duty cycle is SSGAMP/(SSGPS+1), when SSGAMP>=SSGPS+1, it will be 100%. SSGAMP has a buffer register, so changing the SSGAMP will not disturb the amplitude PWM waveform before the synchronous reload takes place.




where : RDR is the RDR bit of SSGCR

Pcounter is the prescaler counter

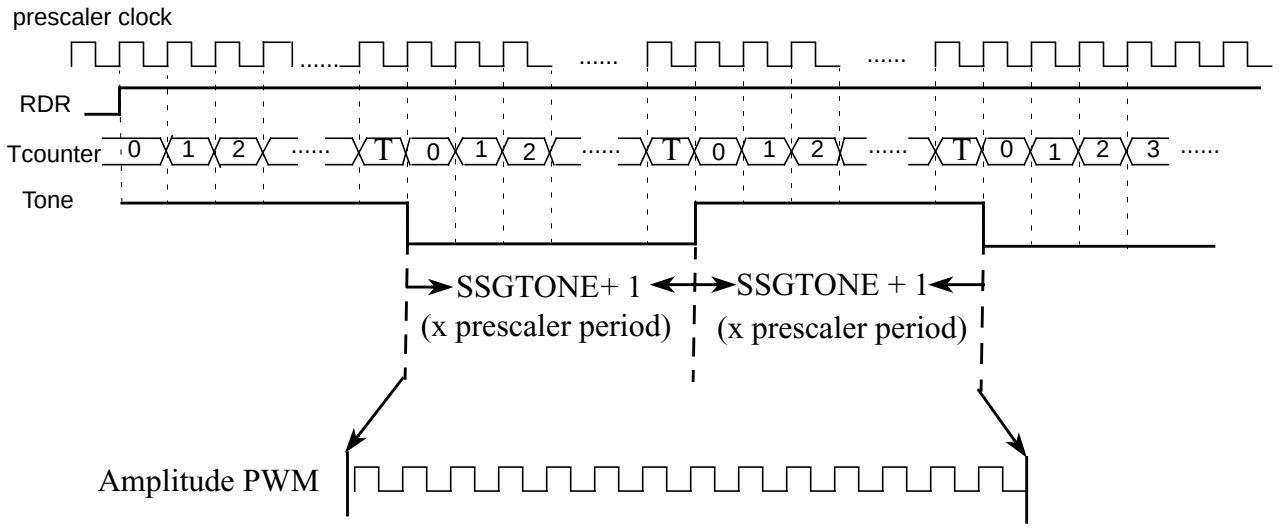
A is value of SSGAMP buffer

P is the value of SSGPS buffer

**Figure 19-21. SSG Amplitude Generation**

## 19.4.2 SSG Tone Generation

The tone is a low frequency square waveform. Tone generation logic consists of a tone counter triggered by prescaler signal and the tone period register SSGTONE with its buffer register. The register SSGTONE contains the coefficient value of half tone period, one tone cycle period = 2 x (SSGTONE + 1) x prescaler cycle. The tone signal is also used as one of the synchronous reload signal.




where : RDR is the RDR bit of SSGCR

Tcounter is the tone counter

T is value of SSGTONE buffer

**Figure 19-22. SSG Tone Generation**

## 19.4.3 SSG Attack and Decay function

In SSG the value hold in the amplitude buffer register can be automatically increased or decreased. The amplitude increase implements sound volume attack and the amplitude decrease implements sound volume decay. If ADS is cleared while the ADE is set, attack function will be enabled, if the ADS is set while ADE is set, decay function will be enabled. If ADE is cleared, attack/decay function will be disabled. The attack/decay function implements sound volume automatic increase/decrease with a small number of CPU interventions.

Attack/decay function includes linear operation, gong operation and exponential operation.

- In linear attack operation SSGAMPB (SSGAMP's buffer) will be increased by SSGAA every SSGSSGDUR + 1 tone cycle. In linear decay operation, SSGAMPB will be decreased by SSGAA every SSGDUR + 1 tone cycle. See below linear attack/decay formula.
- In gong attack operation SSGAMPB will be increased by SSGAMPB/32 every SSGDUR + 1 tone cycle. In gong decay operation SSGAMPB will be decreased by SSGAMPB/32 every SSGDUR + 1 tone cycle. See below gong attack/decay formula.
- In exponential attack operation SSGAMPB will multiply with 2 then add 1 every SSGDUR + 1 tone cycle. In exponential decay operation, SSGAMPB will be divided by 2 every SSGDUR + 1 tone cycle. See below exponential attack/decay formula.

Linear attack operation:

```
SSGAMPB = SSGAMP;
do{
    SSGAMPB = SSGAMPB + SSGAA_buf;
} While (SSGAMPB < AT_buf)
```

Where : AT_buf is the internal buffer of amplitude threshold register SSGAT.
SSGAA_buf is the internal buffer of SSGAA.

Linear decay operation:

```
SSGAMPB = SSGAMP;
do{
    SSGAMPB = SSGAMPB - SSGAA_buf;
} While (SSGAMPB > AT_buf)
```

Where : AT_buf is the internal buffer of amplitude threshold register SSGAT.
SSGAA_buf is the internal buffer of SSGAA.

Gong attack operation:

```
SSGAMPB = SSGAMP;
AMP_int = {SSGAMPB, 5'b00000};

do{
    AMP_int = AMP_int + (AMP_int >> 5);
    SSGAMPB = AMP_int >> 5;
  } While (SSGAMPB < AT_buf)
```

where : AMP_int is a 16 bit internal register.

AT_buf is the internal buffer of amplitude threshold register SSGAT.

Gong decay operation:

```
SSGAMPB = SSGAMP;
AMP_int = {SSGAMPB, 5'b11111};

do{
    AMP_int = AMP_int - (AMP_int >> 5);
    SSGAMPB = AMP_int >> 5;
  } While (SSGAMPB > AT_buf)
```

where : AMP_int is a 16 bit internal register.

AT_buf is the internal buffer of amplitude threshold register SSGAT.

Exponential attack operation:

```
SSGAMPB = SSGAMP;
do {
    SSGAMPB = SSGAMPB << 1 + 1;
} while (SSGAMPB < AT_buf)
```

Where : AT_buf is the internal buffer of amplitude threshold register SSGAT.

Exponential decay operation:

```
SSGAMPB = SSGAMP;
do {
    SSGAMPB = SSGAMPB >> 1;
} while (SSGAMPB > AT_buf)
```

Where : AT_buf is the internal buffer of amplitude threshold register SSGAT.

### 19.4.4 SSG Start and Stop

To start SSG, user should set SSGE of SSGCR and config all registers, then set the RDR of SSGCR, the SSG will start to work.

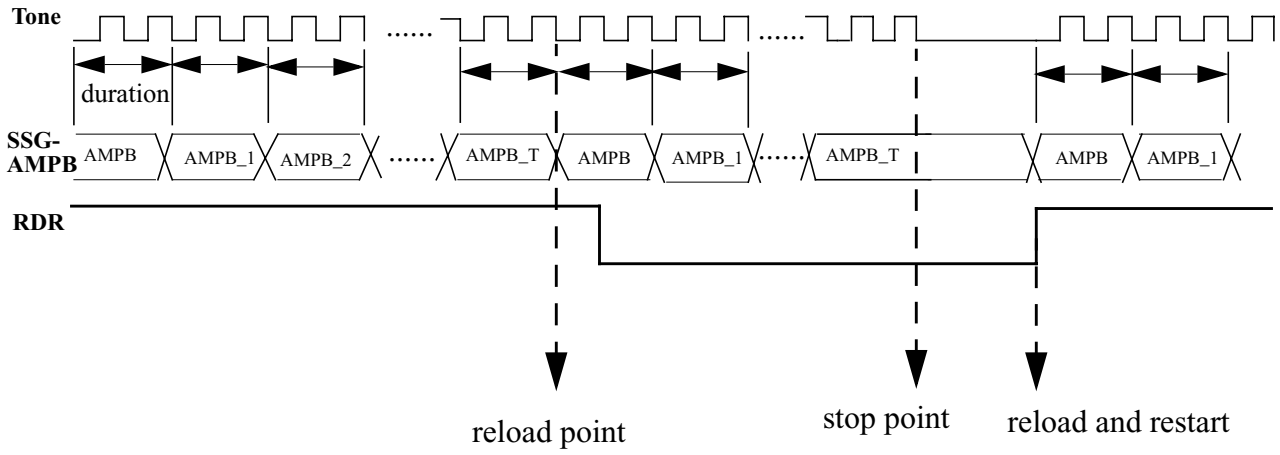
In run time, when the STP of SSGCR is set, the SSG will stop immediately, all counters will be cleared (reset to 0). When STP of SSGCR is cleared, SSG will restart if RDR is set. Same for the SSGE of SSGCR, when clear the SSGE, the SSG will stop immediately, all counters will be cleared (reset to 0). when the SSGE is set, SSG will restart if RDR is set. Clear the SSGE will disable the SGA and SGT output, so it is not recommend to use SSGE to stop the SSG.

The RDR of SSGCR can also stop the SSG, when reload occurs while the RDR is cleared, SSG will stop with all counters cleared, and the interrupt flag (RNDI of SSGIF) will be set. When the RDR is set again, the SSG will restart. Refer to Figure 19-23 and Figure 19-24.

### 19.4.5 Register Reload

In SSG all registers except SSGCR, SSGIE and SSGIF are double buffered. the value will be reloaded from config registers to their buffers in three cases: one is when SSG startup (or restart), the other two are attack/decay mode register reload and non-attack/decay mode register reload. Details as below:

- When at SSG start/restart time, if the SSGCR's RDR is set, data in config registers will be loaded into the internal registers buffers, and the interrupt flag (RNDI of SSGIF) will be set.
- In attack/decay mode, when amplitude (in SSGAMP) reaches the SSGAT value and the SSGCR's RDR is set, all register will reload to their buffer registers, and the interrupt flag (RNDI of SSGIF) will be set. Refer to Figure 19-23.
- In non-attack/decay mode, when one tone duration completes and the SSGCR's RDR is set, all register will reload to their buffer registers, and the interrupt flag (RNDI of SSGIF) will be set. Refer to Figure 19-24.




where : Assume one tone duration contains 2 tone cycle

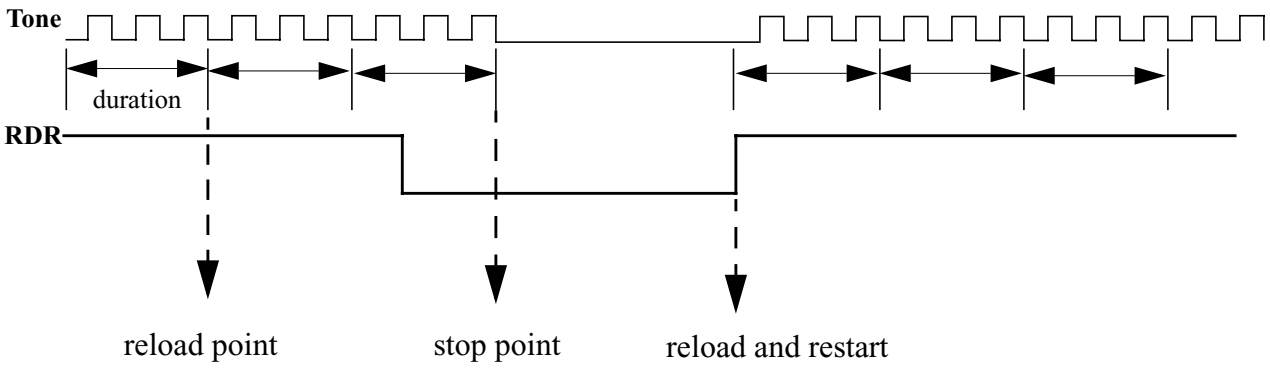
AMPB is the initial value reload from SSGAMP register

AMPB_x is the interim value of SSGAMPB in attack/decay operation

AMPB_T is the value of SSGAT buffer

RDR is the RDR bit of SSGCR register

**Figure 19-23. Attack/decay Mode Register Reload**




where : Assume one tone duration contains 3 tone cycle
RDR is the RDR bit of SSGCR register

**Figure 19-24. Non-attack/decay mode Register Reload**

### 19.4.6 SSG Output Control

There are two output signals, SGA and SGT. SGA outputs the sound amplitude while SGT can output tone or tone mixed with amplitude. The SGT output selection is determined by OMS of SSGCR. When OMS is cleared, the SGA output is disable, the SGA shared pins can be used as other function.

## 19.5 Interrupt

When the RNDI of SSGIF and RNDIE of SSGIE are all set, interrupt will be generated. Besides normal reload interrupt, RNDI will be set when SSG start with RDR set, so the second configuration data can be set base on reload interrupt.

# Chapter 20
# ECC Generation module (SRAM_ECCV1)

## 20.1 Introduction

The purpose of ECC logic is to detect and correct as much as possible memory data bit errors. These soft errors can occur randomly during operation, mainly generated by alpha radiation. Soft Error means, that only the information inside the memory cell is corrupt, the memory cell itself is not damaged. A write access with correct data solves the issue. If the ECC algorithm is able to correct the data, then the system can use this corrected data without any issues. If the ECC algorithm is able to detect, but not correct the error, then the system is able to ignore the memory read data to avoid system malfunction.

The ECC value is calculated based on an aligned 2 byte memory data word. The ECC algorithm is able to detect and correct single bit ECC errors. Double bit ECC errors will be detected but the system is not able to correct these errors. This kind of ECC code is called SECDED code. This ECC code requires 6 additional parity bits for each 2 byte data word.

### 20.1.1 Features

The SRAM_ECC module provides the ECC logic for the system memory based on a SECDED algorithm. Main features of the SRAM_ECC module:

- SECDED ECC code
  - single bit error detection and correction per 2 byte data word
  - double bit error detection per 2 byte data word
- memory initialization function
- byte wide system memory write access
- automatic single bit ECC error correction for read and write accesses
- debug logic to read and write raw use data and ECC values

## 20.2 Memory Map and Register Definition

This section provides a detailed description of all memory and registers for the SRAM_ECC module.

### 20.2.1 Register Summary

Figure 20-1 shows the summary of all implemented registers inside the SRAM_ECC module.

**NOTE**

Register Address = Module Base Address + Address Offset, where the Module Base Address is defined at the MCU level and the Address Offset is defined at the module level.

| Address Offset / Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 ECCSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RDY |
| | W | | | | | | | | |
| 0x0001 ECCIE | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIE |
| | W | | | | | | | | |
| 0x0002 ECCIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIF |
| | W | | | | | | | | |
| 0x0003 - 0x0006 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 ECCDPTRH | R | DPTR[23:16] | | | | | | | |
| | W | | | | | | | | |
| 0x0008 ECCDPTRM | R | DPTR[15:8] | | | | | | | |
| | W | | | | | | | | |
| 0x0009 ECCDPTRL | R | DPTR[7:1] | | | | | | | 0 |
| | W | | | | | | | | |
| 0x000A - 0x000B Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000C ECCDDH | R | DDATA[15:8] | | | | | | | |
| | W | | | | | | | | |
| 0x000D ECCDDL | R | DDATA[7:0] | | | | | | | |
| | W | | | | | | | | |
| 0x000E ECCDE | R | 0 | 0 | DECC[5:0] | | | | | |
| | W | | | | | | | | |
| 0x000F ECCDCMD | R | ECCDRR | 0 | 0 | 0 | 0 | 0 | ECCDW | ECCDR |
| | W | | | | | | | | |

= Unimplemented, Reserved, Read as zero

**Figure 20-1. SRAM_ECC Register Summary**

## 20.2.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field functions follow the register diagrams, in bit order.

### 20.2.2.1 ECC Status Register (ECCSTAT)

Module Base + 0x00000 — Access: User read only[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RDY |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 Read: Anytime
Write: Never

**Figure 20-2. ECC Status Register (ECCSTAT)**

**Table 20-2. ECCSTAT Field Description**

| Field | Description |
|---|---|
| 0<br>RDY | **ECC Ready**— Shows the status of the ECC module.<br>0 Internal SRAM initialization is ongoing, access to the SRAM is disabled<br>1 Internal SRAM initialization is done, access to the SRAM is enabled |

### 20.2.2.2 ECC Interrupt Enable Register (ECCIE)

Module Base + 0x00001 — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 Read: Anytime
Write: Anytime

**Figure 20-3. ECC Interrupt Enable Register (ECCIE)**

**Table 20-3. ECCIE Field Description**

| Field | Description |
|---|---|
| 0<br>SBEEIE | **Single bit ECC Error Interrupt Enable** — Enables Single ECC Error interrupt.<br>0 Interrupt request is disabled<br>1 Interrupt will be requested whenever SBEEIF is set |

### 20.2.2.3 ECC Interrupt Flag Register (ECCIF)

Module Base + 0x0002 — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 Read: Anytime
Write: Anytime, write 1 to clear

**Figure 20-4. ECC Interrupt Flag Register (ECCIF)**

**Table 20-4. ECCIF Field Description**

| Field | Description |
|---|---|
| 0<br>SBEEIF | **Single bit ECC Error Interrupt Flag** — The flag is set to 1 when a single bit ECC error occurs.<br>0 No occurrences of single bit ECC error since the last clearing of the flag<br>1 Single bit ECC error occurs since the last clearing of the flag |

### 20.2.2.4 ECC Debug Pointer Register (ECCDPTRH, ECCDPTRM, ECCDPTRL)

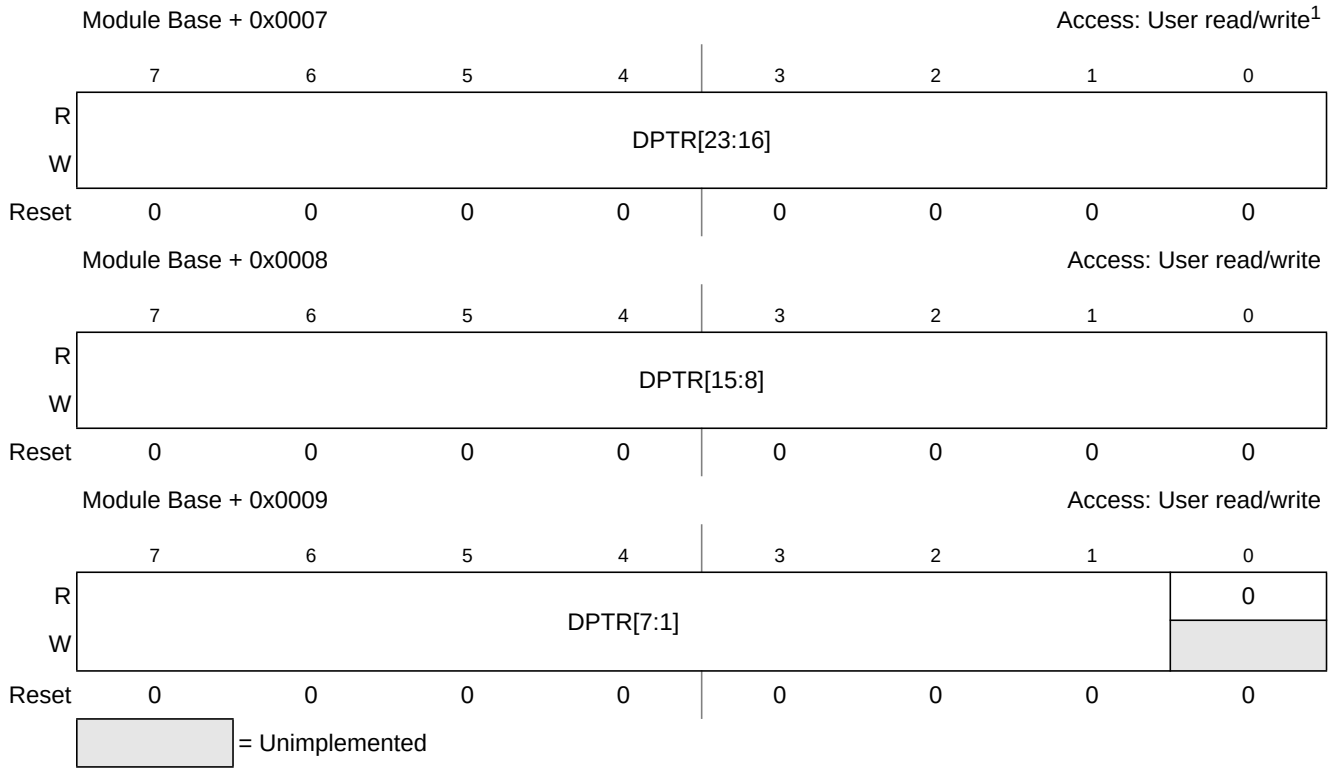



**Figure 20-5. ECC Debug Pointer Register (ECCDPTRH, ECCDPTRM, ECCDPTRL)**

1 Read: Anytime
Write: Anytime

**Table 20-5. ECCDPTR Register Field Descriptions**

| Field | Description |
|---|---|
| DPTR [23:0] | **ECC Debug Pointer** — This register contains the system memory address which will be used for a debug access. Address bits not relevant for SRAM address space are not writeable, so the SW should read back the pointer value to make sure the register contains the intended memory address. Anyway it is possible to write an address value to this register which points outside the system memory. There is no additional monitoring of the register content, therefore the SW must make sure that the address value points to the system memory space. |

### 20.2.2.5 ECC Debug Data (ECCDDH, ECCDDL)

Module Base + 0x000C — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDATA[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Module Base + 0x000D — Access: User read/write

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DDATA[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 20-6. ECC Debug Data (ECCDDH, ECCDDL)**

1 Read: Anytime
Write: Anytime

**Table 20-6. ECCDD Register Field Descriptions**

| Field | Description |
|---|---|
| DDATA [23:0] | **ECC Debug Raw Data** — This register contains the raw data which will be written into the system memory during a debug write command or the read data from the debug read command. |

### 20.2.2.6 ECC Debug ECC (ECCDE)

Module Base + 0x000E — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | DECC[5:0] | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 Read: Anytime
Write: Anytime

**Figure 20-7. ECC Debug ECC (ECCDE)**

**Table 20-7. ECCDE Field Description**

| Field | Description |
|---|---|
| 5:0 DECC[5:0] | **ECC Debug ECC** — This register contains the raw ECC value which will be written into the system memory during a debug write command or the ECC read value from the debug read command. |

### 20.2.2.7 ECC Debug Command (ECCDCMD)

Module Base + 0x000F — Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ECCDRR | 0 | 0 | 0 | 0 | 0 | ECCDW | ECCDR |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[1] Read: Anytime
Write: Anytime, in special mode only

**Figure 20-8. ECC Debug Command (ECCDCMD)**

**Table 20-8. ECCDCMD Field Description**

| Field | Description |
|---|---|
| 7<br>ECCDRR | **ECC Disable Read Repair Function**— Write one to this register bit will disable the automatic single bit ECC error repair function during read access, see also chapter 20.3.7, “ECC Debug Behavior”.<br>0 Automatic single ECC error repair function is enabled<br>1 Automatic single ECC error repair function is disabled |
| 1<br>ECCDW | **ECC Debug Write Command** — Write one to this register bit will perform a debug write access to the system memory. During this access the debug data word (DDATA) and the debug ECC value (DECC) will be written to the system memory address defined by DPTR. If the debug write access is done this bit is cleared. Writing 0 has no effect**.** It is not possible to set this bit if the previous debug access is ongoing. (ECCDW or ECCDR bit set) |
| 0<br>ECCDR | **ECC Debug Read Command** — Write one to this register bit will perform a debug read access from the system memory address defined by DPTR. If the debug read access is done this bit is cleared and the raw memory read data are available in register DDATA and the raw ECC value is available in register DECC. Writing 0 has no effect. If the ECCDW and ECCDR bit are set at the same time, then only the ECCDW bit is set and the Debug Write Command is performed. It is not possible to set this bit if the previous debug access is ongoing. (ECCDW or ECCDR bit set) |

## 20.3 Functional Description

The bus system allows 1, 2, 3 and 4 byte write access to a 4 byte aligned memory address, but the ECC value is generated based on an aligned 2 byte data word. Depending on the access type, the access is separated into different access cycles. Table 20-9 shows the different access types with the expected number of access cycles and the performed internal operations.

**Table 20-9. Memory access cycles**

| Access type | ECC error | access cycle | Internal operation | Memory content | Error indication |
|---|---|---|---|---|---|
| 2 and 4 byte aligned write access | - | 1 | write to memory | new data | - |

**Table 20-9. Memory access cycles**

| Access type | ECC error | access cycle | Internal operation | Memory content | Error indication |
|---|---|---|---|---|---|
| 1 or 3 byte write, non-aligned 2 byte write | no | 2 | read data from the memory<br>write old + new data to the memory | old + new data | - |
| | single bit | 2 | read data from the memory<br>write corrected + new data to the memory | corrected + new data | SBEEIF |
| | double bit | 2 | read data from the memory<br>ignore write data | unchanged | initiator module is informed |
| read access | no | 1 | read from memory | unchanged | - |
| | single bit | 1[1] | read data from the memory<br>write corrected data back to memory | corrected data | SBEEIF |
| | double bit | 1 | read from memory | unchanged | data mark as invalid |

1 the next back to back read access to the memory will be delayed by one clock cycle

The single bit ECC errors generates an interrupt when enabled. The double bit ECC errors are reported by the SRAM_ECC module, but are handled outside by MCU level. For more information see MMC description.

### 20.3.1 Aligned 2 and 4 Byte Memory write access

During an aligned 2 or 4 byte memory write access no ECC check is performed. The internal ECC logic generates the new ECC value based on the write data and writes the data words together with the generated ECC values into the memory.

### 20.3.2 Other Memory write access

Other types of write accesses are separated into read-modify-write operation. During the first cycle, the logic reads the data from the memory and performs an ECC check. If no ECC errors was detected then the logic generates the new ECC value based on the read and write data and writes the new data word together with the new ECC value into the memory. If required both 2 byte data words are updated.

If the module detects a single bit ECC error during the read cycle, then the logic generates the new ECC value based on the corrected read and new write read. In the next cycle new data word and the new ECC value are written into the memory. If required both 2 byte data words are updated. The SBEEIF bit is set. Hence the single bit ECC error was corrected by the write access. Figure 20-9 shows an example of a 2 byte non-aligned memory write access.

If the module detects a double bit ECC error during the read cycle, then the write access to the memory is blocked and the initiator module is informed about the error.

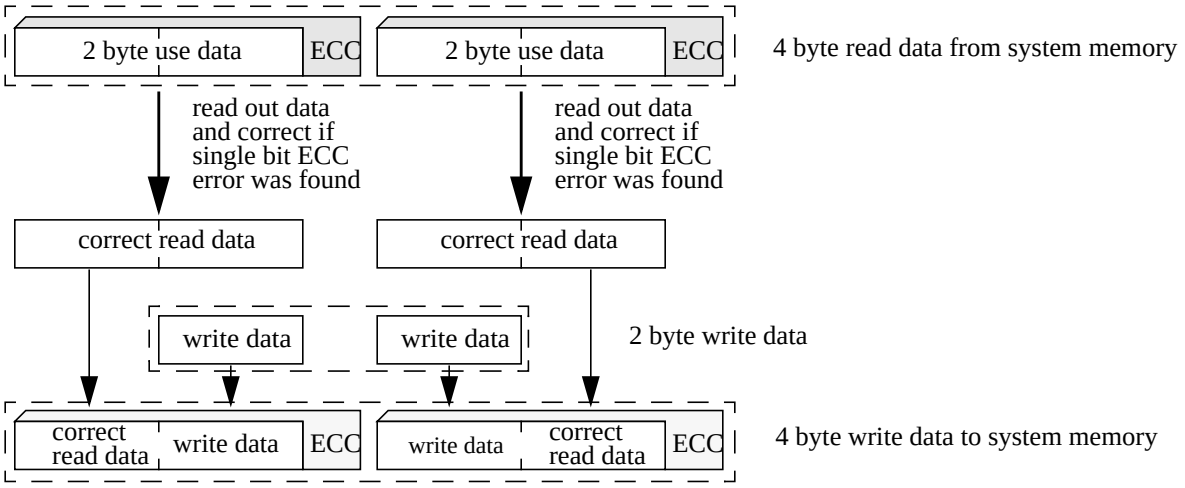



**Figure 20-9. 2 byte non-aligned write access**

## 20.3.3 Memory read access

During each memory read access an ECC check is performed. If the logic detects a single bit ECC error then the module corrects the data, so that the access initiator module receives correct data. In parallel, the logic writes the corrected data back to the memory, so that this read access repairs the single bit ECC error. This automatic ECC read repair function is disabled by setting the ECCDRR bit.

If a single bit ECC error was detected then the SBEEIF flag is set.

If the logic detects a double bit ECC error, then the data word is flagged as invalid, so that the access initiator module can ignore the data.

## 20.3.4 Memory initialization

Memory operation which allows a read before a first write, like the read-modify-write operation of the un-aligned access, requires that the memory contains valid ECC values before the first read-modify-write access is performed to avoid spurious ECC error reporting. The ECC module provides logic to initialize the complete memory content with zero during the power up phase. During the initialization process the access to the SRAM is disabled and the RDY status bit is cleared. If the initialization process is done, the SRAM access is possible and the RDY status bit is set.

## 20.3.5 Interrupt handling

This sections describes the interrupts generated by the SRAM_ECC module and their individual sources, Vector addresses and interrupt priority are defined by MCU level.

**Table 20-10. SRAM_ECC Interrupt Sources**

| Module Interrupt Sources | Local Enable |
|---|---|
| Single bit ECC error | ECCIE[SBEEIE] |

## 20.3.6 ECC Algorithm

The table below shows the equation for each ECC bit based on the 16 bit data word.

**Table 20-11. ECC Calculation**

| ECC bit | Use data |
|---|---|
| ECC[0] | ~ ( ^ ( data[15:0] & 0x443F ) ) |
| ECC[1] | ~ ( ^ ( data[15:0] & 0x13C7 ) ) |
| ECC[2] | ~ ( ^ ( data[15:0] & 0xE1D1 ) ) |
| ECC[3] | ~ ( ^ ( data[15:0] & 0xEE60 ) ) |
| ECC[4] | ~ ( ^ ( data[15:0] & 0x3E8A ) ) |
| ECC[5] | ~ ( ^ ( data[15:0] & 0x993C ) ) |

## 20.3.7 ECC Debug Behavior

For debug purpose it is possible to read and write the uncorrected use data and the raw ECC value direct from the memory. For these debug accesses a register interface is available. The debug access is performed with the lowest priority, other memory accesses must be done before the debug access starts. If a debug access is requested during a ongoing memory initialization process then the debug access is performed if the memory initialization process is done.

If the ECCDRR bit is set then the automatic single bit ECC error repair function for all read accesses is disabled. In this case a read access from a system memory location with single bit ECC error will produce correct data and the single bit ECC error is flagged by the SBEEIF, but the data inside the system memory are unchanged.

By writing wrong ECC values into the system memory the debug access can be used to force single and double bit ECC errors to check the SW error handling.

It is not possible to set the ECCDW or ECCDR bit if the previous debug access is ongoing. (ECCDW or ECCDR bit active) This makes sure that the ECCDD and ECCDE registers contains consistent data. The SW should read out the status of the ECCDW and ECCDR register bit before a new debug access is requested.

### 20.3.7.1 ECC Debug Memory Write Access

Writing one to the ECCDW bit performs a debug write access to the memory address defined by register DPTR. During this access, the raw data DDATA and the ECC value DECC are written direct into the system memory. If the debug write access is done the ECCDW register bit is cleared. The debug write

access is always a 2 byte aligned memory access, so that no ECC check is performed and no single or double bit ECC error indication are activated.

#### 20.3.7.2 ECC Debug Memory Read Access

Writing one to the ECCDR bit performs a debug read access from the memory address defined by register DPTR. If the ECCDR bit is cleared then the register DDATA contains the uncorrected read data from the memory. The register DECC contains the ECC value read from the memory. Independent of the ECCDRR register bit setting, the debug read access will not perform an automatic ECC repair during read access. During the debug read access no ECC check is performed, so that no single or double bit ECC error indication are activated.

If the ECCDW and the ECCDR bits are set at the same time, then only the debug write access is performed.

# Chapter 21
# 64 KB Flash Module (S12ZFTMRZ64K2KV2)

**Table 21-1. Revision History**

| Revision Number | Revision Date | Sections Affected | Description of Changes |
|---|---|---|---|
| V01.00 | 06 Mar 2012 | | - Initial version |
| V02.02 | 20 Mar 2012 | | - Updated feature description, Section 21.4.6 |
| V02.03 | 12 Apr 2012 | 21.3 | Corrected many typo.<br>Changed caution note |
| V02.04 | 17 May 2012 | 21.3.2.6 | - Removed flag DFDIE |
| V02.05 | 11 Jul 2012 | | - Added explanation about when MGSTAT[1:0] bits are cleared, Section 21.3.2.7<br>- Added note about possibility of reading P-Flash and EEPROM simultaneously, Section 21.4.6 |
| V02.06 | 18 Mar 2013 | | - Standardized nomenclature in references to memory sizes |
| V02.07 | 24 May 2013 | | - Revised references to NVM Resource Area to improve readability |
| V02.08 | 12 Jun 2013 | | - Changed MLOADU Section 21.4.7.12 and MLOADF Section 21.4.7.13 **FCCOB1 to FCCOB2** |

## 21.1 Introduction

The FTMRZ64K2K module implements the following:

- 64 KB of P-Flash (Program Flash) memory
- 2 KB of EEPROM memory

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

The Flash memory may be read as bytes and aligned words. Read access time is one bus cycle for bytes and aligned words. For misaligned words access, the CPU has to perform twice the byte read access command. For Flash memory, an erased bit reads 1 and a programmed bit reads 0.

It is possible to read from P-Flash memory while some commands are executing on EEPROM memory. It is not possible to read from EEPROM memory while a command is executing on P-Flash memory. Simultaneous P-Flash and EEPROM operations are discussed in Section 21.4.6.

Both P-Flash and EEPROM memories are implemented with Error Correction Codes (ECC) that can resolve single bit faults and detect double bit faults. For P-Flash memory, the ECC implementation requires that programming be done on an aligned 8 byte basis (a Flash phrase). Since P-Flash memory is always read by half-phrase, only one single bit fault in an aligned 4 byte half-phrase containing the byte or word accessed will be corrected.

### 21.1.1 Glossary

**Command Write Sequence** — An MCU instruction sequence to execute built-in algorithms (including program and erase) on the Flash memory.

**EEPROM Memory** — The EEPROM memory constitutes the nonvolatile memory store for data.

**EEPROM Sector** — The EEPROM sector is the smallest portion of the EEPROM memory that can be erased. The EEPROM sector consists of 4 bytes.

**NVM Command Mode** — An NVM mode using the CPU to setup the FCCOB register to pass parameters required for Flash command execution.

**Phrase** — An aligned group of four 16-bit words within the P-Flash memory. Each phrase includes two sets of aligned double words with each set including 7 ECC bits for single bit fault correction and double bit fault detection within each double word.

**P-Flash Memory** — The P-Flash memory constitutes the main nonvolatile memory store for applications.

**P-Flash Sector** — The P-Flash sector is the smallest portion of the P-Flash memory that can be erased. Each P-Flash sector contains 512 bytes.

**Program IFR** — Nonvolatile information register located in the P-Flash block that contains the Version ID, and the Program Once field.

## 21.1.2 Features

### 21.1.2.1 P-Flash Features

- 64 KB of P-Flash composed of one 64 KB Flash block divided into 128 sectors of 512 bytes
- Single bit fault correction and double bit fault detection within a 32-bit double word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and phrase program operation
- Ability to read the P-Flash memory while programming a word in the EEPROM memory
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

### 21.1.2.2 EEPROM Features

- 2 KB of EEPROM memory composed of one 2 KB Flash block divided into 512 sectors of 4 bytes
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Protection scheme to prevent accidental program or erase of EEPROM memory
- Ability to program up to four words in a burst sequence

### 21.1.2.3 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

## 21.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 21-1.

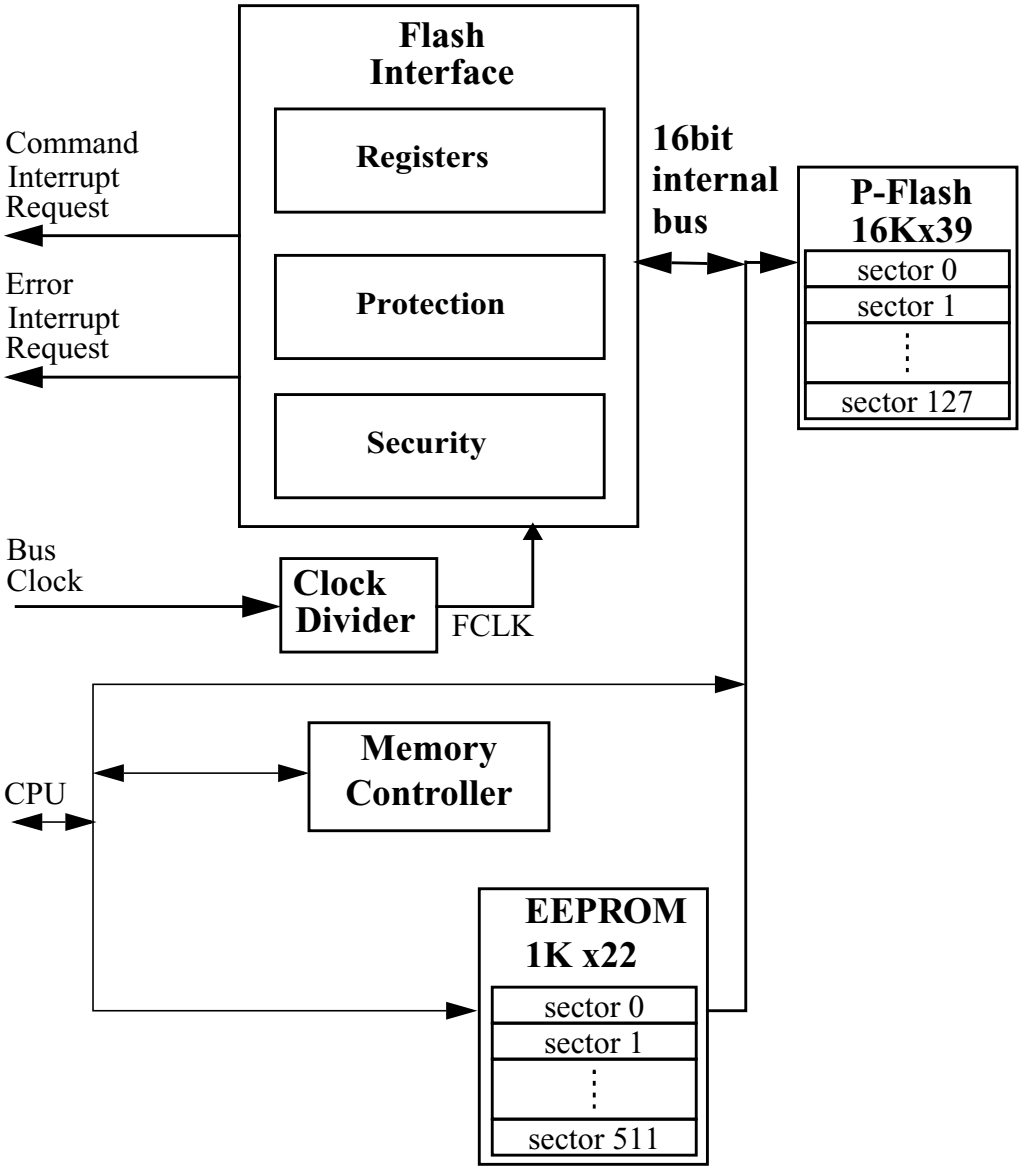



**Figure 21-1. FTMRZ64K2K Block Diagram**

## 21.2 External Signal Description

The Flash module contains no signals that connect off-chip.

## 21.3 Memory Map and Registers

This section describes the memory map and registers for the Flash module. Read data from unimplemented memory space in the Flash module is undefined. Write access to unimplemented or reserved memory space in the Flash module will be ignored by the Flash module.

**CAUTION**

Writing to the Flash registers while a Flash command is executing (that is indicated when the value of flag CCIF reads as ’0’) is not allowed. If such action is attempted, the result of the write operation will be unpredictable.

Writing to the Flash registers is allowed when the Flash is not busy executing commands (CCIF = 1) and during initialization right after reset, despite the value of flag CCIF in that case (refer to Section 21.6 for a complete description of the reset sequence).

.

**Table 21-2. FTMRZ Memory Map**

| Global Address (in Bytes) | Size (Bytes) | Description |
|---|---|---|
| 0x0_0000 – 0x0_0FFF | 4,096 | Register Space |
| 0x10_0000 – 0x10_07FF | 2,048 | EEPROM memory |
| 0x1F_4000 – 0x1F_FFFF | 49,152 | NVM Resource Area[1] (see Figure 21-3) |
| 0xFF_0000 – 0xFF_FFFF | 65,536 | P-Flash Memory |

[1] See NVM Resource area description in Section 21.4.4

### 21.3.1 Module Memory Map

The S12Z architecture places the P-Flash memory between global addresses 0xFF_0000 and 0xFF_FFFF as shown in Table 21-3

The P-Flash memory map is shown in Figure 21-2.

**Table 21-3. P-Flash Memory Addressing**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0xFF_0000 – 0xFF_FFFF | 64 K | P-Flash Block<br>Contains Flash Configuration Field<br>(see Table 21-4) |

The FPROT register, described in Section 21.3.2.9, can be set to protect regions in the Flash memory from accidental program or erase. Three separate memory regions, one growing upward from global address 0xFF_8000 in the Flash memory (called the lower region), one growing downward from global address 0xFF_FFFF in the Flash memory (called the higher region), and the remaining addresses in the Flash

memory, can be activated for protection. The Flash memory addresses covered by these protectable regions are shown in the P-Flash memory map. The higher address region is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field as described in Table 21-4.

**Table 21-4. Flash Configuration Field**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0xFF_FE00-0xFF_FE07 | 8 | Backdoor Comparison Key<br>Refer to Section 21.4.7.11, “Verify Backdoor Access Key Command,” and Section 21.5.1, “Unsecuring the MCU using Backdoor Key Access” |
| 0xFF_FE08-0xFF_FE09[1] | 2 | Protection Override Comparison Key. Refer to Section 21.4.7.17, “Protection Override Command” |
| 0xFF_FE0A-0xFF_FE0B[1] | 2 | Reserved |
| 0xFF_FE0C[1] | 1 | P-Flash Protection byte.<br>Refer to Section 21.3.2.9, “P-Flash Protection Register (FPROT)” |
| 0xFF_FE0D[1] | 1 | EEPROM Protection byte.<br>Refer to Section 21.3.2.10, “EEPROM Protection Register (DFPROT)” |
| 0xFF_FE0E[1] | 1 | Flash Nonvolatile byte<br>Refer to Section 21.3.2.11, “Flash Option Register (FOPT)” |
| 0xFF_FE0F[1] | 1 | Flash Security byte<br>Refer to Section 21.3.2.2, “Flash Security Register (FSEC)” |

1 0xFF_FE08-0xFF_FE0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0xFF_FE0A - 0xFF_FE0B reserved field should be programmed to 0xFF.

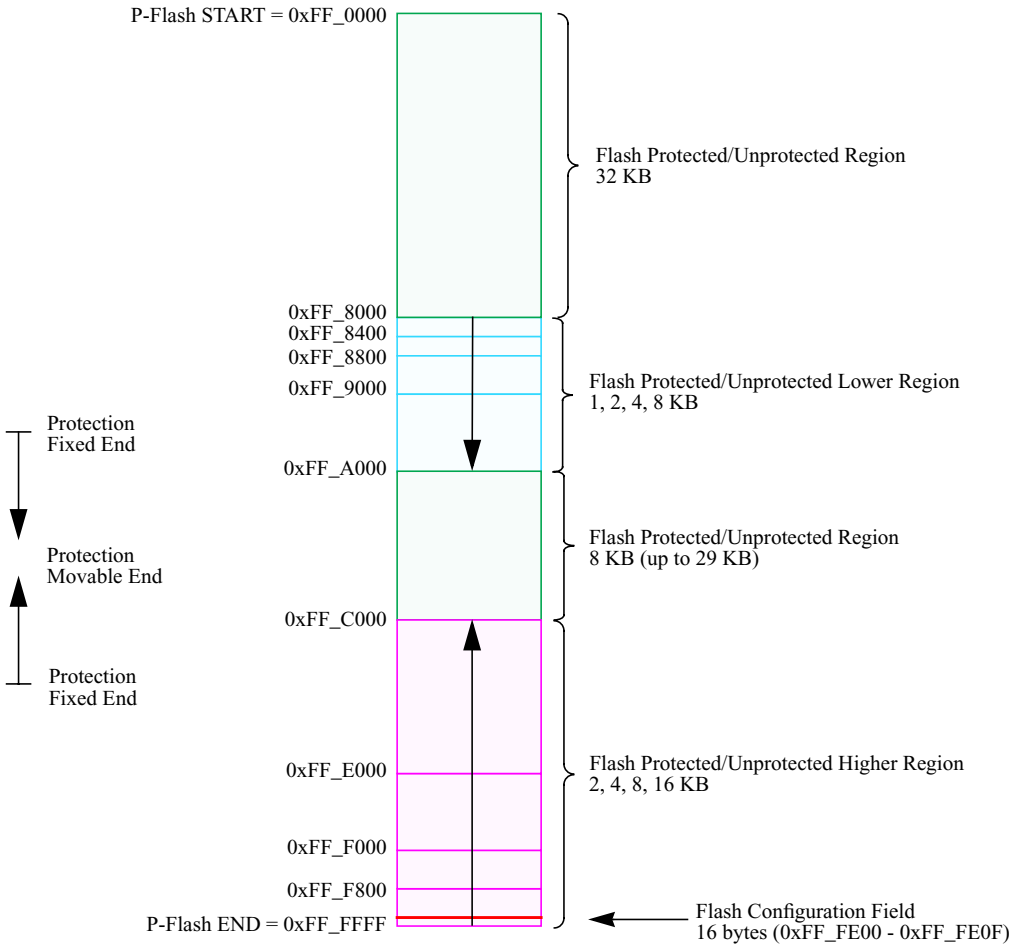



**Figure 21-2. P-Flash Memory Map**

**Table 21-5. Program IFR Fields**

| Global Address | Size (Bytes) | Field Description |
|---|---|---|
| 0x1F_C000 – 0x1F_C007 | 8 | Reserved |
| 0x1F_C008 – 0x1F_C0B5 | 174 | Reserved |
| 0x1F_C0B6 – 0x1F_C0B7 | 2 | Version ID[1] |
| 0x1F_C0B8 – 0x1F_C0BF | 8 | Reserved |
| 0x1F_C0C0 – 0x1F_C0FF | 64 | Program Once Field<br>Refer to Section 21.4.7.6, “Program Once Command” |

1 Used to track firmware patch versions, see Section 21.4.2

**Table 21-6. Memory Controller Resource Fields (NVM Resource Area[1])**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x1F_4000 – 0x1F_41FF | 512 | Reserved |
| 0x1F_4200 – 0x1F_7FFF | 15,872 | Reserved |
| 0x1F_8000 – 0x1F_97FF | 6,144 | Reserved |
| 0x1F_9800 – 0x1F_BFFF | 10,240 | Reserved |
| 0x1F_C000 – 0x1F_C0FF | 256 | P-Flash IFR (see Table 21-5) |
| 0x1F_C100 – 0x1F_C1FF | 256 | Reserved. |
| 0x1F_C200 – 0x1F_FFFF | 15,872 | Reserved. |

1 See Section 21.4.4 for NVM Resources Area description.

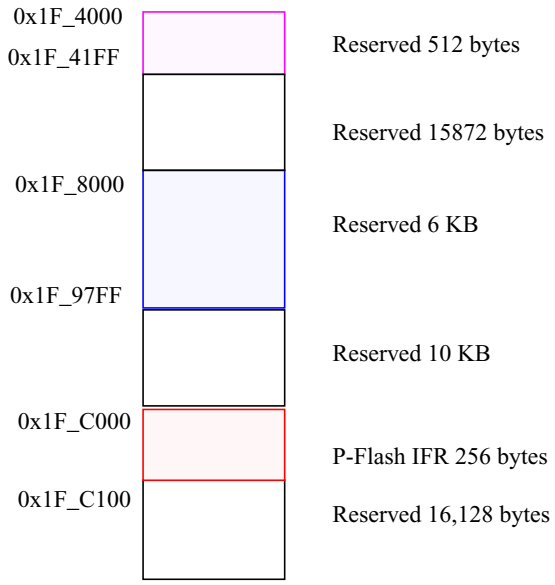



**Figure 21-3. Memory Controller Resource Memory Map (NVM Resources Area)**

## 21.3.2 Register Descriptions

The Flash module contains a set of 24 control and status registers located between Flash module base + 0x0000 and 0x0017.

In the case of the writable registers, the write accesses are forbidden during Flash command execution (for more detail, see Caution note in Section 21.3).

A summary of the Flash module registers is given in Figure 21-4 with detailed descriptions in the following subsections.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 FCLKDIV | R | FDIVLD | FDIVLCK | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | W | | | | | | | | |
| 0x0001 FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | W | | | | | | | | |
| 0x0002 FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | W | | | | | | | | |

**Figure 21-4. FTMRZ64K2K Register Summary**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0003 FPSTAT | R | FPOVRD | 0 | 0 | 0 | 0 | 0 | 0 | WSTATACK |
| | W | | | | | | | | |
| 0x0004 FCNFG | R | CCIE | 0 | ERSAREQ | IGNSF | WSTAT[1:0] | | FDFD | FSFD |
| | W | | | | | | | | |
| 0x0005 FERCNFG | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SFDIE |
| | W | | | | | | | | |
| 0x0006 FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | W | | | | | | | | |
| 0x0007 FERSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | DFDF | SFDIF |
| | W | | | | | | | | |
| 0x0008 FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | W | | | | | | | | |
| 0x0009 DFPROT | R | DPOPEN | 0 | DPS5 | DPS4 | DPS3 | DPS2 | DPS1 | DPS0 |
| | W | | | | | | | | |
| 0x000A FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x000B FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000C FCCOB0HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000D FCCOB0LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000E FCCOB1HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000F FCCOB1LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x0010 FCCOB2HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |

**Figure 21-4. FTMRZ64K2K Register Summary (continued)**

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0011 FCCOB2LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x0012 FCCOB3HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x0013 FCCOB3LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x0014 FCCOB4HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x0015 FCCOB4LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x0016 FCCOB5HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x0017 FCCOB5LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 21-4. FTMRZ64K2K Register Summary (continued)**

### 21.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

Offset Module Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FDIVLD | FDIVLCK | FDIV[5:0] | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-5. Flash Clock Divider Register (FCLKDIV)**

All bits in the FCLKDIV register are readable, bit 7 is not writable, bit 6 is write-once-hi and controls the writability of the FDIV field in normal mode. In special mode, bits 6-0 are writable any number of times but bit 7 remains unwritable.

**CAUTION**

The FCLKDIV register should never be written while a Flash command is executing (CCIF=0).

**Table 21-7. FCLKDIV Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FDIVLD | **Clock Divider Loaded**<br>0 FCLKDIV register has not been written since the last reset<br>1 FCLKDIV register has been written since the last reset |
| 6<br>FDIVLCK | **Clock Divider Locked**<br>0 FDIV field is open for writing<br>1 FDIV value is locked and cannot be changed. Once the lock bit is set high, only reset can clear this bit and restore writability to the FDIV field in normal mode. |
| 5–0<br>FDIV[5:0] | **Clock Divider Bits** — FDIV[5:0] must be set to effectively divide BUSCLK down to 1 MHz to control timed events during Flash program and erase algorithms. Table 21-8 shows recommended values for FDIV[5:0] based on the BUSCLK frequency. Please refer to Section 21.4.5, “Flash Command Operations,” for more information. |

**Table 21-8. FDIV values for various BUSCLK Frequencies**

| BUSCLK Frequency (MHz) | | FDIV[5:0] | BUSCLK Frequency (MHz) | | FDIV[5:0] |
|---|---|---|---|---|---|
| MIN[1] | MAX[2] | | MIN[1] | MAX[2] | |
| 1.0 | 1.6 | 0x00 | 26.6 | 27.6 | 0x1A |
| 1.6 | 2.6 | 0x01 | 27.6 | 28.6 | 0x1B |
| 2.6 | 3.6 | 0x02 | 28.6 | 29.6 | 0x1C |
| 3.6 | 4.6 | 0x03 | 29.6 | 30.6 | 0x1D |
| 4.6 | 5.6 | 0x04 | 30.6 | 31.6 | 0x1E |
| 5.6 | 6.6 | 0x05 | 31.6 | 32.6 | 0x1F |
| 6.6 | 7.6 | 0x06 | 32.6 | 33.6 | 0x20 |
| 7.6 | 8.6 | 0x07 | 33.6 | 34.6 | 0x21 |
| 8.6 | 9.6 | 0x08 | 34.6 | 35.6 | 0x22 |
| 9.6 | 10.6 | 0x09 | 35.6 | 36.6 | 0x23 |
| 10.6 | 11.6 | 0x0A | 36.6 | 37.6 | 0x24 |
| 11.6 | 12.6 | 0x0B | 37.6 | 38.6 | 0x25 |
| 12.6 | 13.6 | 0x0C | 38.6 | 39.6 | 0x26 |
| 13.6 | 14.6 | 0x0D | 39.6 | 40.6 | 0x27 |
| 14.6 | 15.6 | 0x0E | 40.6 | 41.6 | 0x28 |
| 15.6 | 16.6 | 0x0F | 41.6 | 42.6 | 0x29 |
| 16.6 | 17.6 | 0x10 | 42.6 | 43.6 | 0x2A |
| 17.6 | 18.6 | 0x11 | 43.6 | 44.6 | 0x2B |

**Table 21-8. FDIV values for various BUSCLK Frequencies**

| BUSCLK Frequency (MHz) | | FDIV[5:0] | | BUSCLK Frequency (MHz) | | FDIV[5:0] |
|---|---|---|---|---|---|---|
| MIN[1] | MAX[2] | | | MIN[1] | MAX[2] | |
| 18.6 | 19.6 | 0x12 | | 44.6 | 45.6 | 0x2C |
| 19.6 | 20.6 | 0x13 | | 45.6 | 46.6 | 0x2D |
| 20.6 | 21.6 | 0x14 | | 46.6 | 47.6 | 0x2E |
| 21.6 | 22.6 | 0x15 | | 47.6 | 48.6 | 0x2F |
| 22.6 | 23.6 | 0x16 | | 48.6 | 49.6 | 0x30 |
| 23.6 | 24.6 | 0x17 | | 49.6 | 50.6 | 0x31 |
| 24.6 | 25.6 | 0x18 | | | | |
| 25.6 | 26.6 | 0x19 | | | | |

1 BUSCLK is Greater Than this value.

2 BUSCLK is Less Than or Equal to this value.

### 21.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN[1:0] | | RNV[5:2] | | | | SEC[1:0] | |
| W | | | | | | | | |
| Reset | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] |

= Unimplemented or Reserved

**Figure 21-6. Flash Security Register (FSEC)**

1 Loaded from Flash configuration field, during reset sequence.

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0xFF_FE0F located in P-Flash memory (see Table 21-4) as indicated by reset condition F in Figure 21-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 21-9. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 21-10. |
| 5–2<br>RNV[5:2] | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 21-11. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 21-10. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED[1] |
| 10 | ENABLED |
| 11 | DISABLED |

1 Preferred KEYEN state to disable backdoor key access.

**Table 21-11. Flash Security States**

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01 | SECURED[1] |
| 10 | UNSECURED |
| 11 | SECURED |

1 Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 21.5.

### 21.3.2.3 Flash CCOB Index Register (FCCOBIX)

The FCCOBIX register is used to indicate the amount of parameters loaded into the FCCOB registers for Flash memory operations.

Offset Module Base + 0x0002

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CCOBIX[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-7. FCCOB Index Register (FCCOBIX)**

CCOBIX bits are readable and writable while remaining bits read 0 and are not writable.

**Table 21-12. FCCOBIX Field Descriptions**

| Field | Description |
|---|---|
| 2–0<br>CCOBIX[1:0] | **Common Command Register Index**— The CCOBIX bits are used to indicate how many words of the FCCOB register array are being read or written to. See Section 21.3.2.13, “Flash Common Command Object Registers (FCCOB)“,” for more details. |

### 21.3.2.4 Flash Protection Status Register (FPSTAT)

This Flash register holds the status of the Protection Override feature.

Offset Module Base + 0x0003

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOVRD | 0 | 0 | 0 | 0 | 0 | 0 | WSTATACK |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented or Reserved

**Figure 21-8. Flash Protection Status Register (FPSTAT)**

All bits in the FPSTAT register are readable but are not writable.

**Table 21-13. FPSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOVRD | **Flash Protection Override Status** — The FPOVRD bit indicates if the Protection Override feature is currently enabled. See Section 21.4.7.17, “Protection Override Command” for more details.<br>0 Protection is not overridden<br>1 Protection is overridden, contents of registers FPROT and/or DFPROT (and effective protection limits determined by their current contents) were determined during execution of command Protection Override |
| 0<br>WSTATACK | **Wait-State Switch Acknowledge** — The WSTATACK bit indicates that the wait-state configuration is effectively set according to the value configured on bits FCNFG[WSTAT] (see Section 21.3.2.5, “Flash Configuration Register (FCNFG)”). WSTATACK bit is cleared when a change in FCNFG[WSTAT] is requested by writing to those bits, and is set when the Flash has effectively switched to the new wait-state configuration. The application must check the status of WSTATACK bit to make sure it reads as 1 before changing the frequency setup (see Section 21.4.3, “Flash Block Read Access”).<br>0 Wait-State switch is pending, Flash reads are still happening according to the previous value of FCNFG[WSTAT]<br>1 Wait-State switch is complete, Flash reads are already working according to the value set on FCNFG[WSTAT] |

### 21.3.2.5 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash command complete interrupt, control generation of wait-states and forces ECC faults on Flash array read access from the CPU.

Offset Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIE | 0 | ERSAREQ | IGNSF | WSTAT[1:0] | | FDFD | FSFD |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-9. Flash Configuration Register (FCNFG)**

CCIE, IGNSF, WSTAT, FDFD, and FSFD bits are readable and writable, ERSAREQ bit is read only, and remaining bits read 0 and are not writable.

**Table 21-14. FCNFG Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIE | **Command Complete Interrupt Enable** — The CCIE bit controls interrupt generation when a Flash command has completed.<br>0 Command complete interrupt disabled<br>1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set (see Section 21.3.2.7) |
| 5<br>ERSAREQ | **Erase All Request** — Requests the Memory Controller to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. Refer to the Reference Manual for assertion of the *soc_erase_all_req* input to the FTMRZ module.<br>0 No request or request complete<br>1 Request to:<br>a) run the Erase All Blocks command<br>b) verify the erased state<br>c) program the security byte in the Flash Configuration Field to the unsecure state<br>d) release MCU security by setting the SEC field of the FSEC register to the unsecure state as defined in Table 21-9 of Section 21.3.2.2.<br>The ERSAREQ bit sets to 1 when *soc_erase_all_req* is asserted, CCIF=1 and the Memory Controller starts executing the sequence. ERSAREQ will be reset to 0 by the Memory Controller when the operation is completed (see Section 21.4.7.7.1). |
| 4<br>IGNSF | **Ignore Single Bit Fault** — The IGNSF controls single bit fault reporting in the FERSTAT register (see Section 21.3.2.8).<br>0 All single bit faults detected during array reads are reported<br>1 Single bit faults detected during array reads are not reported and the single bit fault interrupt will not be generated |
| 3–2<br>WSTAT[1:0] | **Wait State control bits** — The WSTAT[1:0] bits define how many wait-states are inserted on each read access to the Flash as shown on Table 21-15.Right after reset the maximum amount of wait-states is set, to be later re-configured by the application if needed. Depending on the system operating frequency being used the number of wait-states can be reduced or disabled, please refer to the Data Sheet for details. For additional information regarding the procedure to change this configuration please see Section 21.4.3. The WSTAT[1:0] bits should not be updated while the Flash is executing a command (CCIF=0); if that happens the value of this field will not change and no action will take place. |

**Table 21-14. FCNFG Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FDFD | **Force Double Bit Fault Detect** — The FDFD bit allows the user to simulate a double bit fault during Flash array read operations. The FDFD bit is cleared by writing a 0 to FDFD.<br>0 Flash array read operations will set the DFDF flag in the FERSTAT register only if a double bit fault is detected<br>1 Any Flash array read operation will force the DFDF flag in the FERSTAT register to be set (see Section 21.3.2.7) |
| 0<br>FSFD | **Force Single Bit Fault Detect** — The FSFD bit allows the user to simulate a single bit fault during Flash array read operations and check the associated interrupt routine. The FSFD bit is cleared by writing a 0 to FSFD.<br>0 Flash array read operations will set the SFDIF flag in the FERSTAT register only if a single bit fault is detected<br>1 Flash array read operation will force the SFDIF flag in the FERSTAT register to be set (see Section 21.3.2.7) and an interrupt will be generated as long as the SFDIE interrupt enable in the FERCNFG register is set (see Section 21.3.2.6) |

**Table 21-15. Flash Wait-States control**

| WSTAT[1:0] | Wait-State configuration |
|---|---|
| 00 | ENABLED, maximum number of cycles[1] |
| 01 | reserved[2] |
| 10 | reserved[2] |
| 11 | DISABLED |

1 Reset condition. For a target of 100MHz core frequency / 50MHz bus frequency the maximum number required is 1 cycle.

2 Value will read as 01 or 10, as written. In the current implementation the Flash will behave the same as 00 (wait-states enabled, maximum number of cycles).

### 21.3.2.6 Flash Error Configuration Register (FERCNFG)

The FERCNFG register enables the Flash error interrupts for the FERSTAT flags.

Offset Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SFDIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-10. Flash Error Configuration Register (FERCNFG)**

All assigned bits in the FERCNFG register are readable and writable.

**Table 21-16. FERCNFG Field Descriptions**

| Field | Description |
|---|---|
| 0<br>SFDIE | **Single Bit Fault Detect Interrupt Enable** — The SFDIE bit controls interrupt generation when a single bit fault is detected during a Flash block read operation.<br>0 SFDIF interrupt disabled whenever the SFDIF flag is set (see Section 21.3.2.8)<br>1 An interrupt will be requested whenever the SFDIF flag is set (see Section 21.3.2.8) |

### 21.3.2.7 Flash Status Register (FSTAT)

The FSTAT register reports the operational status of the Flash module.

Offset Module Base + 0x0006

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT[1:0] | |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0[1] | 0[1] |

= Unimplemented or Reserved

**Figure 21-11. Flash Status Register (FSTAT)**

[1] Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence (see Section 21.6).

CCIF, ACCERR, and FPVIOL bits are readable and writable, MGBUSY and MGSTAT bits are readable but not writable, while remaining bits read 0 and are not writable.

**Table 21-17. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CCIF | **Command Complete Interrupt Flag** — The CCIF flag indicates that a Flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation.<br>0 Flash command in progress<br>1 Flash command has completed |
| 5<br>ACCERR | **Flash Access Error Flag** — The ACCERR bit indicates an illegal access has occurred to the Flash memory caused by either a violation of the command write sequence (see Section 21.4.5.2) or issuing an illegal Flash command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to ACCERR. Writing a 0 to the ACCERR bit has no effect on ACCERR.<br>0 No access error detected<br>1 Access error detected |
| 4<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL bit indicates an attempt was made to program or erase an address in a protected area of P-Flash or EEPROM memory during a command write sequence. The FPVIOL bit is cleared by writing a 1 to FPVIOL. Writing a 0 to the FPVIOL bit has no effect on FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected<br>1 Protection violation detected |
| 3<br>MGBUSY | **Memory Controller Busy Flag** — The MGBUSY flag reflects the active state of the Memory Controller.<br>0 Memory Controller is idle<br>1 Memory Controller is busy executing a Flash command (CCIF = 0) |

**Table 21-17. FSTAT Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. The MGSTAT bits are cleared automatically at the start of the execution of a Flash command. See Section 21.4.7, “Flash Command Description,” and Section 21.6, “Initialization” for details. |

### 21.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | DFDF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 21-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 1<br>DFDF | **Double Bit Fault Detect Flag** — The setting of the DFDF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation returning invalid data was attempted on a Flash block that was under a Flash command operation.[1] The DFDF flag is cleared by writing a 1 to DFDF. Writing a 0 to DFDF has no effect on DFDF.[2]<br>0 No double bit fault detected<br>1 Double bit fault detected or a Flash array read operation returning invalid data was attempted while command running. See Section 21.4.3, “Flash Block Read Access” for details |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation returning invalid data was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or a Flash array read operation returning invalid data was attempted while command running |

1 In case of ECC errors the corresponding flag must be cleared for the proper setting of any further error, i.e. any new error will only be indicated properly when DFDF and/or SFDIF are clear at the time the error condition is detected.

2 There is a one cycle delay in storing the ECC DFDF and SFDIF fault flags in this register. At least one NOP is required after a flash memory read before checking FERSTAT for the occurrence of ECC errors.

### 21.3.2.9 P-Flash Protection Register (FPROT)

The FPROT register defines which P-Flash sectors are protected against program and erase operations.

Offset Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FPOPEN | RNV6 | FPHDIS | FPHS[1:0] | | FPLDIS | FPLS[1:0] | |
| W | | | | | | | | |
| Reset | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] |

= Unimplemented or Reserved

**Figure 21-13. Flash Protection Register (FPROT)**

[1] Loaded from Flash configuration field, during reset sequence.

The (unreserved) bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased(see Section 21.3.2.9.1, “P-Flash Protection Restrictions,” and Table 21-23).

During the reset sequence, the FPROT register is loaded with the contents of the P-Flash protection byte in the Flash configuration field at global address 0xFF_FE0C located in P-Flash memory (see Table 21-4) as indicated by reset condition ‘F’ in Figure 21-13. To change the P-Flash protection that will be loaded during the reset sequence, the upper sector of the P-Flash memory must be unprotected, then the P-Flash protection byte must be reprogrammed. If a double bit fault is detected while reading the P-Flash phrase containing the P-Flash protection byte during the reset sequence, the FPOPEN bit will be cleared and remaining bits in the FPROT register will be set to leave the P-Flash memory fully protected.

Trying to alter data in any protected area in the P-Flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a P-Flash block is not possible if any of the P-Flash sectors contained in the same P-Flash block are protected.

**Table 21-19. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Flash Protection Operation Enable** — The FPOPEN bit determines the protection function for program or erase operations as shown in Table 21-20 for the P-Flash block.<br>0 When FPOPEN is clear, the FPHDIS and FPLDIS bits define unprotected address ranges as specified by the corresponding FPHS and FPLS bits<br>1 When FPOPEN is set, the FPHDIS and FPLDIS bits enable protection for the address range specified by the corresponding FPHS and FPLS bits |
| 6<br>RNV[6] | **Reserved Nonvolatile Bit** — The RNV bit should remain in the erased state for future enhancements. |
| 5<br>FPHDIS | **Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory ending with global address 0xFF_FFFF.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 4–3<br>FPHS[1:0] | **Flash Protection Higher Address Size** — The FPHS bits determine the size of the protected/unprotected area in P-Flash memory as shown inTable 21-21. The FPHS bits can only be written to while the FPHDIS bit is set. |

**Table 21-19. FPROT Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>FPLDIS | **Flash Protection Lower Address Range Disable** — The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the P-Flash memory beginning with global address 0xFF_8000.<br>0 Protection/Unprotection enabled<br>1 Protection/Unprotection disabled |
| 1–0<br>FPLS[1:0] | **Flash Protection Lower Address Size** — The FPLS bits determine the size of the protected/unprotected area in P-Flash memory as shown in Table 21-22. The FPLS bits can only be written to while the FPLDIS bit is set. |

**Table 21-20. P-Flash Protection Function**

| FPOPEN | FPHDIS | FPLDIS | Function[1] |
|---|---|---|---|
| 1 | 1 | 1 | No P-Flash Protection |
| 1 | 1 | 0 | Protected Low Range |
| 1 | 0 | 1 | Protected High Range |
| 1 | 0 | 0 | Protected High and Low Ranges |
| 0 | 1 | 1 | Full P-Flash Memory Protected |
| 0 | 1 | 0 | Unprotected Low Range |
| 0 | 0 | 1 | Unprotected High Range |
| 0 | 0 | 0 | Unprotected High and Low Ranges |

1 For range sizes, refer to Table 21-21 and Table 21-22.

**Table 21-21. P-Flash Protection Higher Address Range**

| FPHS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0xFF_F800–0xFF_FFFF | 2 KB |
| 01 | 0xFF_F000–0xFF_FFFF | 4 KB |
| 10 | 0xFF_E000–0xFF_FFFF | 8 KB |
| 11 | 0xFF_C000–0xFF_FFFF | 16 KB |

**Table 21-22. P-Flash Protection Lower Address Range**

| FPLS[1:0] | Global Address Range | Protected Size |
|---|---|---|
| 00 | 0xFF_8000–0xFF_83FF | 1 KB |
| 01 | 0xFF_8000–0xFF_87FF | 2 KB |
| 10 | 0xFF_8000–0xFF_8FFF | 4 KB |
| 11 | 0xFF_8000–0xFF_9FFF | 8 KB |

All possible P-Flash protection scenarios are shown in Figure 21-14 . Although the protection scheme is loaded from the Flash memory at global address 0xFF_FE0C during the reset sequence, it can be changed by the user. The P-Flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

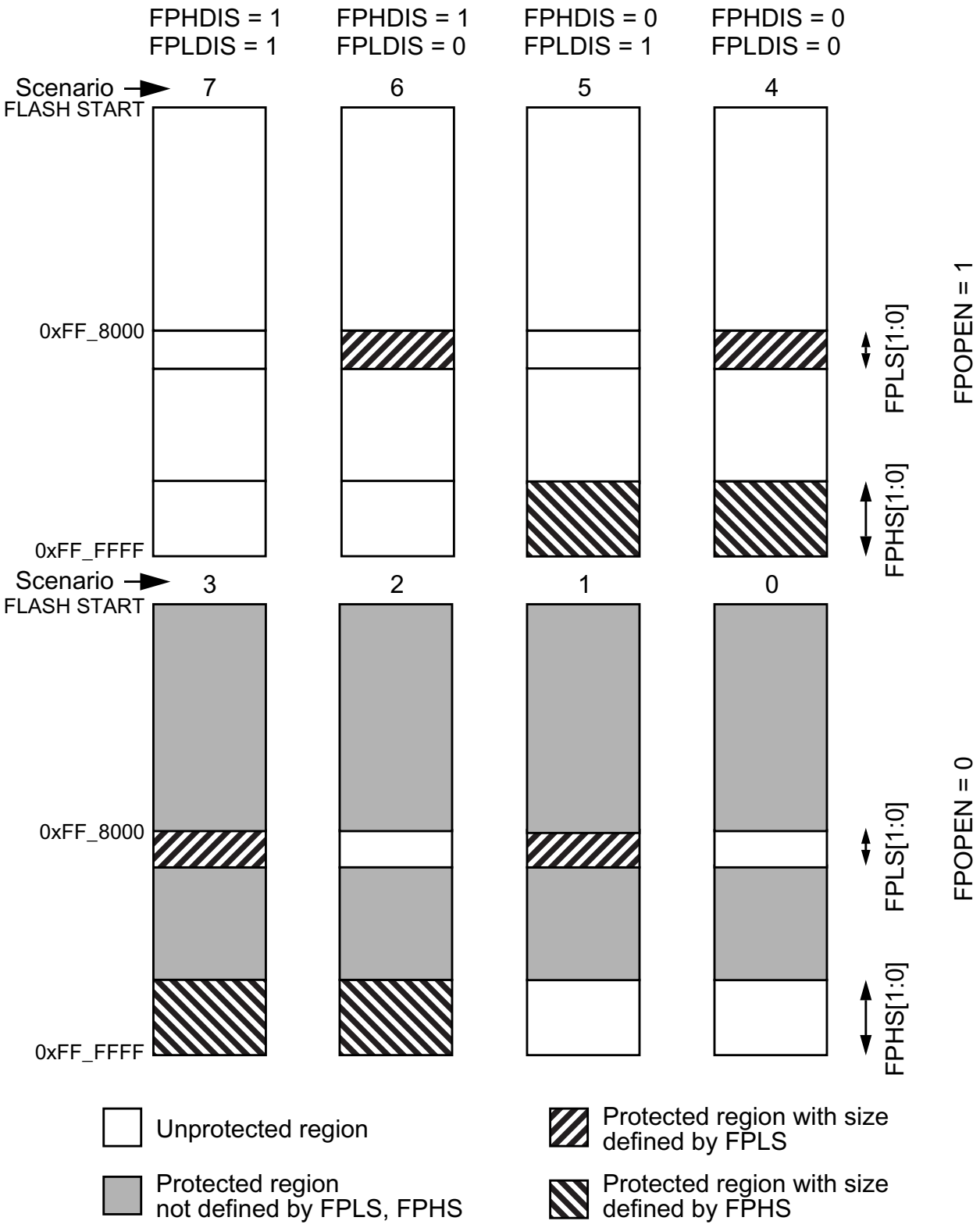



**Figure 21-14. P-Flash Protection Scenarios**

#### 21.3.2.9.1 P-Flash Protection Restrictions

The general guideline is that P-Flash protection can only be added and not removed. Table 21-23 specifies all valid transitions between P-Flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPHS and FPLS bit descriptions for additional restrictions.

**Table 21-23. P-Flash Protection Scenario Transitions**

| From Protection Scenario | To Protection Scenario[1] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | X | X | X | X | | | | |
| 1 | | X | | X | | | | |
| 2 | | | X | X | | | | |
| 3 | | | | X | | | | |
| 4 | | | | X | X | | | |
| 5 | | | X | X | X | X | | |
| 6 | | X | | X | X | | X | |
| 7 | X | X | X | X | X | X | X | X |

1 Allowed transitions marked with X, see Figure 21-14 for a definition of the scenarios.

### 21.3.2.10 EEPROM Protection Register (DFPROT)

The DFPROT register defines which EEPROM sectors are protected against program and erase operations.

Offset Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DPOPEN | 0 | DPS[5:0] | | | | | |
| W | | | | | | | | |
| Reset | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] |

**Figure 21-15. EEPROM Protection Register (DFPROT)**

1 Loaded from Flash configuration field, during reset sequence.

The (unreserved) bits of the DFPROT register are writable with the restriction that protection can be added but not removed. Writes must increase the DPS value and the DPOPEN bit can only be written from 1 (protection disabled) to 0 (protection enabled). If the DPOPEN bit is set, the state of the DPS bits is irrelevant.

During the reset sequence, fields DPOPEN and DPS of the DFPROT register are loaded with the contents of the EEPROM protection byte in the Flash configuration field at global address 0xFF_FE0D located in P-Flash memory (see Table 21-4) as indicated by reset condition F in Table 21-25. To change the

EEPROM protection that will be loaded during the reset sequence, the P-Flash sector containing the EEPROM protection byte must be unprotected, then the EEPROM protection byte must be programmed. If a double bit fault is detected while reading the P-Flash phrase containing the EEPROM protection byte during the reset sequence, the DPOPEN bit will be cleared and DPS bits will be set to leave the EEPROM memory fully protected.

Trying to alter data in any protected area in the EEPROM memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. Block erase of the EEPROM memory is not possible if any of the EEPROM sectors are protected.

**Table 21-24. DFPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DPOPEN | **EEPROM Protection Control**<br>0 Enables EEPROM memory protection from program and erase with protected address range defined by DPS bits<br>1 Disables EEPROM memory protection from program and erase |
| 5–0<br>DPS[5:0] | **EEPROM Protection Size** — The DPS[5:0] bits determine the size of the protected area in the EEPROM memory, this size increase in step of 32 bytes, as shown in Table 21-25 . |

**Table 21-25. EEPROM Protection Address Range**

| DPS[5:0] | Global Address Range | Protected Size |
|---|---|---|
| 000000 | 0x10_0000 – 0x10_001F | 32 bytes |
| 000001 | 0x10_0000 – 0x10_003F | 64 bytes |
| 000010 | 0x10_0000 – 0x10_005F | 96 bytes |
| 000011 | 0x10_0000 – 0x10_007F | 128 bytes |
| 000100 | 0x10_0000 – 0x10_009F | 160 bytes |
| 000101 | 0x10_0000 – 0x10_00BF | 192 bytes |
| The Protection Size goes on enlarging in step of 32 bytes, for each DPS value increasing of one.<br>.<br>.<br>. | | |
| 111111 | 0x10_0000 – 0x10_07FF | 2,048 bytes |

### 21.3.2.11 Flash Option Register (FOPT)

The FOPT register is the Flash option register.

Offset Module Base + 0x000A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NV[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] | F[1] |

= Unimplemented or Reserved

**Figure 21-16. Flash Option Register (FOPT)**

[1] Loaded from Flash configuration field, during reset sequence.

All bits in the FOPT register are readable but are not writable.

During the reset sequence, the FOPT register is loaded from the Flash nonvolatile byte in the Flash configuration field at global address 0xFF_FE0E located in P-Flash memory (see Table 21-4) as indicated by reset condition F in Figure 21-16. If a double bit fault is detected while reading the P-Flash phrase containing the Flash nonvolatile byte during the reset sequence, all bits in the FOPT register will be set.

**Table 21-26. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>NV[7:0] | **Nonvolatile Bits** — The NV[7:0] bits are available as nonvolatile bits. Refer to the device user guide for proper use of the NV bits. |

### 21.3.2.12 Flash Reserved1 Register (FRSV1)

This Flash register is reserved for factory testing.

Offset Module Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 21-17. Flash Reserved1 Register (FRSV1)**

All bits in the FRSV1 register read 0 and are not writable.

### 21.3.2.13 Flash Common Command Object Registers (FCCOB)

The FCCOB is an array of six words. Byte wide reads and writes are allowed to the FCCOB registers.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-18. Flash Common Command Object 0 High Register (FCCOB0HI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-19. Flash Common Command Object 0 Low Register (FCCOB0LO)**

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-20. Flash Common Command Object 1 High Register (FCCOB1HI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-21. Flash Common Command Object 1 Low Register (FCCOB1LO)**

Offset Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-22. Flash Common Command Object 2 High Register (FCCOB2HI)**

Offset Module Base + 0x0011

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-23. Flash Common Command Object 2 Low Register (FCCOB2LO)**

Offset Module Base + 0x0012

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-24. Flash Common Command Object 3 High Register (FCCOB3HI)**

Offset Module Base + 0x0013

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-25. Flash Common Command Object 3 Low Register (FCCOB3LO)**

Offset Module Base + 0x0014

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-26. Flash Common Command Object 4 High Register (FCCOB4HI)**

Offset Module Base + 0x0015

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-27. Flash Common Command Object 4 Low Register (FCCOB4LO)**

Offset Module Base + 0x0016

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-28. Flash Common Command Object 5 High Register (FCCOB5HI)**

Offset Module Base + 0x0017

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-29. Flash Common Command Object 5 Low Register (FCCOB5LO)**

### 21.3.2.13.1 FCCOB - NVM Command Mode

NVM command mode uses the FCCOB registers to provide a command code and its relevant parameters to the Memory Controller. The user first sets up all required FCCOB fields and then initiates the command's execution by writing a 1 to the CCIF bit in the FSTAT register (a 1 written by the user clears the CCIF command completion flag to 0). When the user clears the CCIF bit in the FSTAT register all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (as evidenced by the Memory Controller returning CCIF to 1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in NVM command mode is shown in Table 21-27. The return values are available for reading after the CCIF flag in the FSTAT register has been returned to 1 by the Memory Controller. The value written to the FCCOBIX field must reflect the amount of CCOB words loaded for command execution.

Table 21-27 shows the generic Flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific Flash command. For details on the FCCOB settings required by each command, see the Flash command descriptions in Section 21.4.7.

**Table 21-27. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Register | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|---|
| 000 | FCCOB0 | HI | FCMD[7:0] defining Flash command |
| | | LO | Global address [23:16] |
| 001 | FCCOB1 | HI | Global address [15:8] |
| | | LO | Global address [7:0] |
| 010 | FCCOB2 | HI | Data 0 [15:8] |
| | | LO | Data 0 [7:0] |

**Table 21-27. FCCOB - NVM Command Mode (Typical Usage)**

| CCOBIX[2:0] | Register | Byte | FCCOB Parameter Fields (NVM Command Mode) |
|---|---|---|---|
| 011 | FCCOB3 | HI | Data 1 [15:8] |
| | | LO | Data 1 [7:0] |
| 100 | FCCOB4 | HI | Data 2 [15:8] |
| | | LO | Data 2 [7:0] |
| 101 | FCCOB5 | HI | Data 3 [15:8] |
| | | LO | Data 3 [7:0] |

## 21.4 Functional Description

### 21.4.1 Modes of Operation

The FTMRZ64K2K module provides the modes of operation normal and special . The operating mode is determined by module-level inputs and affects the FCLKDIV, FCNFG, and DFPROT registers (see Table 21-29).

### 21.4.2 IFR Version ID Word

The version ID word is stored in the IFR at address 0x1F_C0B6. The contents of the word are defined in Table 21-28.

**Table 21-28. IFR Version ID Fields**

| [15:4] | [3:0] |
|---|---|
| Reserved | VERNUM |

- VERNUM: Version number. The first version is number 0b_0001 with both 0b_0000 and 0b_1111 meaning ‘none’.

### 21.4.3 Flash Block Read Access

If data read from the Flash block results in a double-bit fault ECC error (meaning that data is detected to be in error and cannot be corrected), the read data will be tagged as invalid during that access (please look into the Reference Manual for details). Forcing the DFDF status bit by setting FDFD (see Section 21.3.2.5) has effect only on the DFDF status bit value and does not result in an invalid access.

To guarantee the proper read timing from the Flash array, the FTMRZ64K2K FMU will control (i.e. pause) the S12Z core accesses, considering that the MCU can be configured to fetch data at a faster frequency than the Flash block can support. Right after reset the FTMRZ64K2K FMU will be configured to run with the maximum amount of wait-states enabled; if the user application is setup to run at a slower frequency the control bits FCNFG[WSTAT] (see Section 21.3.2.5) can be configured by the user to disable the generation of wait-states, so it does not impose a performance penalty to the system if the read timing of the S12Z core is setup to be within the margins of the Flash block. For a definition of the frequency values where wait-states can be disabled please look into the Reference Manual.

The following sequence must be followed when the transition from a higher frequency to a lower frequency is going to happen:

- Flash resets with wait-states enabled;
- system frequency must be configured to the lower target;
- user writes to FNCNF[WSTAT] to disable wait-states;
- user reads the value of FPSTAT[WSTATACK], the new wait-state configuration will be effective when it reads as 1;
- user must re-write FCLKDIV to set a new value based on the lower frequency.

The following sequence must be followed on the contrary direction, going from a lower frequency to a higher frequency:

- user writes to FCNFG[WSTAT] to enable wait-states;
- user reads the value of FPSTAT[WSTATACK], the new wait-state configuration will be effective when it reads as 1;
- user must re-write FCLKDIV to set a new value based on the higher frequency;
- system frequency must be set to the upper target.

**CAUTION**

If the application is going to require the frequency setup to change, the value to be loaded on register FCLKDIV will have to be updated according to the new frequency value. In this scenario the application must take care to avoid locking the value of the FCLKDIV register: bit FDIVLCK must not be set if the value to be loaded on FDIV is going to be re-written, otherwise a reset is going to be required. Please refer to Section 21.3.2.1, “Flash Clock Divider Register (FCLKDIV) and Section 21.4.5.1, “Writing the FCLKDIV Register.

### 21.4.4 Internal NVM resource

IFR is an internal NVM resource readable by CPU . The IFR fields are shown in Table 21-5.

The NVM Resource Area global address map is shown in Table 21-6.

### 21.4.5 Flash Command Operations

Flash command operations are used to modify Flash memory contents.

The next sections describe:

- How to write the FCLKDIV register that is used to generate a time base (FCLK) derived from BUSCLK for Flash program and erase command operations
- The command write sequence used to set Flash command parameters and launch execution
- Valid Flash commands available for execution, according to MCU functional mode and MCU security state.

#### 21.4.5.1 Writing the FCLKDIV Register

Prior to issuing any Flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide BUSCLK down to a target FCLK of 1 MHz. Table 21-8 shows recommended values for the FDIV field based on BUSCLK frequency.

**NOTE**

Programming or erasing the Flash memory cannot be performed if the bus clock runs at less than 0.8 MHz. Setting FDIV too high can destroy the Flash memory due to overstress. Setting FDIV too low can result in incomplete programming or erasure of the Flash memory cells.

When the FCLKDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any Flash program or erase command loaded during a command write sequence will not execute and the ACCERR bit in the FSTAT register will set.

#### 21.4.5.2 Command Write Sequence

The Memory Controller will launch all valid Flash commands entered using a command write sequence.

Before launching a command, the ACCERR and FPVIOL bits in the FSTAT register must be clear (see Section 21.3.2.7) and the CCIF flag should be tested to determine the status of the current command write sequence. If CCIF is 0, the previous command write sequence is still active, a new command write sequence cannot be started, and all writes to the FCCOB register are ignored.

##### 21.4.5.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. The CCOBIX bits in the FCCOBIX register must reflect the amount of words loaded into the FCCOB registers (see Section 21.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 21-30.

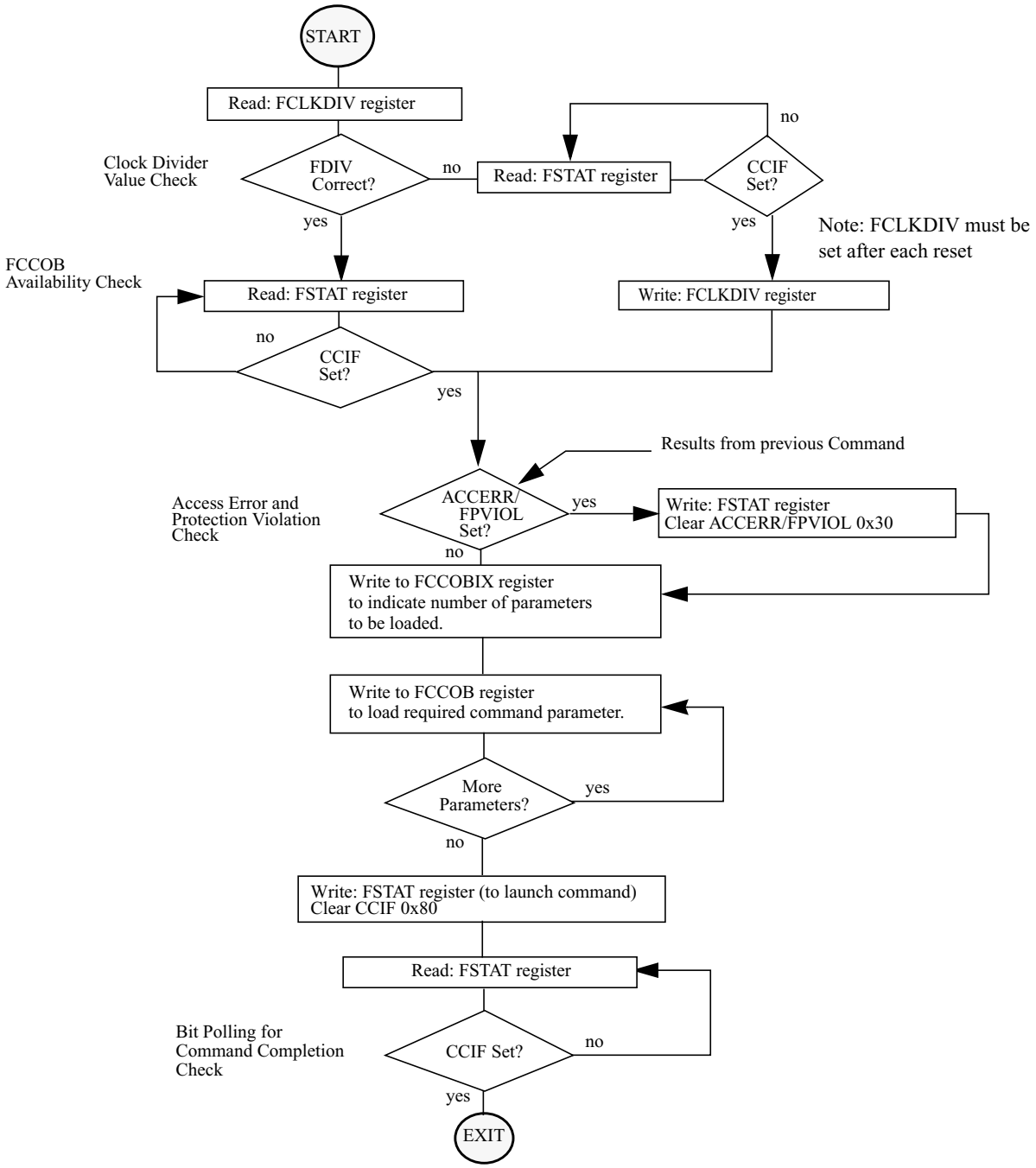



**Figure 21-30. Generic Flash Command Write Sequence Flowchart**

### 21.4.5.3 Valid Flash Module Commands

Table 21-29 present the valid Flash commands, as enabled by the combination of the functional MCU mode (Normal SingleChip NS, Special Singlechip SS) with the MCU security state (Unsecured, Secured).

**Table 21-29. Flash Commands by Mode and Security State**

| FCMD | Command | Unsecured | | Secured | |
|---|---|---|---|---|---|
| | | NS[1] | SS[2] | NS[3] | SS[4] |
| 0x01 | Erase Verify All Blocks | * | * | * | * |
| 0x02 | Erase Verify Block | * | * | * | * |
| 0x03 | Erase Verify P-Flash Section | * | * | * | |
| 0x04 | Read Once | * | * | * | |
| 0x06 | Program P-Flash | * | * | * | |
| 0x07 | Program Once | * | * | * | |
| 0x08 | Erase All Blocks | | * | | * |
| 0x09 | Erase Flash Block | * | * | * | |
| 0x0A | Erase P-Flash Sector | * | * | * | |
| 0x0B | Unsecure Flash | | * | | * |
| 0x0C | Verify Backdoor Access Key | * | | * | |
| 0x0D | Set User Margin Level | * | * | * | |
| 0x0E | Set Field Margin Level | | * | | |
| 0x10 | Erase Verify EEPROM Section | * | * | * | |
| 0x11 | Program EEPROM | * | * | * | |
| 0x12 | Erase EEPROM Sector | * | * | * | |
| 0x13 | Protection Override | * | * | * | * |

1 Unsecured Normal Single Chip mode

2 Unsecured Special Single Chip mode.

3 Secured Normal Single Chip mode.

4 Secured Special Single Chip mode.

### 21.4.5.4 P-Flash Commands

Table 21-30 summarizes the valid P-Flash commands along with the effects of the commands on the P-Flash block and other resources within the Flash module.

**Table 21-30. P-Flash Commands**

| FCMD | Command | Function on P-Flash Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all P-Flash (and EEPROM) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that a P-Flash block is erased. |
| 0x03 | Erase Verify P-Flash Section | Verify that a given number of words starting at the address provided are erased. |
| 0x04 | Read Once | Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block that was previously programmed using the Program Once command. |
| 0x06 | Program P-Flash | Program a phrase in a P-Flash block. |
| 0x07 | Program Once | Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block that is allowed to be programmed only once. |
| 0x08 | Erase All Blocks | Erase all P-Flash (and EEPROM) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the DPOPEN bit in the DFPROT register are set prior to launching the command. |
| 0x09 | Erase Flash Block | Erase a P-Flash (or EEPROM) block.<br>An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command. |
| 0x0A | Erase P-Flash Sector | Erase all bytes in a P-Flash sector. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all P-Flash (and EEPROM) blocks and verifying that all P-Flash (and EEPROM) blocks are erased. |
| 0x0C | Verify Backdoor Access Key | Supports a method of releasing MCU security by verifying a set of security keys. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for all P-Flash blocks. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for all P-Flash blocks (special modes only). |
| 0x13 | Protection Override | Supports a mode to temporarily override Protection configuration (for P-Flash and/or EEPROM) by verifying a key. |

### 21.4.5.5 EEPROM Commands

Table 21-31 summarizes the valid EEPROM commands along with the effects of the commands on the EEPROM block.

**Table 21-31. EEPROM Commands**

| FCMD | Command | Function on EEPROM Memory |
|---|---|---|
| 0x01 | Erase Verify All Blocks | Verify that all EEPROM (and P-Flash) blocks are erased. |
| 0x02 | Erase Verify Block | Verify that the EEPROM block is erased. |
| 0x08 | Erase All Blocks | Erase all EEPROM (and P-Flash) blocks.<br>An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the DPOPEN bit in the DFPROT register are set prior to launching the command. |
| 0x09 | Erase Flash Block | Erase a EEPROM (or P-Flash) block.<br>An erase of the full EEPROM block is only possible when DPOPEN bit in the DFPROT register is set prior to launching the command. |
| 0x0B | Unsecure Flash | Supports a method of releasing MCU security by erasing all EEPROM (and P-Flash) blocks and verifying that all EEPROM (and P-Flash) blocks are erased. |
| 0x0D | Set User Margin Level | Specifies a user margin read level for the EEPROM block. |
| 0x0E | Set Field Margin Level | Specifies a field margin read level for the EEPROM block (special modes only). |
| 0x10 | Erase Verify EEPROM Section | Verify that a given number of words starting at the address provided are erased. |
| 0x11 | Program EEPROM | Program up to four words in the EEPROM block. |
| 0x12 | Erase EEPROM Sector | Erase all bytes in a sector of the EEPROM block. |
| 0x13 | Protection Override | Supports a mode to temporarily override Protection configuration (for P-Flash and/or EEPROM) by verifying a key. |

### 21.4.6 Allowed Simultaneous P-Flash and EEPROM Operations

Only the operations marked ‘OK’ in Table 21-32 are permitted to be run simultaneously on the Program Flash and EEPROM blocks. Some operations cannot be executed simultaneously because certain hardware resources are shared by the two memories. The priority has been placed on permitting Program Flash reads while program and erase operations execute on the EEPROM, providing read (P-Flash) while write (EEPROM) functionality. Any attempt to access P-Flash and EEPROM simultaneously when it is not allowed will result in an illegal access that will trigger a machine exception in the CPU (please look into the Reference Manual for details). Please note that during the execution of each command there is a period, before the operation in the Flash array actually starts, where reading is allowed and valid data is returned. Even if the simultaneous operation is marked as not allowed the Flash will report an illegal access only in the cycle the read collision actually happens, maximizing the time the array is available for reading.

**Table 21-32. Allowed P-Flash and EEPROM Simultaneous Operations**

| | EEPROM | | | | |
|---|---|---|---|---|---|
| **Program Flash** | **Read** | **Margin Read[2]** | **Program** | **Sector Erase** | **Mass Erase[2]** |
| **Read** | OK[1] | OK | OK | OK | |
| **Margin Read[2]** | | | | | |
| **Program** | | | | | |
| **Sector Erase** | | | | | |
| **Mass Erase[3]** | | | | | OK |

1 Strictly speaking, only one read of either the P-Flash or EEPROM can occur at any given instant, but the memory controller will transparently arbitrate P-Flash and EEPROM accesses giving uninterrupted read access whenever possible.

2 A 'Margin Read' is any read after executing the margin setting commands 'Set User Margin Level' or 'Set Field Margin Level' with anything but the 'normal' level specified. See the Note on margin settings in Section 21.4.7.12 and Section 21.4.7.13.

3 The 'Mass Erase' operations are commands 'Erase All Blocks' and 'Erase Flash Block'

## 21.4.7 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation may return invalid data resulting in an illegal access (as described on Section 21.4.6).

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 21.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 21.4.7.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and EEPROM blocks have been erased.

**Table 21-33. Erase Verify All Blocks Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed. If all blocks are not erased, it means blank check failed, both MGSTAT bits will be set.

**Table 21-34. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the reador if blank check failed . |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read or if blank check failed. |

### 21.4.7.2 Erase Verify Block Command

The Erase Verify Block command allows the user to verify that an entire P-Flash or EEPROM block has been erased.

**Table 21-35. Erase Verify Block Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x02 | Global address [23:16] to identify Flash block |
| FCCOB1 | Global address [15:0] to identify Flash block | |

Upon clearing CCIF to launch the Erase Verify Block command, the Memory Controller will verify that the selected P-Flash or EEPROM block is erased. The CCIF flag will set after the Erase Verify Block operation has completed.If the block is not erased, it means blank check failed, both MGSTAT bits will be set.

**Table 21-36. Erase Verify Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read or if blank check failed. |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read or if blank check failed. |

### 21.4.7.3 Erase Verify P-Flash Section Command

The Erase Verify P-Flash Section command will verify that a section of code in the P-Flash memory is erased. The Erase Verify P-Flash Section command defines the starting point of the code to be verified and the number of phrases.

**Table 21-37. Erase Verify P-Flash Section Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x03 | Global address [23:16] of a P-Flash block |
| FCCOB1 | Global address [15:0] of the first phrase to be verified | |
| FCCOB2 | Number of phrases to be verified | |

Upon clearing CCIF to launch the Erase Verify P-Flash Section command, the Memory Controller will verify the selected section of Flash memory is erased. The CCIF flag will set after the Erase Verify P-Flash Section operation has completed. If the section is not erased, it means blank check failed, both MGSTAT bits will be set.

**Table 21-38. Erase Verify P-Flash Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | | Set if the requested section crosses a the P-Flash address boundary |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read or if blank check failed. |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read or if blank check failed. |

### 21.4.7.4 Read Once Command

The Read Once command provides read access to a reserved 64 byte field (8 phrases) located in the nonvolatile information register of P-Flash. The Read Once field is programmed using the Program Once command described in Section 21.4.7.6. The Read Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 21-39. Read Once Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x04 | Not Required |
| FCCOB1 | Read Once phrase index (0x0000 - 0x0007) | |
| FCCOB2 | Read Once word 0 value | |

**Table 21-39. Read Once Command FCCOB Requirements**

| Register | FCCOB Parameters |
|---|---|
| FCCOB3 | Read Once word 1 value |
| FCCOB4 | Read Once word 2 value |
| FCCOB5 | Read Once word 3 value |

Upon clearing CCIF to launch the Read Once command, a Read Once phrase is fetched and stored in the FCCOB indexed register. The CCIF flag will set after the Read Once operation has completed. Valid phrase index values for the Read Once command range from 0x0000 to 0x0007. During execution of the Read Once command, any attempt to read addresses within P-Flash block will return invalid data.

**Table 21-40. Read Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid phrase index is supplied |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read |

### 21.4.7.5 Program P-Flash Command

The Program P-Flash operation will program a previously erased phrase in the P-Flash memory using an embedded algorithm.

**CAUTION**

A P-Flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash phrase is not allowed.

**Table 21-41. Program P-Flash Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x06 | Global address [23:16] to identify P-Flash block |
| FCCOB1 | Global address [15:0] of phrase location to be programmed[1] | |
| FCCOB2 | Word 0 program value | |
| FCCOB3 | Word 1 program value | |
| FCCOB4 | Word 2 program value | |
| FCCOB5 | Word 3 program value | |

1 Global address [2:0] must be 000

Upon clearing CCIF to launch the Program P-Flash command, the Memory Controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The CCIF flag will set after the Program P-Flash operation has completed.

**Table 21-42. Program P-Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the global address [17:0] points to a protected area |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 21.4.7.6 Program Once Command

The Program Once command restricts programming to a reserved 64 byte field (8 phrases) in the nonvolatile information register located in P-Flash. The Program Once reserved field can be read using the Read Once command as described in Section 21.4.7.4. The Program Once command must only be issued once since the nonvolatile information register in P-Flash cannot be erased. The Program Once command must not be executed from the Flash block containing the Program Once reserved field to avoid code runaway.

**Table 21-43. Program Once Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x07 | Not Required |
| FCCOB1 | Program Once phrase index (0x0000 - 0x0007) | |
| FCCOB2 | Program Once word 0 value | |
| FCCOB3 | Program Once word 1 value | |
| FCCOB4 | Program Once word 2 value | |
| FCCOB5 | Program Once word 3 value | |

Upon clearing CCIF to launch the Program Once command, the Memory Controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The CCIF flag will remain clear, setting only after the Program Once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index values for the Program Once command range from 0x0000 to 0x0007. During execution of the Program Once command, any attempt to read addresses within P-Flash will return invalid data.

**Table 21-44. Program Once Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 101 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid phrase index is supplied |
| | | Set if the requested phrase has already been programmed[1] |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

1 If a Program Once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the Program Once command will be allowed to execute again on that same phrase.

### 21.4.7.7 Erase All Blocks Command

The Erase All Blocks operation will erase the entire P-Flash and EEPROM memory space.

**Table 21-45. Erase All Blocks Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 21-46. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | FPVIOL | Set if any area of the P-Flash or EEPROM memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

#### 21.4.7.7.1 Erase All Pin

The functionality of the Erase All Blocks command is also available in an uncommanded fashion from the *soc_erase_all_req* input pin on the Flash module. Refer to the Reference Manual for information on control of *soc_erase_all_req*.

The erase-all function requires the clock divider register FCLKDIV (see Section 21.3.2.1) to be loaded before invoking this function using *soc_erase_all_req* input pin. Please refer to the Reference Manual for information about the default value of FCLKDIV in case direct writes to register FCLKDIV are not allowed by the time this feature is invoked. If FCLKDIV is not properly set the erase-all operation will not execute and the ACCERR flag in FSTAT register will set. After the execution of the erase-all function the FCLKDIV register will be reset and the value of register FCLKDIV must be loaded before launching any other command afterwards.

Before invoking the erase-all function using the *soc_erase_all_req* pin, the ACCERR and FPVIOL flags in the FSTAT register must be clear. When invoked from *soc_erase_all_req* the erase-all function will erase all P-Flash memory and EEPROM memory space regardless of the protection settings. If the post-erase verify passes, the routine will then release security by setting the SEC field of the FSEC register to the unsecure state (see Section 21.3.2.2). The security byte in the Flash Configuration Field will be programmed to the unsecure state (see Table 21-9). The status of the erase-all request is reflected in the ERSAREQ bit in the FCNFG register (see Section 21.3.2.5). The ERSAREQ bit in FCNFG will be cleared once the operation has completed and the normal FSTAT error reporting will be available as described inTable 21-47.

At the end of the erase-all sequence Protection will remain configured as it was before executing the erase-all function. If the application requires programming P-Flash and/or EEPROM after the erase-all function completes, the existing protection limits must be taken into account. If protection needs to be disabled the user may need to reset the system right after completing the erase-all function.

**Table 21-47. Erase All Pin Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if command not available in current mode (see Table 21-29) |
| | MGSTAT1 | Set if any errors have been encountered during the erase verify operation, or during the program verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the erase verify operation, or during the program verify operation |

### 21.4.7.8 Erase Flash Block Command

The Erase Flash Block operation will erase all addresses in a P-Flash or EEPROM block.

**Table 21-48. Erase Flash Block Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x09 | Global address [23:16] to identify Flash block |
| FCCOB1 | Global address [15:0] in Flash block to be erased | |

Upon clearing CCIF to launch the Erase Flash Block command, the Memory Controller will erase the selected Flash block and verify that it is erased. The CCIF flag will set after the Erase Flash Block operation has completed.

**Table 21-49. Erase Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied |
| | | Set if the supplied P-Flash address is not phrase-aligned or if the EEPROM address is not word-aligned |
| | FPVIOL | Set if an area of the selected Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 21.4.7.9 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 21-50. Erase P-Flash Sector Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x0A | Global address [23:16] to identify P-Flash block to be erased |
| FCCOB1 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 21.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

**Table 21-51. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 21.4.7.10 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and EEPROM memory space and, if the erase is successful, will release security.

**Table 21-52. Unsecure Flash Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and EEPROM memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 21-53. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | FPVIOL | Set if any area of the P-Flash or EEPROM memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 21.4.7.11 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 21-10). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see Table 21-4). The Verify Backdoor Access Key command must not be executed from the Flash block containing the backdoor comparison key to avoid code runaway.

**Table 21-54. Verify Backdoor Access Key Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x0C | Not required |
| FCCOB1 | Key 0 | |
| FCCOB2 | Key 1 | |
| FCCOB3 | Key 2 | |
| FCCOB4 | Key 3 | |

Upon clearing CCIF to launch the Verify Backdoor Access Key command, the Memory Controller will check the FSEC KEYEN bits to verify that this command is enabled. If not enabled, the Memory Controller sets the ACCERR bit in the FSTAT register and terminates. If the command is enabled, the Memory Controller compares the key provided in FCCOB to the backdoor comparison key in the Flash configuration field with Key 0 compared to 0xFF_FE00, etc. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the Verify Backdoor Access Key command are aborted (set ACCERR) until a reset occurs. The CCIF flag is set after the Verify Backdoor Access Key operation has completed.

**Table 21-55. Verify Backdoor Access Key Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 100 at command launch |
| | | Set if an incorrect backdoor key is supplied |
| | | Set if backdoor key access has not been enabled (KEYEN[1:0] != 10, see Section 21.3.2.2) |
| | | Set if the backdoor key has mismatched since the last reset |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |

### 21.4.7.12 Set User Margin Level Command

The Set User Margin Level command causes the Memory Controller to set the margin level for future read operations of the P-Flash or EEPROM block.

**Table 21-56. Set User Margin Level Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x0D | Global address [23:16] to identify Flash block |
| FCCOB1 | Global address [15:0] to identify Flash block | |
| FCCOB2 | Margin level setting. | |

Upon clearing CCIF to launch the Set User Margin Level command, the Memory Controller will set the user margin level for the targeted block and then set the CCIF flag.

**NOTE**

When the EEPROM block is targeted, the EEPROM user margin levels are applied only to the EEPROM reads. However, when the P-Flash block is targeted, the P-Flash user margin levels are applied to both P-Flash and EEPROM reads. It is not possible to apply user margin levels to the P-Flash block only.

Valid margin level settings for the Set User Margin Level command are defined in Table 21-57.

**Table 21-57. Valid Set User Margin Level Settings**

| FCCOB2 | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level[1] |
| 0x0002 | User Margin-0 Level[2] |

1 Read margin to the erased state

2 Read margin to the programmed state

**Table 21-58. Set User Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |

**NOTE**

User margin levels can be used to check that Flash memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking Flash memory contents at user margin levels, a potential loss of information has been detected.

### 21.4.7.13 Set Field Margin Level Command

The Set Field Margin Level command, valid in special modes only, causes the Memory Controller to set the margin level specified for future read operations of the P-Flash or EEPROM block.

**Table 21-59. Set Field Margin Level Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x0E | Global address [23:16] to identify Flash block |
| FCCOB1 | Global address [15:0] to identify Flash block | |
| FCCOB2 | Margin level setting. | |

Upon clearing CCIF to launch the Set Field Margin Level command, the Memory Controller will set the field margin level for the targeted block and then set the CCIF flag.

**NOTE**

When the EEPROM block is targeted, the EEPROM field margin levels are applied only to the EEPROM reads. However, when the P-Flash block is targeted, the P-Flash field margin levels are applied to both P-Flash and EEPROM reads. It is not possible to apply field margin levels to the P-Flash block only.

Valid margin level settings for the Set Field Margin Level command are defined in Table 21-60.

**Table 21-60. Valid Set Field Margin Level Settings**

| FCCOB2 | Level Description |
|---|---|
| 0x0000 | Return to Normal Level |
| 0x0001 | User Margin-1 Level[1] |
| 0x0002 | User Margin-0 Level[2] |
| 0x0003 | Field Margin-1 Level[1] |
| 0x0004 | Field Margin-0 Level[2] |

1 Read margin to the erased state

2 Read margin to the programmed state

**Table 21-61. Set Field Margin Level Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied see Table 21-3) |
| | | Set if an invalid margin level setting is supplied |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |

**CAUTION**

Field margin levels must only be used during verify of the initial factory programming.

**NOTE**

Field margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking Flash memory contents at field margin levels, the Flash memory contents should be erased and reprogrammed.

### 21.4.7.14 Erase Verify EEPROM Section Command

The Erase Verify EEPROM Section command will verify that a section of code in the EEPROM is erased. The Erase Verify EEPROM Section command defines the starting point of the data to be verified and the number of words.

**Table 21-62. Erase Verify EEPROM Section Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x10 | Global address [23:16] to identify the EEPROM block |
| FCCOB1 | Global address [15:0] of the first word to be verified | |
| FCCOB2 | Number of words to be verified | |

Upon clearing CCIF to launch the Erase Verify EEPROM Section command, the Memory Controller will verify the selected section of EEPROM memory is erased. The CCIF flag will set after the Erase Verify EEPROM Section operation has completed. If the section is not erased, it means blank check failed, both MGSTAT bits will be set.

**Table 21-63. Erase Verify EEPROM Section Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 010 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the requested section breaches the end of the EEPROM block |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read or if blank check failed. |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read or if blank check failed. |

### 21.4.7.15 Program EEPROM Command

The Program EEPROM operation programs one to four previously erased words in the EEPROM block. The Program EEPROM operation will confirm that the targeted location(s) were successfully programmed upon completion.

**CAUTION**

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

**Table 21-64. Program EEPROM Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x11 | Global address [23:16] to identify the EEPROM block |
| FCCOB1 | Global address [15:0] of word to be programmed | |
| FCCOB2 | Word 0 program value | |
| FCCOB3 | Word 1 program value, if desired | |
| FCCOB4 | Word 2 program value, if desired | |
| FCCOB5 | Word 3 program value, if desired | |

Upon clearing CCIF to launch the Program EEPROM command, the user-supplied words will be transferred to the Memory Controller and be programmed if the area is unprotected. The CCOBIX index value at Program EEPROM command launch determines how many words will be programmed in the EEPROM block. The CCIF flag is set when the operation has completed.

**Table 21-65. Program EEPROM Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] < 010 at command launch |
| | | Set if CCOBIX[2:0] > 101 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is supplied |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | | Set if the requested group of words breaches the end of the EEPROM block |
| | FPVIOL | Set if the selected area of the EEPROM memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 21.4.7.16 Erase EEPROM Sector Command

The Erase EEPROM Sector operation will erase all addresses in a sector of the EEPROM block.

**Table 21-66. Erase EEPROM Sector Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x12 | Global address [23:16] to identify EEPROM block |
| FCCOB1 | Global address [15:0] anywhere within the sector to be erased. See Section 21.1.2.2 for EEPROM sector size. | |

Upon clearing CCIF to launch the Erase EEPROM Sector command, the Memory Controller will erase the selected Flash sector and verify that it is erased. The CCIF flag will set after the Erase EEPROM Sector operation has completed.

**Table 21-67. Erase EEPROM Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 21-29) |
| | | Set if an invalid global address [23:0] is suppliedsee Table 21-3 |
| | | Set if a misaligned word address is supplied (global address [0] != 0) |
| | FPVIOL | Set if the selected area of the EEPROM memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 21.4.7.17 Protection Override Command

The Protection Override command allows the user to temporarily override the protection limits, either decreasing, increasing or disabling protection limits, on P-Flash and/or EEPROM, if the comparison key provided as a parameter loaded on FCCOB matches the value of the key previously programmed on the Flash Configuration Field (see Table 21-4). The value of the Protection Override Comparison Key must not be 16'hFFFF, that is considered invalid and if used as argument will cause the Protection Override feature to be disabled. Any valid key value that does not match the value programmed in the Flash Configuration Field will cause the Protection Override feature to be disabled. Current status of the Protection Override feature can be observed on FPSTAT FPOVRD bit (see Section 21.3.2.4, "Flash Protection Status Register (FPSTAT)).

**Table 21-68. Protection Override Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x13 | Protection Update Selection [1:0] See Table 21-69 |
| FCCOB1 | Comparison Key | |
| FCCOB2 | reserved | New FPROT value |
| FCCOB3 | reserved | New DFPROT value |

**Table 21-69. Protection Override selection description**

| Protection Update Selection code [1:0] | Protection register selection |
|---|---|
| bit 0 | Update P-Flash protection<br>0 - keep unchanged (do not update)<br>1 - update P-Flash protection with new FPROT value loaded on FCCOB |

**Table 21-69. Protection Override selection description**

| Protection Update Selection code [1:0] | Protection register selection |
|---|---|
| bit 1 | Update EEPROM protection<br>0 - keep unchanged (do not update)<br>1 - update EEPROM protection with new DFPROT value loaded on FCCOB |

If the comparison key successfully matches the key programmed in the Flash Configuration Field the Protection Override command will preserve the current values of registers FPROT and DFPROT stored in an internal area and will override these registers as selected by the Protection Update Selection field with the value(s) loaded on FCCOB parameters. The new values loaded into FPROT and/or DFPROT can reconfigure protection without any restriction (by increasing, decreasing or disabling protection limits). If the command executes successfully the FPSTAT FPOVRD bit will set.

If the comparison key does not match the key programmed in the Flash Configuration Field, or if the key loaded on FCCOB is 16'hFFFF, the value of registers FPROT and DFPROT will be restored to their original contents before executing the Protection Override command and the FPSTAT FPOVRD bit will be cleared. If the contents of the Protection Override Comparison Key in the Flash Configuration Field is left in the erased state (i.e. 16'hFFFF) the Protection Override feature is permanently disabled. If the command execution is flagged as an error (ACCERR being set for incorrect command launch) the values of FPROT and DFPROT will not be modified.

The Protection Override command can be called multiple times and every time it is launched it will preserve the current values of registers FPROT and DFPROT in a single-entry buffer to be restored later; when the Protection Override command is launched to restore FPROT and DFPROT these registers will assume the values they had before executing the Protection Override command on the last time. If contents of FPROT and/or DFPROT registers were modified by direct register writes while protection is overridden these modifications will be lost. Running Protection Override command to restore the contents of registers FPROT and DFPROT will not force them to the reset values.

**Table 21-70. Protection Override Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != (001, 010 or 011) at command launch. |
| | | Set if command not available in current mode (see Table 21-29). |
| | | Set if protection is supposed to be restored (if key does not match or is invalid) and Protection Override command was not run previously (bit FPSTAT FPOVRD is 0), so there are no previous valid values of FPROT and DFPROT to be re-loaded. |
| | | Set if Protection Update Selection[1:0] = 00 (in case of CCOBIX[2:0] = 010 or 011) |
| | | Set if Protection Update Selection[1:0] = 00, CCOBIX[2:0] = 001 and a valid comparison key is loaded as a command parameter. |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |

## 21.4.8 Interrupts

The Flash module can generate an interrupt when a Flash command operation has completed or when a Flash command operation has detected an ECC fault.

**Table 21-71. Flash Interrupt Sources**

| Interrupt Source | Interrupt Flag | Local Enable | Global (CCR) Mask |
|---|---|---|---|
| Flash Command Complete | CCIF (FSTAT register) | CCIE (FCNFG register) | I Bit |
| ECC Single Bit Fault on Flash Read | SFDIF (FERSTAT register) | SFDIE (FERCNFG register) | I Bit |

**NOTE**

Vector addresses and their relative interrupt priority are determined at the MCU level.

### 21.4.8.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the SFDIF flag in combination with the SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 21.3.2.5, “Flash Configuration Register (FCNFG)”, Section 21.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 21.3.2.7, “Flash Status Register (FSTAT)”, and Section 21.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 21-31.

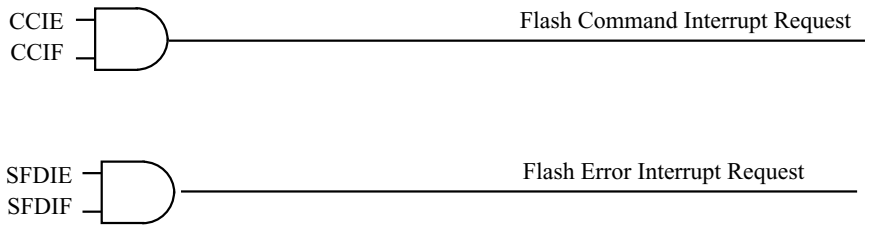



**Figure 21-31. Flash Module Interrupts Implementation**

## 21.4.9 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 21.4.8, “Interrupts”).

### 21.4.10 Stop Mode

If a Flash command is active (CCIF = 0) when the MCU requests stop mode, the current Flash operation will be completed before the MCU is allowed to enter stop mode.

## 21.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 21-11). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0xFF_FE0F. The security state out of reset can be permanently changed by programming the security byte assuming that the MCU is starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

### 21.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0xFF_FE00-0xFF_FE07). If the KEYEN[1:0] bits are in the enabled state (see Section 21.3.2.2), the Verify Backdoor Access Key command (see Section 21.4.7.11) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC register (see Table 21-11) will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, P-Flash memory and EEPROM memory will not be available for read access and will return invalid data.

The user code stored in the P-Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 21.3.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in Section 21.4.7.11
2. If the Verify Backdoor Access Key command is successful, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 10

The Verify Backdoor Access Key command is monitored by the Memory Controller and an illegal key will prohibit future use of the Verify Backdoor Access Key command. A reset of the MCU is the only method

to re-enable the Verify Backdoor Access Key command. The security as defined in the Flash security byte (0xFF_FE0F) is not changed by using the Verify Backdoor Access Key command sequence. The backdoor keys stored in addresses 0xFF_FE00-0xFF_FE07 are unaffected by the Verify Backdoor Access Key command sequence. The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the Flash protection register, FPROT.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the Flash security byte can be erased and the Flash security byte can be reprogrammed to the unsecure state, if desired. In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0xFF_FE00-0xFF_FE07 in the Flash configuration field.

### 21.5.2 Unsecuring the MCU in Special Single Chip Mode using BDM

A secured MCU can be unsecured in special single chip mode using an automated procedure described in Section 21.4.7.7.1, “Erase All Pin”, For a complete description about how to activate that procedure please refer to the Reference Manual. Alternatively, a similar (non-automated) procedure to unsecure the MCU in special single chip mode can be done by using the following method to erase the P-Flash and EEPROM memory:

1. Reset the MCU into special single chip mode
2. Delay while the BDM executes the Erase Verify All Blocks command write sequence to check if the P-Flash and EEPROM memories are erased
3. Send BDM commands to disable protection in the P-Flash and EEPROM memory
4. Execute the Erase All Blocks command write sequence to erase the P-Flash and EEPROM memory. Alternatively the Unsecure Flash command can be executed, if so the steps 5 and 6 below are skipped.
5. After the CCIF flag sets to indicate that the Erase All Blocks operation has completed, reset the MCU into special single chip mode
6. Delay while the BDM executes the Erase Verify All Blocks command write sequence to verify that the P-Flash and EEPROM memory are erased

If the P-Flash and EEPROM memory are verified as erased, the MCU will be unsecured. All BDM commands will now be enabled and the Flash security byte may be programmed to the unsecure state by continuing with the following steps:

7. Send BDM commands to execute the Program P-Flash command write sequence to program the Flash security byte to the unsecured state
8. Reset the MCU

### 21.5.3 Mode and Security Effects on Flash Command Availability

The availability of Flash module commands depends on the MCU operating mode and security state as shown in Table 21-29.

## 21.6 Initialization

On each system reset the flash module executes an initialization sequence which establishes initial values for the Flash Block Configuration Parameters, the FPROT and DFPROT protection registers, and the FOPT and FSEC registers. The initialization routine reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If a double bit fault is detected during the reset sequence, both MGSTAT bits in the FSTAT register will be set.

CCIF is cleared throughout the initialization sequence. The Flash module holds off all CPU access for a portion of the initialization sequence. Flash reads are allowed once the hold is removed. Completion of the initialization sequence is marked by setting CCIF high which enables user commands.

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

# Chapter 22
# Supply Voltage Sensor - (BATSV2)

**Table 22-1. Revision History Table**

| Rev. No. (Item No.) | Data | Sections Affected | Substantial Change(s) |
|---|---|---|---|
| V01.00 | 15 Dec 2010 | all | Initial Version |
| V02.00 | 16 Mar 2011 | 22.3.2.1<br>22.4.2.1 | - added BVLS[1] to support four voltage level<br>- moved BVHS to register bit 6 |

## 22.1 Introduction

The BATS module provides the functionality to measure the voltage of the battery supply pin VSENSE or of the chip supply pin VSUP.

### 22.1.1 Features

Either One of the voltage present on the VSENSE or VSUP pin can be routed via an internal divider to the internal Analog to Digital Converter. Independent of the routing to the Analog to Digital Converter, it is possible to route one of these voltages to a comparator to generate a low or a high voltage interrupt to alert the MCU.

### 22.1.2 Modes of Operation

The BATS module behaves as follows in the system power modes:

1. Run mode

   The activation of the VSENSE Level Sense Enable (BSESE=1) or ADC connection Enable (BSEAE=1) closes the path from the VSENSE pin through the resistor chain to ground and enables the associated features if selected.

   The activation of the VSUP Level Sense Enable (BSUSE=1) or ADC connection Enable (BSUAE=1) closes the path from VSUP pin through the resistor chain to ground and enables the associated features if selected.

   BSESE takes precedence over BSUSE. BSEAE takes precedence over BSUAE.

2. Stop mode

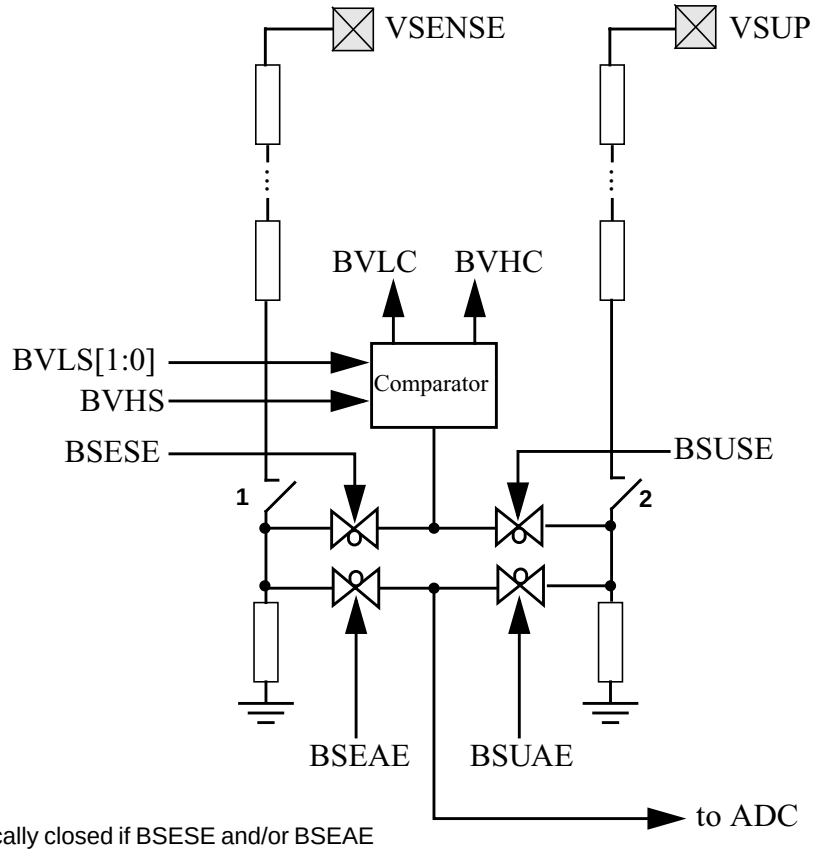
   During stop mode operation the path from the VSENSE pin through the resistor chain to ground is opened and the low voltage sense features are disabled.

During stop mode operation the path from the VSUP pin through the resistor chain to ground is opened and the low voltage sense features are disabled.
The content of the configuration register is unchanged.

### 22.1.3 Block Diagram

Figure 22-1 shows a block diagram of the BATS module. See device guide for connectivity to ADC channel.

**Figure 22-1. BATS Block Diagram**




1 automatically closed if BSESE and/or BSEAE is active, open during Stop mode
2 automatically closed if BSUSE and/or BSUAE is active, open during Stop mode

## 22.2 External Signal Description

This section lists the name and description of all external ports.

### 22.2.1 VSENSE — Supply (Battery) Voltage Sense Pin

This pin can be connected to the supply (Battery) line for voltage measurements. The voltage present at this input is scaled down by an internal voltage divider, and can be routed to the internal ADC or to a

comparator via an analog multiplexer. The pin itself is protected against reverse battery connections. To protect the pin from external fast transients an external resistor ($R_{VSENSE_R}$) is needed for protection.

### 22.2.2 VSUP — Voltage Supply Pin

This pin is the chip supply. It can be internally connected for voltage measurement. The voltage present at this input is scaled down by an internal voltage divider, and can be routed to the internal ADC or to a comparator via an analog multiplexer.

## 22.3 Memory Map and Register Definition

This section provides the detailed information of all registers for the BATS module.

### 22.3.1 Register Summary

Figure 22-2 shows the summary of all implemented registers inside the BATS module.

**NOTE**

Register Address = Module Base Address + Address Offset, where the Module Base Address is defined at the MCU level and the Address Offset is defined at the module level.

| Address Offset / Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 BATE | R | 0 | BVHS | BVLS[1:0] | | BSUAE | BSUSE | BSEAE | BSESE |
| | W | | | | | | | | |
| 0x0001 BATSR | R | 0 | 0 | 0 | 0 | 0 | 0 | BVHC | BVLC |
| | W | | | | | | | | |
| 0x0002 BATIE | R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIE | BVLIE |
| | W | | | | | | | | |
| 0x0003 BATIF | R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIF | BVLIF |
| | W | | | | | | | | |
| 0x0004 - 0x0005 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0006 - 0x0007 Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | W | | | | | | | | |

= Unimplemented

**Figure 22-2. BATS Register Summary**

## 22.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. Unused bits read back zero.

### 22.3.2.1 BATS Module Enable Register (BATE)

Module Base + 0x0000 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | BVHS | BVLS[1:0] | | BSUAE | BSUSE | BSEAE | BSESE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 22-3. BATS Module Enable Register (BATE)**

[1] Read: Anytime
Write: Anytime

**Table 22-2. BATE Field Description**

| Field | Description |
|---|---|
| 6<br>BVHS | **BATS Voltage High Select** — This bit selects the trigger level for the Voltage Level High Condition (BVHC).<br><br>0 Voltage level $V_{HBI1}$ is selected<br>1 Voltage level $V_{HBI2}$ is selected |
| 5:4<br>BVLS[1:0] | **BATS Voltage Low Select** — This bit selects the trigger level for the Voltage Level Low Condition (BVLC).<br><br>00 Voltage level $V_{LBI1}$ is selected<br>01 Voltage level $V_{LBI2}$ is selected<br>10 Voltage level $V_{LBI3}$ is selected<br>11 Voltage level $V_{LBI4}$ is selected |
| 3<br>BSUAE | **BATS VSUP ADC Connection Enable** — This bit connects the VSUP pin through the resistor chain to ground and connects the ADC channel to the divided down voltage. This bit can be set only if the BSEAE bit is cleared.<br><br>0 ADC Channel is disconnected<br>1 ADC Channel is connected |
| 2<br>BSUSE | **BATS VSUP Level Sense Enable** — This bit connects the VSUP pin through the resistor chain to ground and enables the Voltage Level Sense features measuring BVLC and BVHC. This bit can be set only if the BSESE bit is cleared.<br><br>0 Level Sense features disabled<br>1 Level Sense features enabled |
| 1<br>BSEAE | **BATS VSENSE ADC Connection Enable** — This bit connects the VSENSE pin through the resistor chain to ground and connects the ADC channel to divided down voltage. Setting this bit will clear bit BSUAE .<br><br>0 ADC Channel is disconnected<br>1 ADC Channel is connected |
| 0<br>BSESE | **BATS VSENSE Level Sense Enable** — This bit connects the VSENSE pin through the resistor chain to ground and enables the Voltage Level Sense features measuring BVLC and BVHC.Setting this bit will clear bit BSUSE<br><br>0 Level Sense features disabled<br>1 Level Sense features enabled |

## NOTE

When opening the resistors path to ground by changing BSESE, BSEAE or BSUSE, BSUAE then for a time $T_{EN_UNC}$ + two bus cycles the measured value is invalid. This is to let internal nodes be charged to correct value. BVHIE, BVLIE might be cleared for this time period to avoid false interrupts.

### 22.3.2.2 BATS Module Status Register (BATSR)

Module Base + 0x0001 Access: User read only[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | BVHC | BVLC |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

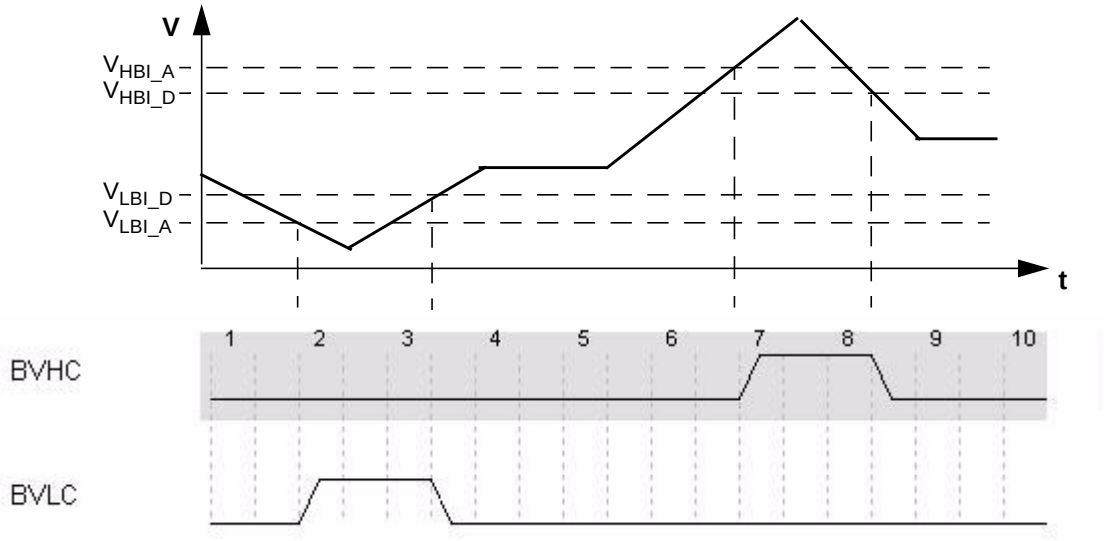
**Figure 22-4. BATS Module Status Register (BATSR)**

[1] Read: Anytime
Write: Never

**Table 22-3. BATSR - Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>BVHC | **BATS Voltage Sense High Condition Bit** — This status bit indicates that a high voltage at VSENSE or VSUP, depending on selection, is present.<br><br>0 $V_{measured} < V_{HBI_A}$ (rising edge) or $V_{measured} < V_{HBI_D}$ (falling edge)<br>1 $V_{measured} \geq V_{HBI_A}$ (rising edge) or $V_{measured} \geq V_{HBI_D}$ (falling edge) |
| 0<br>BVLC | **BATS Voltage Sense Low Condition Bit** — This status bit indicates that a low voltage at VSENSE or VSUP, depending on selection, is present.<br><br>0 $V_{measured} \geq V_{LBI_A}$ (falling edge) or $V_{measured} \geq V_{LBI_D}$ (rising edge)<br>1 $V_{measured} < V_{LBI_A}$ (falling edge) or $V_{measured} < V_{LBI_D}$ (rising edge) |

**Figure 22-5. BATS Voltage Sensing**



### 22.3.2.3 BATS Interrupt Enable Register (BATIE)

Module Base + 0x0002 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIE | BVLIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 22-6. BATS Interrupt Enable Register (BATIE)**

[1] Read: Anytime
Write: Anytime

**Table 22-4. BATIE Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>BVHIE | **BATS Interrupt Enable High** — Enables High Voltage Interrupt **.**<br><br>0 No interrupt will be requested whenever BVHIF flag is set .<br>1 Interrupt will be requested whenever BVHIF flag is set |
| 0<br>BVLIE | **BATS Interrupt Enable Low** — Enables Low Voltage Interrupt .<br><br>0 No interrupt will be requested whenever BVLIF flag is set .<br>1 Interrupt will be requested whenever BVLIF flag is set . |

### 22.3.2.4 BATS Interrupt Flag Register (BATIF)

Module Base + 0x0003 Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIF | BVLIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 22-7. BATS Interrupt Flag Register (BATIF)**

[1] Read: Anytime
Write: Anytime, write 1 to clear

**Table 22-5. BATIF Register Field Descriptions**

| Field | Description |
|---|---|
| 1<br>BVHIF | **BATS Interrupt Flag High Detect** — The flag is set to 1 when BVHC status bit changes.<br><br>0 No change of the BVHC status bit since the last clearing of the flag.<br>1 BVHC status bit has changed since the last clearing of the flag. |
| 0<br>BVLIF | **BATS Interrupt Flag Low Detect** — The flag is set to 1 when BVLC status bit changes.<br><br>0 No change of the BVLC status bit since the last clearing of the flag.<br>1 BVLC status bit has changed since the last clearing of the flag. |

### 22.3.2.5 Reserved Register

Module Base + 0x0006
Module Base + 0x0007

Access: User read/write[1]

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Reset | x | x | x | x | x | x | x | x |

**Figure 22-8. Reserved Register**

[1] Read: Anytime
Write: Only in special mode

**NOTE**

These reserved registers are designed for factory test purposes only and are not intended for general user access. Writing to these registers when in special mode can alter the module's functionality.

## 22.4 Functional Description

### 22.4.1 General

The BATS module allows measuring voltages on the VSENSE and VSUP pins. The VSENSE pin is implemented to allow measurement of the supply Line (Battery) Voltage $V_{BAT}$ directly. By bypassing the device supply capacitor and the external reversed battery protection diode this pin allows to detect under/over voltage conditions without delay. A series resistor ($R_{VSENSE_R}$) is required to protect the VSENSE pin from fast transients.

The voltage at the VSENSE or VSUP pin can be routed via an internal voltage divider to an internal Analog to Digital Converter Channel. Also the BATS module can be configured to generate a low and high voltage interrupt based on VSENSE or VSUP. The trigger level of the high and low interrupt are selectable.

In a typical application, the module could be used as follows: The voltage at VSENSE is observed via usage of the interrupt feature (BSESE=1, BVHIE=1), while the VSUP pin voltage is routed to the ADC to allow regular measurement (BSUAE=1).

## 22.4.2 Interrupts

This section describes the interrupt generated by the BATS module. The interrupt is only available in CPU run mode. Entering and exiting CPU stop mode has no effect on the interrupt flags.

To make sure the interrupt generation works properly the bus clock frequency must be higher than the Voltage Warning Low Pass Filter frequency ($f_{VWLP_filter}$).

The comparator outputs BVLC and BVHC are forced to zero if the comparator is disabled (configuration bits BSESE and BSUSE are cleared). If the software disables the comparator during a high or low Voltage condition (BVHC or BVLC active), then an additional interrupt is generated. To avoid this behavior the software must disable the interrupt generation before disabling the comparator.

The BATS interrupt vector is named in Table 22-6. Vector addresses and interrupt priorities are defined at MCU level.

The module internal interrupt sources are combined into one module interrupt signal.

**Table 22-6. BATS Interrupt Sources**

| Module Interrupt Source | Module Internal Interrupt Source | Local Enable |
|---|---|---|
| BATS Interrupt (BATI) | BATS Voltage Low Condition Interrupt (BVLI) | BVLIE = 1 |
| | BATS Voltage High Condition Interrupt (BVHI) | BVHIE = 1 |

### 22.4.2.1 BATS Voltage Low Condition Interrupt (BVLI)

To use the Voltage Low Interrupt the Level Sensing must be enabled (BSESE =1 or BSUSE =1).

If measured when

a) $V_{LBI1}$ selected with BVLS[1:0] = 0x0
at selected pin $V_{measure} < V_{LBI1_A}$ (falling edge) or $V_{measure} < V_{LBI1_D}$ (rising edge)

or when

b) $V_{LBI2}$ selected with BVLS[1:0] = 0x1
at selected pin $V_{measure} < V_{LBI2_A}$ (falling edge) or $V_{measure} < V_{LBI2_D}$ (rising edge)

or when

c) $V_{LBI3}$ selected with BVLS[1:0] = 0x2
at selected pin $V_{measure} < V_{LBI3_A}$ (falling edge) or $V_{measure} < V_{LBI3_D}$ (rising edge)

or when

d) $V_{LBI4}$ selected with BVLS[1:0] = 0x3
at selected pin $V_{measure} < V_{LBI4_A}$ (falling edge) or $V_{measure} < V_{LBI4_D}$ (rising edge)

then BVLC is set. BVLC status bit indicates that a low voltage at the selected pin is present. The Low Voltage Interrupt flag (BVLIF) is set to 1 when the Voltage Low Condition (BVLC) changes state . The

Interrupt flag BVLIF can only be cleared by writing a 1. If the interrupt is enabled by bit BVLIE the module requests an interrupt to MCU (BATI).

#### 22.4.2.2 BATS Voltage High Condition Interrupt (BVHI)

To use the Voltage High Interrupt the Level Sensing must be enabled (BSESE =1 or BSUSE).

If measured when

a) $V_{HBI1}$ selected with BVHS = 0
   at selected pin $V_{measure} \geq V_{HBI1_A}$ (rising edge) or $V_{measure} \geq V_{HBI1_D}$ (falling edge)

or when

a) $V_{HBI2}$ selected with BVHS = 1
   at selected pin $V_{measure} \geq V_{HBI2_A}$ (rising edge) or $V_{measure} \geq V_{HBI2_D}$ (falling edge)

then BVHC is set. BVHC status bit indicates that a high voltage at the selected pin is present. The High Voltage Interrupt flag (BVHIF) is set to 1 when a Voltage High Condition (BVHC) changes state. The Interrupt flag BVHIF can only be cleared by writing a 1. If the interrupt is enabled by bit BVHIE the module requests an interrupt to MCU (BATI).

# Appendix A MCU Electrical Specifications

## A.1 General

This supplement contains the most accurate electrical information for the MC9S12ZVHY-Family available at the time of publication.

This introduction is intended to give an overview on several common topics like power supply, current injection etc.

### A.1.1 Parameter Classification

The electrical parameters shown in this supplement are guaranteed by various methods. To give the customer a better understanding the following classification is used and the parameters are tagged accordingly in the tables where appropriate.

**NOTE**

This classification is shown in the column labeled “C” in the parameter tables where appropriate.

- P:Those parameters are guaranteed during production testing on each individual device.
- C: Those parameters are achieved in design characterization by measuring a statistically relevant sample size across process variations.
- T:Those parameters are achieved by design characterization on a small sample size from typical devices under typical conditions unless otherwise noted. All values shown in the typical column are within this category.
- D:Those parameters are derived mainly from simulations.

**Table A-1. Power Supplies**

| Mnemonic | Nominal Voltage | Description |
|---|---|---|
| VSS1 | 0V | Ground pin for 2.8V core supply voltage generated by on chip voltage regulator |
| VSS2 | 0V | Ground pin for 1.8V core supply voltage generated by on chip voltage regulator |
| VSS3 | 0V | Ground pin for padring |
| VDDX1[(1)] | 5.0 V | 5V power supply output for I/O drivers generated by on chip voltage regulator |
| VSSX1,2,3 | 0V | Ground pin for I/O drivers |
| VDDX2,3 | 5.0 V | 5V power supply for I/O drivers which should be supply from external power transistor |
| VDDA | 5.0 V | 5V Power supply for the analog-to-digital converter and for the reference circuit of the internal voltage regulator |
| VSSA | 0V | Ground pin for VDDA analog supply |
| VDDM1 | 5.0 V | 5V power supply for Port U I/O drivers which should be supply from external power transistor |
| VSSM1 | 0V | Ground pin for Port U I/O drivers |
| VSUP | 12V/18V | External power supply for voltage regulator |

1. VDDX2 pin is internally connected to VDDX1 by metal

VDDA and VDDM1 are connected to VDDX pins by diodes for ESD protection such that VDDX must not exceed VDDA and VDDM1,2 by more than a diode voltage drop. VSSA, VSSM1 and VSSX are connected by anti-parallel diodes for ESD protection.

## A.1.2 Pins

There are 4 groups of functional pins.

### A.1.2.1 General Purpose I/O Pins (GPIO)

The I/O pins have a level in the range of 4.5V to 5.5V. This class of pins is comprised of all port I/O pins (except PE0,1), BKGD and the $\overline{\text{RESET}}$ pins. Some functionality may be disabled.

### A.1.2.2 High Voltage Pins

These consist of the VSENSE, VSUP and BCTL pins. These pins are intended to interface to external components operating in the automotive battery range. They have nominal voltages above the standard 5V I/O voltage range.

### A.1.2.3 Main Oscillator

If the designated PE0 (EXTAL) and PE1 (XTAL) pins are configured for external oscillator operation then these pins have a nominal voltage of 1.8V.

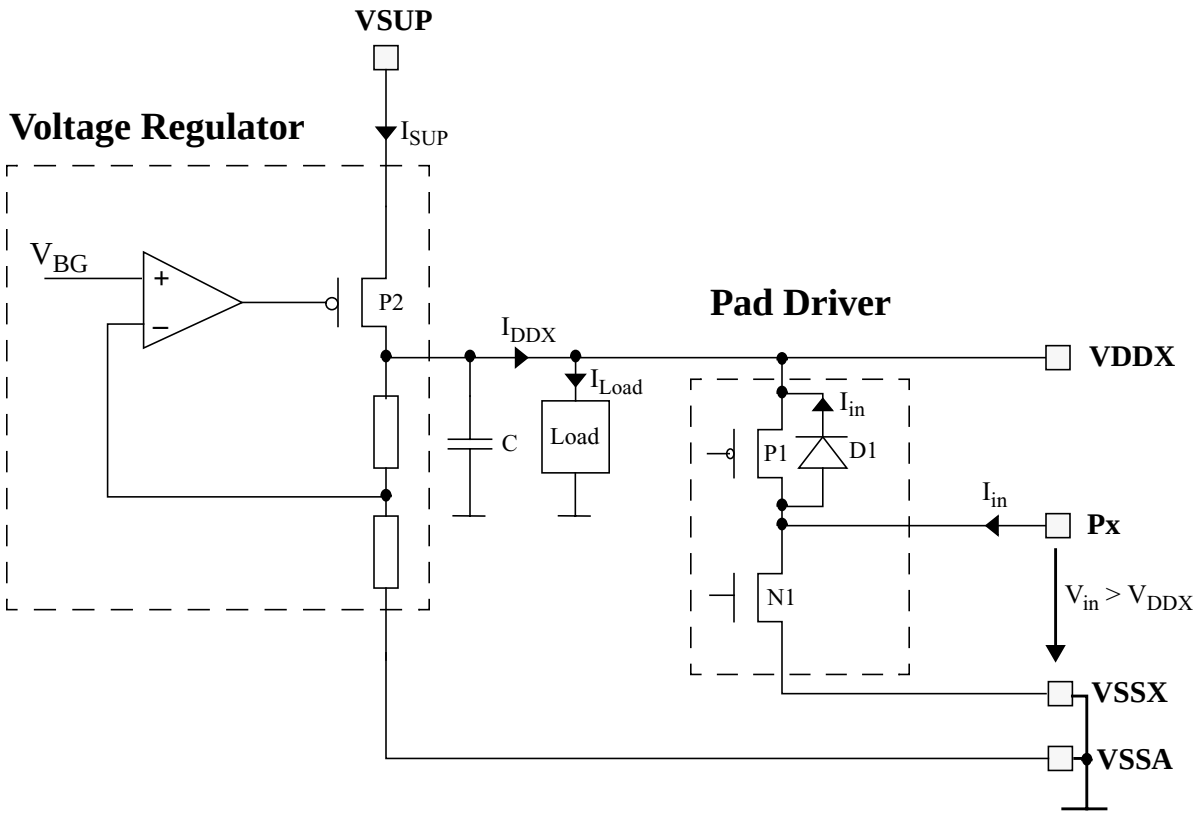
### A.1.2.4 TEST

This pin is used for production testing only. The TEST pin must be tied to ground in all applications.

## A.1.3 Current Injection

Power supply must maintain regulation within operating VDDX or $V_{DD}$ range during instantaneous and operating maximum current conditions. **Figure A-1.** shows a 5 V GPIO pad driver and the on chip voltage regulator with VDDX output. It shows also the power and ground pins VSUP, VDDX, VSSX and VSSA. Px represents any 5 V GPIO pin. Assume Px is configured as an input. The pad driver transistors P1 and N1 are switched off (high impedance). If the voltage $V_{in}$ on Px is greater than $V_{DDX}$ a positive injection current $I_{in}$ will flow through diode D1 into VDDX node. If this injection current $I_{in}$ is greater than $I_{Load}$, the internal power supply VDDX may go out of regulation. Ensure the external $V_{DDX}$ load will shunt current greater than maximum injection current. This is the greatest risk when the MCU is not consuming power; e.g., if no system clock is present, or if the clock rate is very low which would reduce overall power consumption.

**Figure A-1. Current Injection on GPIO Port if $V_{in} > V_{DDX}$**



## A.1.4 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only. A functional operation outside these ranges is not guaranteed. Stress beyond these limits may affect the reliability or cause permanent damage of the device.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level.

**Table A-2. Absolute Maximum Ratings(1)**

| Num | Rating | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Voltage regulator supply voltage | $V_{SUP}$ | -0.3V | 42 | V |
| 2 | Voltage Regulator basis control | $V_{BCTL}$ | -0.3V | 42 | V |
| 3 | Voltage difference $V_{DDX}$ to $V_{DDM}$ and $V_{DDA}$(2) | $\Delta_{VDDXM}$ | –0.1 | 0.1 | V |
| 4 | Voltage difference $V_{SSX}$ to $V_{SSM}$ and $V_{SSA}$ | $\Delta_{VSSXM}$ | –0.3 | 0.3 | V |
| 5 | Digital I/O input voltage | $V_{IN}$ | –0.3 | 6.0 | V |
| 6 | EXTAL, XTAL (3) | $V_{ILV}$ | –0.3 | 2.16 | V |
| 7 | TEST input | $V_{TEST}$ | –0.3 | 10.0 | V |
| 8 | Instantaneous maximum current<br>Single pin limit for all digital I/O pins(4) except PU | $I_D$ | –25 | +25 | mA |
| 9 | Instantaneous maximum current<br>Single pin limit for all PU I/O pins | $I_D$ | -55 | 55 | mA |
| 10 | Instantaneous maximum current<br>Single pin limit for all LCD pins when PAD is use as LCD function | $I_D$ | -2.5 | 2.5 | mA |
| 11 | Instantaneous maximum current on VDDM | $I_{VDDM}$ | -220 | 220 | mA |
| 12 | Instantaneous maximum current<br>Single pin limit for EXTAL, XTAL | $I_{DL}$ | –25 | +25 | mA |
| 13 | Storage temperature range | $T_{stg}$ | –65 | 155 | °C |

1. Beyond absolute maximum ratings device might be damaged.
2. VDDX and VDDA must be shorted
3. EXTAL, XTAL pins configured for external oscillator operation only
4. All digital I/O pins are internally clamped to $V_{SSX}$ and $V_{DDX}$, or $V_{SSA}$ and $V_{DDA}$.

## A.1.5 ESD Protection and Latch-up Immunity

All ESD testing is in conformity with CDF-AEC-Q100 stress test qualification for automotive grade integrated circuits. During the device qualification ESD stresses were performed for the Human Body Model (HBM) and the Charged-Device Model.

A device will be defined as a failure if after exposure to ESD pulses the device no longer meets the device specification. Complete DC parametric and functional testing is performed per the applicable device specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

**Table A-3. ESD and Latch-up Test Conditions**

| Model | Spec | Description | Symbol | Value | Unit |
|---|---|---|---|---|---|
| Human Body | JESD22-A114 | Series Resistance | R | 1500 | Ω |
| | | Storage Capacitance | C | 100 | pF |
| | | Number of Pulse per pin<br>positive<br>negative | - | -<br>3<br>3 | |
| Charged- Device | JESD22-C101 | Series Resistance | R | 0 | Ω |
| | | Storage Capacitance | C | 4 | pF |
| Latch-up for 5V GPIO's | | Minimum Input Voltage Limit | | -2.5 | V |
| | | Maximum Input Voltage Limit | | +7.5 | V |
| Latch-up for BCTL/VSENSE | | Minimum Input Voltage Limit | | -7 | V |
| | | Maximum Input Voltage Limit | | +21 | V |

**Table A-4. ESD Protection and Latch-up Characteristics**

| Num | C | Rating | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | C | Human Body Model (HBM):<br>-VSENSE vs VSS-all pins | $V_{HBM}$ | +/-4<br>+/-2 | - | kV |
| 2 | C | Charged-Device Model (CDM):<br>Corner Pins | $V_{CDM}$ | +/-750 | - | V |
| 3 | C | Charged-Device Model (CDM):<br>all other pins | $V_{CDM}$ | +/-500 | - | V |
| 4 | C | Latch-up Current of 5V GPIO's at T= 27°C<br>positive<br>negative | $I_{LAT}$ | +100<br>-100 | - | mA |
| 5 | C | Latch-up Current of VSENSET and BCTL at T= 27°C<br>positive<br>negative | $I_{LAT}$ | +100<br>-100 | - | mA |

## A.1.6 Operating Conditions

This section describes the operating conditions of the device. Unless otherwise noted those conditions apply to all the following data.

**NOTE**

Please refer to the temperature rating of the device with regards to the ambient temperature $T_A$ and the junction temperature $T_J$. For power dissipation calculations refer to Section A.1.7, “Power Dissipation and Thermal Characteristics”.

**Table A-5. Operating Conditions**

| Num | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | Voltage regulator supply | $V_{SUP}$ | 5.5 | 12 | 40(1) | V |
| 3 | main Oscillator | $f_{osc}$ | 4 | — | 20 | MHz |
| 4 | 32K Oscillator | $f_{osc32k}$ | 32 | — | 40 | KHz |
| 4 | Bus frequency | $f_{bus}$ | see Footnote(2) | — | 32 | MHz |
| 5 | Operating junction temperature range<br>Operating ambient temperature range(3) | $T_J$<br>$T_A$ | –40<br>–40 | —<br>— | 150<br>105 | °C |

1. Normal operating range is 6V - 18V. Continuos operation at 40V is not allowed. Only Transient Conditions (Load Dump) single pulse $t_{max}$<400ms
2. Minimum bus frequency for ADC module refer to fADCCLK and for Flash program and erase please refer to $f_{NVMOP}$
3. Please refer to Section A.1.7, “Power Dissipation and Thermal Characteristics” for more details about the relation between ambient temperature $T_A$ and device junction temperature $T_J$.

**NOTE**

Operation is guaranteed when powering down until low voltage reset assertion.

## A.1.7 Power Dissipation and Thermal Characteristics

Power dissipation and thermal characteristics are closely related. The user must assure that the maximum operating junction temperature is not exceeded. The average chip-junction temperature ($T_J$) in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \Theta_{JA})$$

$T_J$ = Junction Temperature, [°C]

$T_A$ = Ambient Temperature, [°C]

$P_D$ = Total Chip Power Dissipation, [W]

$\Theta_{JA}$ = Package Thermal Resistance, [°C/W]

The total power dissipation $P_D$ can be calculated from the equation below. Table A-6 below lists the power dissipation components . Table A-6 gives an overview of the supply currents.

$$P_D = P_{VSUP} + P_{INT} + P_{BCTL} + P_{INT} - P_{GPIO}$$

**Table A-6. Power Dissipation Components**

| Power Component | Description |
|---|---|
| $P_{SUP} = V_{SUP}\ I_{SUP}$ | Internal Power through VSUP pin |
| $P_{BCTL} = V_{BCTL}\ I_{BCTL}$ | Internal Power through BCTL pin |
| $P_{INT} = V_{DDX}\ I_{VDDX} + V_{DDA}\ I_{VDDA} + V_{DDM}\ I_{VDDM}$ | Internal Power through VDDX/A/M pins. |
| $P_{GPIO} = V_{I/O}\ I_{I/O}$ | Power dissipation of external load driven by GPIO Port. Assuming the load is connected between GPIO and ground. This power component is included in $P_{INT}$ and is subtracted from overall MCU power dissipation $P_D$ |

**Figure A-2. Supply Currents Overview**

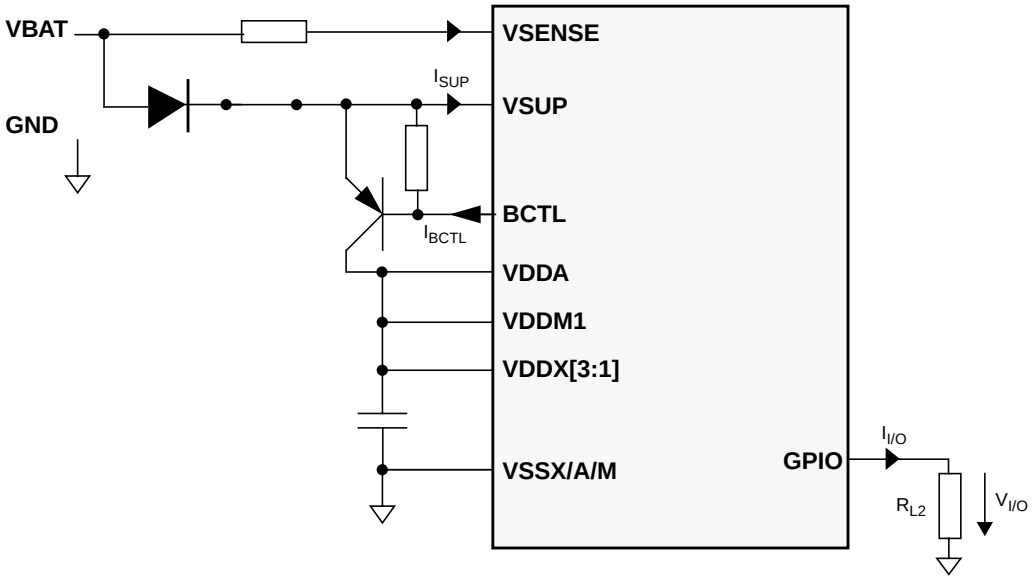


**Table A-7. Thermal Package Characteristics[1]**

| Num | C | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| LQFP 100 | | | | | | | |
| 1 | D | Thermal resistance 100LQFP, single sided PCB(1) | $\theta_{JA}$ | — | **58** | — | °C/W |
| 2 | D | Thermal resistance 100LQFP, double sided PCB with 2 internal planes(2) | $\theta_{JA}$ | — | **45** | — | °C/W |
| 3 | D | Junction to Board 100LQFP | $\theta_{JB}$ | — | **30** | — | °C/W |
| 4 | D | Junction to Case 100LQFP4 | $\theta_{JC}$ | — | **13** | — | °C/W |
| 144LQFP | | | | | | | |
| 5 | D | Thermal resistance 144LQFP, single sided PCB(3) | $\theta_{JA}$ | — | **50** | — | °C/W |
| 6 | D | Thermal resistance 144LQFP, double sided PCB with 2 internal planes(4) | $\theta_{JA}$ | — | **41** | — | °C/W |
| 7 | D | Junction to Board 144LQFP | $\theta_{JB}$ | — | **29** | — | °C/W |
| 8 | D | Junction to Case 144LQFP4 | $\theta_{JC}$ | — | **11** | — | °C/W |

1. Junction to ambient thermal resistance, $\theta_{JA}$ was simulated to be equivalent to the JEDEC specification JESD51-2 in a horizontal configuration in natural convection.
2. Junction to ambient thermal resistance, $\theta_{JA}$ was simulated to be equivalent to the JEDEC specification JESD51-7 in a horizontal configuration in natural convection.
3. Junction to ambient thermal resistance, $\theta_{JA}$ was simulated to be equivalent to the JEDEC specification JESD51-2 in a horizontal configuration in natural convection.
4. Junction to ambient thermal resistance, $\theta_{JA}$ was simulated to be equivalent to the JEDEC specification JESD51-7 in a horizontal configuration in natural convection.

1. The values for thermal resistance are achieved by package simulations

## A.1.8 I/O Characteristics

This section describes the characteristics of I/O pins

**Table A-8. 5-V I/O Characteristics**

| Conditions are 4.5 V < $V_{DDX}$< 5.5 V junction temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all GPIO pins (defined in A.1.2.1/A-754). | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Input high voltage | $V_{IH}$ | 0.65*$V_{DDX}$ | — | — | V |
| 2 | T | Input high voltage | $V_{IH}$ | — | — | $V_{DDX}$+0.3 | V |
| 3 | P | Input low voltage | $V_{IL}$ | — | — | 0.35*$V_{DDX}$ | V |
| 4 | T | Input low voltage | $V_{IL}$ | $V_{SSX}$–0.3 | — | — | V |
| 5 | C | Input hysteresis | $V_{HYS}$ | | 250 | — | mV |
| 6 | P | Input leakage current (pins in high impedance input mode)(1) all except PU<br>$V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>**V** Temperature range -40°C to 105°C | $I_{in}$ | -1 | — | 1 | µA |
| | P | Input leakage current (pins in high impedance input mode)(2) . PU<br>$V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>**V** Temperature range -40°C to 105°C | $I_{in}$ | -2.5 | — | 2.5 | µA |
| 6a | C | Input leakage current (pins in high impedance input mode) all except PU<br>$V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>25°C<br>150°C | $I_{in}$ | — | <br><br><br>±1<br>±26 | — | nA |
| | C | Input leakage current (pins in high impedance input mode) PU<br>$V_{in}$ = $V_{DD35}$ or $V_{SS35}$<br>25°C<br>150°C | $I_{in}$ | — | <br><br><br>±10<br>±1500 | — | nA |
| 7 | P | Output high voltage (pins in output mode) all except PU<br>Full drive $I_{OH}$ = -4mA | $V_{OH}$ | $V_{DD35}$ – 0.8 | — | — | V |
| | P | Output high voltage (pins in output mode) PU<br>Full drive $I_{OH}$ = –20 mA | $V_{OH}$ | $V_{DDM}$ – 0.4 | — | — | V |
| 8 | P | Output low voltage (pins in output mode) all except PU<br>Full drive $I_{OL}$ = +4 mA | $V_{OL}$ | — | — | 0.8 | V |
| | P | Output low voltage (pins in output mode) PU<br>Full drive $I_{OL}$ = +20 mA | $V_{OL}$ | — | — | 0.4 | V |
| 9 | C | Port U Output Rise Time<br>$V_{DD5}$=5V, 10% to 90% of $V_{OH}$<br>Cload 47pF connected to GND, slew disabled<br>Rload=1KΩ connected to GND, slew enabled<br>Rload=1KΩ connected to VDD, slew enabled | $t_r$ | <br><br>— | <br><br>9.9<br>87<br>107 | <br><br>— | ns |

## Table A-8. 5-V I/O Characteristics

| Conditions are 4.5 V < $V_{DDX}$< 5.5 V junction temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all GPIO pins (defined in A.1.2.1/A-754). | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | C | Port U Output Fall Time<br>$V_{DD5}$=5V, 10% to 90% of $V_{OH}$<br>Cload 47pF connected to GND, slew disabled<br>Rload=1KΩ connected to GND, slew enabled<br>Rload=1KΩ connected to VDD,, slew enabled | $t_f$ | —<br><br><br><br> | 9.9<br>105<br>87 | —<br><br><br><br> | ns |
| 11 | P | Internal pull up current (All GPIO except RESET and PU)<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $I_{PUL}$ | -10 | — | -130 | µA |
| | P | Internal pull up current (RESET pin)<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $R_{PUL}$ | 3.8 | 5 | 10.5 | KΩ |
| | P | Internal pull up device current PU<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $I_{PUL}$ | -10 | — | -130 | µA |
| 12 | P | Internal pull down current (All GPIO except PU)<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $I_{PDH}$ | 10 | — | 130 | µA |
| | P | Internal pull down device current PU<br>$V_{IH}$ min > input voltage > $V_{IL}$ max | $I_{PDH}$ | 10 | — | 130 | µA |
| 13 | D | Input capacitance | $C_{in}$ | — | 7 | — | pF |
| 14 | T | Injection current(3)<br>Single pin limit<br>Total device Limit, sum of all injected currents | <br>$I_{ICS}$<br>$I_{ICP}$ | <br>–2.5<br>–25 | — | <br>2.5<br>25 | mA |
| 12 | D | Input capacitance | $C_{in}$ | — | 7 | — | pF |
| 13 | T | Injection current(4)<br>Single pin limit<br>Total device Limit, sum of all injected currents | <br>$I_{ICS}$<br>$I_{ICP}$ | <br>–2.5<br>–25 | — | <br>2.5<br>25 | µA |

1. Maximum leakage current occurs at maximum operating temperature.
2. Maximum leakage current occurs at maximum operating temperature.
3. Refer to Section A.1.3, “Current Injection” for more details
4. Refer to Section A.1.3, “Current Injection” for more details

## Table A-9. 5-V I/O Timing Characteristics

| Conditions are 4.5 V < $V_{DDX}$< 5.5 V junction temperature from –40°C to +150°C, unless otherwise noted<br>I/O Characteristics for all GPIO pins (defined in A.1.2.1/A-754). | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Port T, S, AD interrupt input pulse filtered (STOP)(1) | $t_{P_MASK}$ | — | — | 3 | µs |
| 2 | P | Port T, S, AD interrupt input pulse passed (STOP)1 | $t_{P_PASS}$ | 10 | — | — | µs |
| 3 | D | Port T, S, AD interrupt input pulse filtered ($\overline{\text{STOP}}$) in number of bus clock cycles of period 1/$f_{bus}$ | $n_{P_MASK}$ | — | — | 3 | |
| 4 | D | Port T, S, AD interrupt input pulse passed ($\overline{\text{STOP}}$) in number of bus clock cycles of period 1/$f_{bus}$ | $n_{P_PASS}$ | 4 | — | — | |
| 5 | D | $\overline{\text{IRQ}}$ pulse width, edge-sensitive mode ($\overline{\text{STOP}}$) in number of bus clock cycles of period 1/$f_{bus}$ | $n_{IRQ}$ | 1 | — | — | |

1. Parameter only applies in stop or pseudo stop mode.

## A.1.9 Supply Currents

This section describes the current consumption characteristics of the device as well as the conditions for the measurements.

### A.1.9.1 Measurement Conditions

Current is measured on VSUP. VDDX is connected to VDDA. It does not include the current to drive external loads. Unless otherwise noted the currents are measured in special single chip mode and the CPU code is executed from RAM. For Run and Wait current measurements PLL is on and the reference clock is the IRC1M trimmed to 1MHz. The bus frequency is 32MHz and the CPU frequency is 64MHz. Table A-10, Table A-11 and Table A-12 show the configuration of the CPMU module and the peripherals for Run, Wait and Stop current measurement.

**Table A-10. CPMU Configuration for Pseudo Stop Current Measurement**

| CPMU REGISTER | Bit settings/Conditions |
|---|---|
| CPMUCLKS | PLLSEL=0, PSTP=1, CSAD=0, PRE=PCE=RTIOSCSEL=COPOSCSEL=1 |
| CPMUOSC | OSCE=1, External Square wave on EXTAL $f_{EXTAL}$=4MHz, $V_{IH}$= 1.8V, $V_{IL}$=0V |
| CPMURTI | RTDEC=0, RTR[6:4]=111, RTR[3:0]=1111; |
| CPMUCOP | WCOP=1, CR[2:0]=111 |

**Table A-11. CPMU Configuration for Run/Wait and Full Stop Current Measurement**

| CPMU REGISTER | Bit settings/Conditions |
|---|---|
| CPMUSYNR | VCOFRQ[1:0]=01,SYNDIV[5:0] = 31 |
| CPMUPOSTDIV | POSTDIV[4:0]=0 |
| CPMUCLKS | PLLSEL=1, CSAD=0 |
| CPMUOSC | OSCE=0, Reference clock for PLL is $f_{ref}$=$f_{irc1m}$ trimmed to 1MHz |
| API settings for STOP current measurement | |
| CPMUAPICTL | APIEA=0, APIFE=1, APIE=0 |
| CPMUAPITR | trimmed to >=10Khz |
| CPMUAPIRH/RL | set to 0xFFFF |

**Table A-12. Peripheral Configurations for Run & Wait Current Measurement**

| Peripheral | Configuration |
|---|---|
| SCI | Continuously transmit data (0x55) at speed of 19200 baud |
| SPI | Configured to master mode, continuously transmit data (0x55) at 1Mbit/s |
| MSCAN0 | Configured to loop-back mode using a bit rate of 500Kbits/s |
| PWM | Configured to toggle its pins at the rate of 40kHz |
| ADC | The peripheral is configured to operate at its maximum specified frequency and to continuously convert voltages on all input channels in sequence. |
| IIC | Operate in master mode and continuously transimit data 0x55 at 100Kbits/s |
| DBG | The module is enabled and the comparators are configured to trigger in outside range.The range covers all the code executed by the core. |
| TIM | The peripheral is configured to output compare mode, pulse accumulator and modulus counter enabled. |
| LCD | Configured to 244Hz frame frequency, 1/4 Duty, 1/3 Bias with all FP/BP enabled and all segment on |
| MC | configured to full H-bridge mode MCPER=0x3FF, 1/2fbus motor controller timer counter clock, MCDC=0x20 |
| SSD | disabled |
| SSG | Configured to 1K Hz Tone with attached mode, mix mode output |
| RTC | Enabled from 32K osc |
| COP & RTI | Enabled |
| BATS | Enabled |

**Table A-13. Run and Wait Current Characteristics**

| Conditions are: $V_{SUP}$=12V, $T_A$=125°C, see Table A-11 and Table A-12 | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Run Current | $I_{SUPR}$ | | 23 | 30 | mA |
| 2 | P | Wait Current | $I_{SUPW}$ | | 15 | 20 | mA |

**Table A-14. Stop Current Characteristics**

| Conditions are: $V_{SUP}$=12V | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| Stop Current all modules off | | | | | | | |
| 1 | P | $T_A$= -40°C | $I_{SUPS}$ | | 21 | 45 | µA |

**Table A-14. Stop Current Characteristics**

| Conditions are: $V_{SUP}$=12V | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | P | $T_A$= 125°C | $I_{SUPS}$ | | 280 | 500 | µA |
| 3 | C | $T_A$= 25°C | $I_{SUPS}$ | | 23 | | µA |
| Stop Current with 32K OSC and RTC | | | | | | | |
| 4 | C | $T_A$=25°C | $I_{SUPS}$ | | 60 | | µA |

**Table A-15. Pseudo Stop Current Characteristics**

| Conditions are: $V_{SUP}$=12V, API, COP & RTI enabled | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| Pseudo Stop Current with API, COP and RTI enabled | | | | | | | |
| 1 | C | $T_A$= 25°C | $I_{SUPS}$ | | 233 | | µA |
| Pseudo Stop Current with RTC enabled(clock source from external main OSC) | | | | | | | |
| 2 | C | $T_A$= 25°C | $I_{SUPS}$ | | 223 | | µA |

## A.1.10 ADC Conversion Result Reference

The reference voltage $V_{BG}$ is measured under the conditions shown in Table A-16. The value stored in the IFR is the average of 8 consecutive conversions.

**Table A-16. Measurement Conditions**

| Description | Symbol | Value | Unit |
|---|---|---|---|
| Regulator supply voltage | VSUP | 5 | V |
| I/O supply voltage | $V_{DDX, A,M}$ | 5 | V |
| ADC clock | $f_{ADCCLK}$ | 2 | MHz |
| ADC sample time | $t_{SMP}$ | 4 | ADC clock cycles |
| Bus frequency | $f_{bus}$ | 32 | MHz |
| Ambient temperature | $T_A$ | 150 | °C |
| Code execution | | from RAM | |
| NVM activity | | none | |

# Appendix B ADC Electricals

This section describes the characteristics of the analog-to-digital converter.

## B.1 ADC Operating Characteristics

The Table B-1 shows conditions under which the ADC operates.

The following constraints exist to obtain full-scale, full range results:

$VSSA \le VRL \le V_{IN} \le VRH \le VDDA$.

This constraint exists since the sample buffer amplifier can not drive beyond the power supply levels that it ties to. If the input level goes outside of this range it will effectively be clipped.

**Table B-1. ADC Operating Characteristics**

| Supply voltage 4.5V < $V_{DDA}$ < 5.5 V, -40oC < TJ < 150oC | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | Reference potential<br>Low<br>High | <br>$V_{RL}$<br>$V_{RH}$ | <br>$V_{SSA}$<br>$V_{DDA}/2$ | <br>—<br>— | <br>$V_{DDA}/2$<br>$V_{DDA}$ | <br>V<br>V |
| 2 | D | Voltage difference $V_{DDX}$ to $V_{DDA}$ | $\Delta_{VDDX}$ | -0.1 | 0 | 0.1 | V |
| 3 | D | Voltage difference $V_{SSX}$ to $V_{SSA}$ | $\Delta_{VSSX}$ | –0.1 | 0 | 0.1 | V |
| 4 | C | Differential reference voltage(1) | $V_{RH}$-$V_{RL}$ | 3.13 | 5.0 | 5.5 | V |
| 5 | C | ADC Clock Frequency (derived from bus clock via the prescaler bus) | $f_{ADCCLk}$ | 0.25 | | 8.0 | MHz |
| 6 | C | Buffer amplifier turn on time (delay after module start/recovery from Stop mode) | $t_{REC}$ | — | — | 1 | µs |
| 7 | D | ADC disable then re-enable require time | $t_{DISABLE}$ | — | — | 3 | Bus Cycles |
| 8 | D | ADC Conversion Period(2)<br>10 bit resolution:<br>8 bit resolution: | <br>$N_{CONV10}$<br>$N_{CONV8}$ | <br>18<br>16 | | <br>38<br>36 | <br>ADC clock Cycles |

1. The accuracy is reduced if differential reference voltage is less than 4.50 V
2. The minimum time assumes a sample time of 4 ADC clock cycles. The maximum time assumes a sample time of 24 ADC clock cycles.

### B.1.1 Factors Influencing Accuracy

Source resistance, source capacitance and current injection have an influence on the accuracy of the ADC. A further factor is that PortAD pins that are configured as output drivers switching.

### B.1.1.1 Port AD Output Drivers Switching

PortAD output drivers switching can adversely affect the ADC accuracy whilst converting the analog voltage on other PortAD pins because the output drivers are supplied from the VDDA/VSSA ADC supply pins. Although internal design measures are implemented to minimize the affect of output driver noise, it is recommended to configure PortAD pins as outputs only for low frequency, low load outputs. The impact on ADC accuracy is load dependent and not specified. The values specified are valid under condition that no PortAD output drivers switch during conversion.

### B.1.1.2 Source Resistance

Due to the input pin leakage current as specified in conjunction with the source resistance there will be a voltage drop from the signal source to the ADC input. The maximum source resistance $R_S$ specifies results in an error (10-bit resolution) of less than 1/2 LSB (2.5 mV) at the maximum leakage current.

### B.1.1.3 Source Capacitance

When sampling an additional internal capacitor is switched to the input. This can cause a voltage drop due to charge sharing with the external and the pin capacitance. For a maximum sampling error of the input voltage ≤ 1LSB (10-bit resilution), then the external filter capacitor, $C_f \geq 1024 * (C_{INS}–C_{INN})$.

### B.1.1.4 Current Injection

There are two cases to consider.

1. A current is injected into the channel being converted. The channel being stressed has conversion values of 0x3FF (in 10-bit mode) for analog inputs greater than $V_{RH}$ and 0x000 for values less than $V_{RL}$ unless the current is higher than specified as disruptive condition.
2. Current is injected into pins in the neighborhood of the channel being converted. A portion of this current is picked up by the channel (coupling ratio K), This additional current impacts the accuracy of the conversion depending on the source resistance.

   The additional input voltage error on the converted channel can be calculated as:

   $$V_{ERR} = K * R_S * I_{INJ}$$

   with $I_{INJ}$ being the sum of the currents injected into the two pins adjacent to the converted channel.

**Table B-2. ADC Electrical Characteristics**

| Supply voltage 3.13 V < $V_{DDA}$ < 5.5 V, -40°C < TJ < 150oC | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | C | Max input source resistance | $R_S$ | — | — | 10 | KΩ |
| 2 | D | Total input capacitance Non sampling<br>Total input capacitance Sampling | $C_{INN}$<br>$C_{INS}$ | —<br>— | —<br>— | 10<br>16 | pF |
| 3 | D | Input internal Resistance | $R_{INA}$ | - | 5 | 15 | kΩ |
| 4 | C | Disruptive analog input current | $I_{NA}$ | -2.5 | — | 2.5 | mA |
| 5 | C | Coupling ratio positive current injection | $K_p$ | — | — | 1E-4 | A/A |
| 6 | C | Coupling ratio negative current injection | $K_n$ | — | — | 5E-3 | A/A |

## B.1.2 ADC Accuracy

**Table B-3.** specifies the ADC conversion performance excluding any errors due to current injection, input capacitance and source resistance.

### B.1.2.1 ADC Accuracy Definitions

For the following definitions see also Figure B-1.
Differential non-linearity (DNL) is defined as the difference between two adjacent switching steps.

$$DNL(i) = \frac{V_i - V_{i-1}}{1LSB} - 1$$

The integral non-linearity (INL) is defined as the sum of all DNLs:

$$INL(n) = \sum_{i=1}^{n} DNL(i) = \frac{V_n - V_0}{1LSB} - n$$

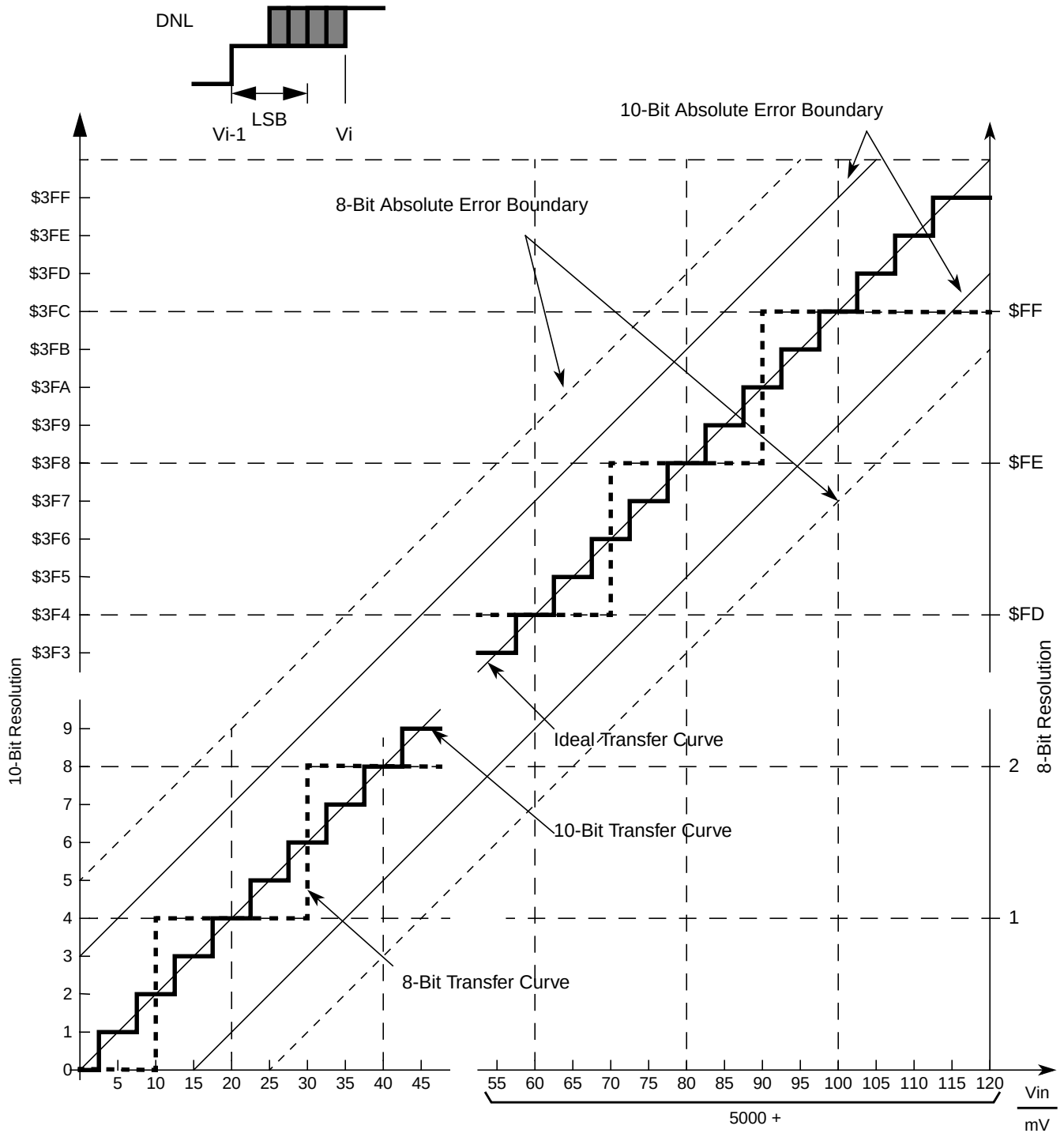



**Figure B-1. ADC Accuracy Definitions**

### Table B-3. ADC Conversion Performance 5V range

| Supply voltage $V_{DDA}$ =5.12 V, -40$^{o}$C < $T_J$ < 150oC. $V_{REF}$ = $V_{RH}$ - $V_{RL}$ = 5.12V. $f_{ADCCLK}$ = 8.0MHz<br>The values are tested to be valid with no PortAD output drivers switching simultaneous with conversions. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating(1)** | | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 5 | P | Resolution | 10-Bit | LSB | | 5 | | mV |
| 6 | P | Differential Nonlinearity | 10-Bit | DNL | -1 | ±0.5 | 1 | counts |
| 7 | P | Integral Nonlinearity | 10-Bit | INL | -2 | ±1 | 2 | counts |
| 8 | P | Absolute Error[2.] | 10-Bit | AE | -3 | ±2 | 3 | counts |
| 9 | C | Resolution | 8-Bit | LSB | | 20 | | mV |
| 10 | C | Differential Nonlinearity | 8-Bit | DNL | -0.5 | ±0.3 | 0.5 | counts |
| 11 | C | Integral Nonlinearity | 8-Bit | INL | -1 | ±0.5 | 1 | counts |
| 12 | C | Absolute Error[2.] | 8-Bit | AE | -1.5 | ±1 | 1.5 | counts |

1. The 8-bit mode operation is structurally tested in production test. Absolute values are tested in 10-bit mode.

# Appendix C PLL Electrical Specifications

## C.1 Reset, Oscillator and PLL

### C.1.1 Phase Locked Loop

#### C.1.1.1 Jitter Information

With each transition of the feedback clock, the deviation from the reference clock is measured and the input voltage to the VCO is adjusted accordingly.The adjustment is done continuously with no abrupt changes in the VCOCLK frequency. Noise, voltage, temperature and other factors cause slight variations in the control loop resulting in a clock jitter. This jitter affects the real minimum and maximum clock periods as illustrated in **Figure C-1.**.

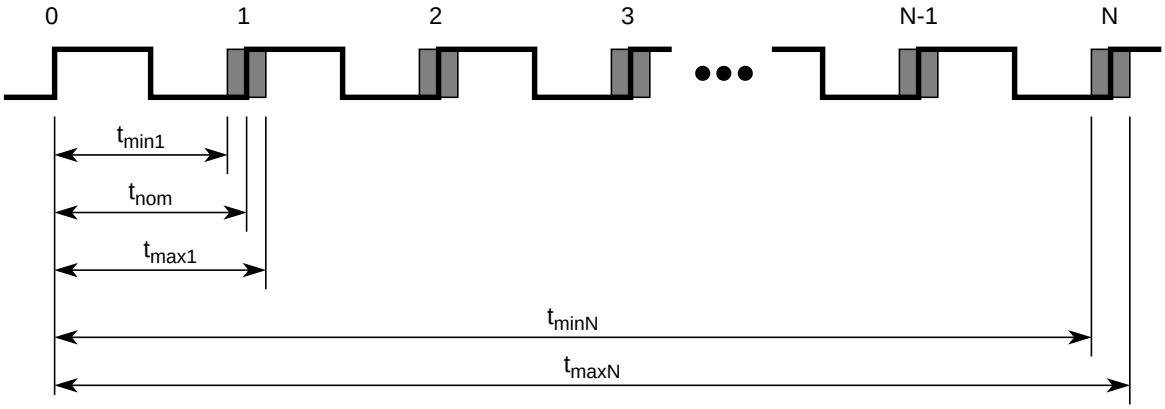



**Figure C-1. Jitter Definitions**

The relative deviation of $t_{nom}$ is at its maximum for one clock period, and decreases towards zero for larger number of clock periods (N).

Defining the jitter as:

$$J(N) = \max\left(\left|1 - \frac{t_{max}(N)}{N \cdot t_{nom}}\right|, \left|1 - \frac{t_{min}(N)}{N \cdot t_{nom}}\right|\right)$$

The following equation is a good fit for the maximum jitter:

$$J(N) = \frac{j_1}{\sqrt{N(POSTDIV + 1)}}$$

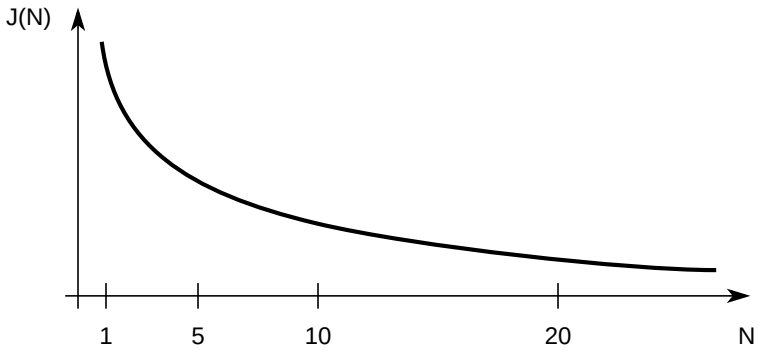



**Figure C-2. Maximum Bus Clock Jitter Approximation (N = number of Bus Cycles)**

**NOTE**

On timers and serial modules a prescaler will eliminate the effect of the jitter to a large extent.

**Table C-1. ipll_1vdd_ll18 Characteristics**

| Conditions are 4.5 V < $V_{DDX}$< 5.5 V junction temperature from –40°C to +150°C, unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | VCO frequency during system reset | $f_{VCORST}$ | 8 | | 32 | MHz |
| 2 | C | VCO locking range | $f_{VCO}$ | 32 | | 64 | MHz |
| 3 | C | Reference Clock | $f_{REF}$ | 1 | | | MHz |
| 4 | D | Lock Detection | $\|\Delta_{Lock}\|$ | 0 | | 1.5 | %(1) |
| 5 | D | Un-Lock Detection | $\|\Delta_{unl}\|$ | 0.5 | | 2.5 | %1 |
| 7 | C | Time to lock | $t_{lock}$ | | | 150 + 256/$f_{REF}$ | µs |
| 8 | C | Jitter fit parameter 1(2) | $j_1$ | | | 2 | % |
| 9 | P | PLL Clock Monitor Failure assert frequency | $f_{PMFA}$ | 0.45 | 1.1 | 1.6 | MHz |

1. % deviation from target frequency
2. $f_{REF}$ = 1MHz, $f_{BUS}$ = 32MHz

# Appendix D IRC Electrical Specifications

**Table D-1. IRC electrical characteristics**

| Num | C | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | P | Junction Temperature - 40 to 150 Celsius<br>Internal Reference Frequency, factory trimmed | $f_{IRC1M_TRIM}$ | 0.987 | 1 | 1.013 | MHz |

# Appendix E LCD Electrical Specifications

## E.1 LCD Driver

**Table E-1. LCD40F4B Driver Electrical Characteristics**

| Characteristic | C | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| LCD Output Impedance(BP[3:0],FP[39:0]) for outputs to charge to higher voltage level or to GND [1)] | D | $Z_{BP/FP}$ | - | - | 5.0 | kΩ |
| LCD Output Current (BP[3:0],FP[39:0]) for outputs to discharge to lower voltage level except GND [1)] | D | $I_{BP/FP}$ | 50 | - | - | µA |

1) Outputs measured one at a time, low impedance voltage source connected to the VLCD pin.

The 1/3, 1/2 and 2/3 VLCD voltage levels are buffered internally with an asymmetric output stage, as shown in **Figure E-1.**

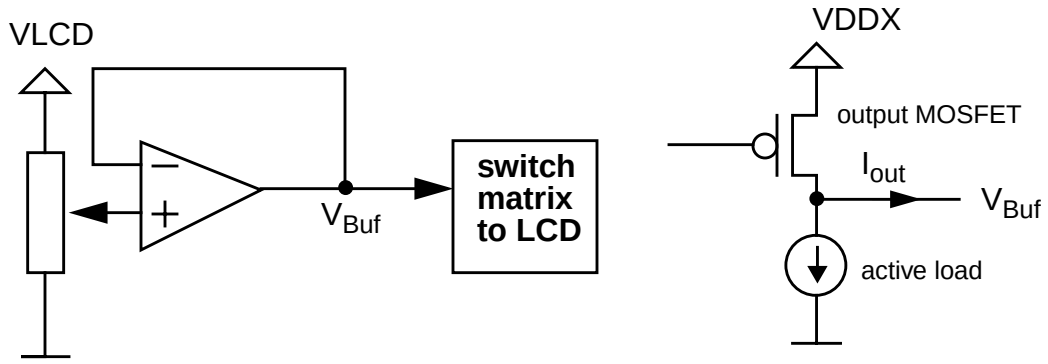



**Figure E-1. Buffer configuration (left) and buffer output stage (right)**

The switching matrix applies a capacitive load (LCD elements) to the buffer output. The charge excites the buffer output voltage $V_{Buf}$ from the target output voltage which can be 1/3, 1/2 or 2/3 VLCD. After a positive spike on $V_{Buf}$ a frontplane or backplane is discharged by an active load with a constant current. After a negative spike on $V_{Buf}$ the output is charged through a transistor which is switched on and which behaves like a resistor. Simplified output voltage transients are shown in **Figure E-2.**. The shown

transients emphasize the spikes and the voltage recovery. They are not to scale. The buffer output characteristic is shown in **Figure E-3.**. The resistive output characteristic is also valid if an output is forced to GND or VLCD.

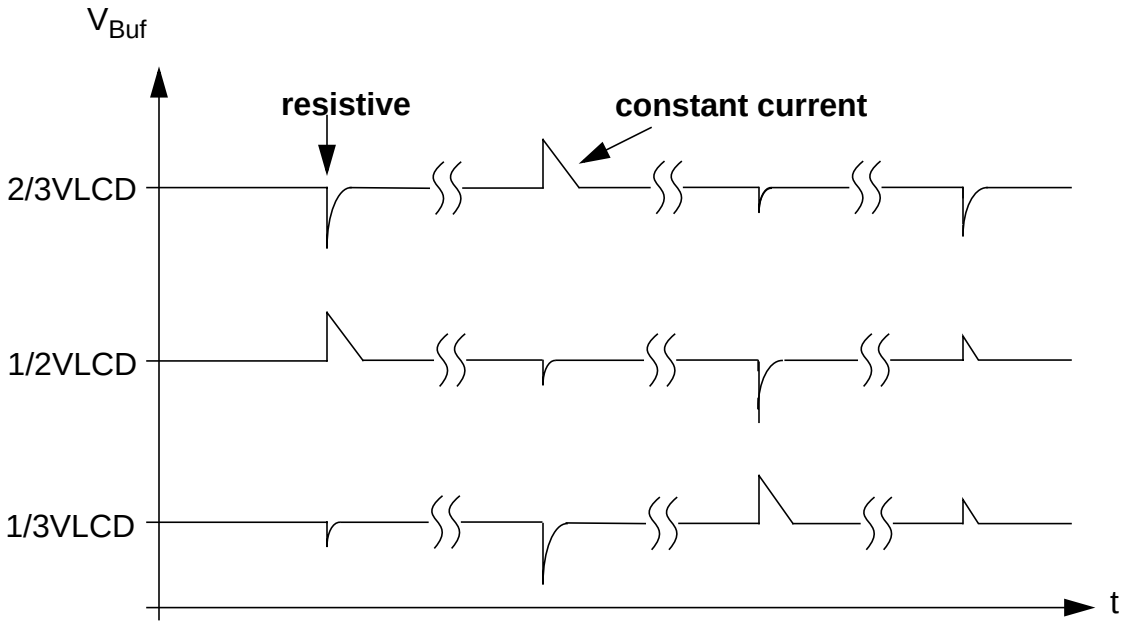



**Figure E-2. $V_{Buf}$ transients (not to scale)**

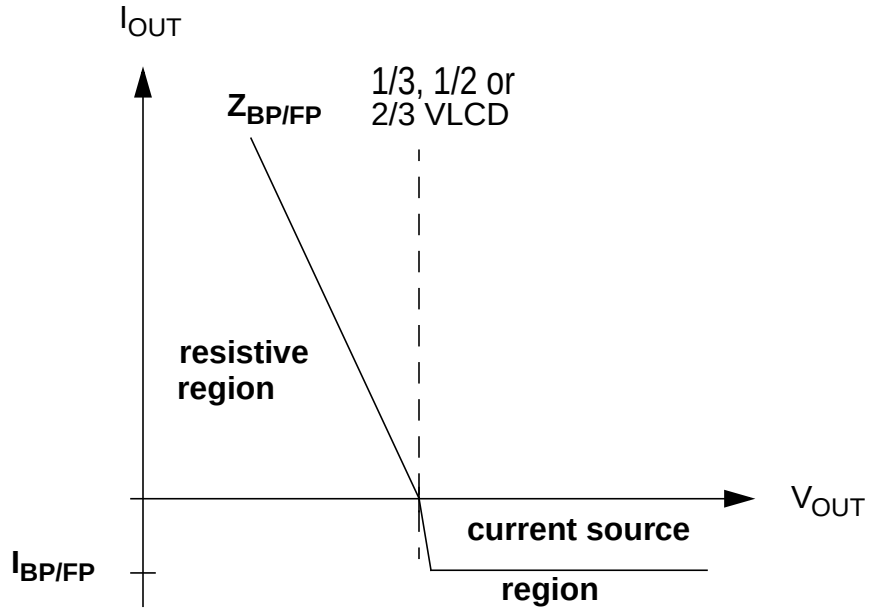



**Figure E-3. buffer output characteristic**

# Appendix F MSCAN Electrical Specifications

## F.1 MSCAN Electrical Characteristics

**Table F-1. MSCAN Wake-up Pulse Characteristics**

| Conditions are shown in **Table F-1.** unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | MSCAN wakeup dominant pulse filtered | $t_{WUP}$ | — | — | 1.5 | µs |
| 2 | P | MSCAN wakeup dominant pulse pass | $t_{WUP}$ | 5 | — | — | µs |

# Appendix G
# NVM Electrical Parameters

## G.1 NVM Timing Parameters

The time base for all NVM program or erase operations is derived from the bus clock using the FCLKDIV register. The frequency of this derived clock must be set within the limits specified as $f_{NVMOP}$. The NVM module does not have any means to monitor the frequency and will not prevent program or erase operation at frequencies above or below the specified minimum. When attempting to program or erase the NVM module at a lower frequency, a full program or erase transition is not assured.

The following sections provide equations which can be used to determine the time required to execute specific flash commands. All timing parameters are a function of the bus clock frequency, $f_{NVMBUS}$. All program and erase times are also a function of the NVM operating frequency, $f_{NVMOP}$. A summary of key timing parameters can be found in Table G-1.

**Table G-1. NVM Timing Characteristics (32 MHz - 64KB PFlash - 2KB EEPROM)**

| Num | Command | $f_{NVMOP}$ cycle | $f_{NVMBUS}$ cycle | Symbol | Min (1) | Typ(2) | Max(3) | Worst (4) | Unit |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Erase Verify All Blocks[4,5] | 0 | 34528 | $t_{RD1ALL}$ | 1.08 | 1.08 | 2.16 | 69.06 | ms |
| 2 | Erase Verify Block (Pflash)(5) | 0 | 33320 | $t_{RD1BLK_P}$ | 1.04 | 1.04 | 2.08 | 66.64 | ms |
| 3 | Erase Verify Block (EEPROM)(6) | 0 | 1591 | $t_{RD1BLK_D}$ | 0.05 | 0.05 | 0.10 | 3.18 | ms |
| 4 | Erase Verify P-Flash Section | 0 | 505 | $t_{RD1SEC}$ | 0.02 | 0.02 | 0.06 | 2.02 | ms |
| 5 | Read Once | 0 | 481 | $t_{RDONCE}$ | 15.03 | 15.03 | 15.03 | 481.00 | us |
| 6 | Program P-Flash (4 Word) | 164 | 3077 | $t_{PGM_4}$ | 0.25 | 0.26 | 0.55 | 12.51 | ms |
| 7 | Program Once | 164 | 3054 | $t_{PGMONCE}$ | 0.25 | 0.26 | 0.26 | 3.26 | ms |
| 8 | Erase All Blocks[4,5] | 100066 | 34991 | $t_{ERSALL}$ | 96.39 | 101.16 | 102.25 | 195.06 | ms |
| 9 | Erase Flash Block (Pflash)[4] | 100060 | 33681 | $t_{ERSBLK_P}$ | 96.35 | 101.11 | 102.17 | 192.44 | ms |
| 10 | Erase Flash Block (EEPROM)[5] | 100060 | 1922 | $t_{ERSBLK_D}$ | 95.36 | 100.12 | 100.18 | 128.92 | ms |
| 11 | Erase P-Flash Sector | 20015 | 914 | $t_{ERSPG}$ | 19.09 | 20.04 | 20.07 | 26.85 | ms |
| 12 | Unsecure Flash | 100066 | 35056 | $t_{UNSECU}$ | 96.40 | 101.16 | 102.26 | 195.19 | ms |
| 13 | Verify Backdoor Access Key | 0 | 493 | $t_{VFYKEY}$ | 15.41 | 15.41 | 15.41 | 493.00 | us |
| 14 | Set User Margin Level | 0 | 427 | $t_{MLOADU}$ | 13.34 | 13.34 | 13.34 | 427.00 | us |
| 15 | Set Factory Margin Level | 0 | 436 | $t_{MLOADF}$ | 13.63 | 13.63 | 13.63 | 436.00 | us |
| 16 | Erase Verify EEPROM Section | 0 | 583 | $t_{DRD1SEC}$ | 0.02 | 0.02 | 0.07 | 2.33 | ms |
| 17 | Program EEPROM (1 Word) | 68 | 1657 | $t_{DPGM_1}$ | 0.12 | 0.12 | 0.28 | 6.71 | ms |
| 18 | Program EEPROM (2 Word) | 136 | 2660 | $t_{DPGM_2}$ | 0.21 | 0.22 | 0.47 | 10.81 | ms |
| 19 | Program EEPROM (3 Word) | 204 | 3663 | $t_{DPGM_3}$ | 0.31 | 0.32 | 0.66 | 14.91 | ms |
| 20 | Program EEPROM (4 Word) | 272 | 4666 | $t_{DPGM_4}$ | 0.40 | 0.42 | 0.86 | 19.00 | ms |

**Table G-1. NVM Timing Characteristics (32 MHz - 64KB PFlash - 2KB EEPROM)**

| Num | Command | $f_{NVMOP}$ cycle | $f_{NVMBUS}$ cycle | Symbol | Min (1) | Typ(2) | Max(3) | Worst (4) | Unit |
|---|---|---|---|---|---|---|---|---|---|
| 21 | Erase EEPROM Sector | 5015 | 810 | $t_{DERSPG}$ | 4.80 | 5.04 | 20.49 | 38.85 | ms |
| 22 | Protection Override | 0 | 475 | $t_{PRTOVRD}$ | 14.84 | 14.84 | 14.84 | 475.00 | us |

1. Minimum times are based on maximum $f_{NVMOP}$ and maximum $f_{NVMBUS}$
2. Typical times are based on typical $f_{NVMOP}$ and typical $f_{NVMBUS}$
3. Maximum times are based on typical $f_{NVMOP}$ and typical $f_{NVMBUS}$ plus aging
4. Worst times are based on minimum $f_{NVMOP}$ and minimum $f_{NVMBUS}$ plus aging
5. Affected by Pflash size
6. Affected by EEPROM size

## G.2 NVM Reliability Parameters

The reliability of the NVM blocks is guaranteed by stress test during qualification, constant process monitors and burn-in to screen early life failures.

The data retention and program/erase cycling failure rates are specified at the operating conditions noted. The program/erase cycle count on the sector is incremented every time a sector or mass erase event is executed.

**NOTE**

All values shown in Table G-2 are preliminary and subject to further characterization.

**Table G-2. NVM Reliability Characteristics**

| NUM | C | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| | | **Program Flash Arrays** | | | | | |
| 1 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C(1) after up to 10,000 program/erase cycles | $t_{NVMRET}$ | 20 | 100(2) | — | Years |
| 2 | C | Program Flash number of program/erase cycles (-40°C ≤ tj ≤ 150°C) | $n_{FLPE}$ | 10K | 100K(3) | — | Cycles |
| | | **EEPROM Array** | | | | | |
| 3 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C1 after up to 100,000 program/erase cycles | $t_{NVMRET}$ | 5 | 1002 | — | Years |
| 4 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C1 after up to 10,000 program/erase cycles | $t_{NVMRET}$ | 10 | 1002 | — | Years |
| 5 | C | Data retention at an average junction temperature of $T_{Javg}$ = 85°C1 after less than 100 program/erase cycles | $t_{NVMRET}$ | 20 | 1002 | — | Years |
| 6 | C | EEPROM number of program/erase cycles (-40°C ≤ tj ≤ 150°C) | $n_{FLPE}$ | 100K | 500K3 | — | Cycles |

1. $T_{Javg}$ does not exceed 85°C in a typical temperature profile over the lifetime of a consumer, industrial or automotive application.
2. Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale defines Typical Data Retention, please refer to Engineering Bulletin EB618

3. Spec table quotes typical endurance evaluated at 25°C for this product family. For additional information on how Freescale defines Typical Endurance, please refer to Engineering Bulletin EB619.

# Appendix H
# BATS Electrical Specifications

This section describe the electrical characteristics of the Supply Voltage Sense module.

## H.1 Maximum Ratings

**Table H-1. Maximum ratings of the Supply Voltage Sense - (BATS).**

| Characteristics noted under conditions 5.5V ≤ VSUP ≤ 18 V, -40˚C ≤ $T_J$ ≤ 150˚C[1] unless otherwise noted. Typical values noted reflect the approximate parameter mean at $T_A$ = 25˚C[2] under nominal conditions unless otherwise noted. | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Ratings** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | VSENSE Max Rating | $V_{VSENSE_M}$ | -27 | – | 42 | V |

1 $T_J$: Junction Temperature

2 $T_A$: Ambient Temperature

# H.2 Static Electrical Characteristics

**Table H-2. Static Electrical Characteristics - Supply Voltage Sense - (BATS).**

| Characteristics noted under conditions 5.5V ≤ VSUP ≤ 18 V, -40˚C ≤ $T_J$ ≤ 150˚C[1] unless otherwise noted. Typical values noted reflect the approximate parameter mean at $T_A$ = 25˚C[2] under nominal conditions unless otherwise noted. | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Ratings** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | P | Low Voltage Warning (LBI 1)<br>Assert (Measured on selected pin, falling edge)<br>Deassert (Measured on selected pin, rising edge)<br>Hysteresis (measured on selected pin) | <br>$V_{LBI1_A}$<br>$V_{LBI1_D}$<br>$V_{LBI1_H}$ | <br>5<br>–<br>– | <br>5.5<br>–<br>0.4 | <br>6<br>6.5<br>– | <br>V<br>V<br>V |
| 2 | P | Low Voltage Warning (LBI 2)<br>Assert (Measured on selected pin, falling edge)<br>Deassert (Measured on selected pin, rising edge)<br>Hysteresis (measured on selected pin) | <br>$V_{LBI2_A}$<br>$V_{LBI2_D}$<br>$V_{LBI2_H}$ | <br>6<br>–<br>– | <br>6.75<br>–<br>0.4 | <br>7<br>7.75<br>– | <br>V<br>V<br>V |
| 3 | P | Low Voltage Warning (LBI 3)<br>Assert (Measured on selected pin, falling edge)<br>Deassert (Measured on selected pin, rising edge)<br>Hysteresis (measured on selected pin) | <br>$V_{LBI3_A}$<br>$V_{LBI3_D}$<br>$V_{LBI3_H}$ | <br>7<br>–<br>– | <br>7.75<br>–<br>0.4 | <br>8.5<br>9<br>– | <br>V<br>V<br>V |
| 4 | P | Low Voltage Warning (LBI 4)<br>Assert (Measured on selected pin, falling edge)<br>Deassert (Measured on selected pin, rising edge)<br>Hysteresis (measured on selected pin) | <br>$V_{LBI4_A}$<br>$V_{LBI4_D}$<br>$V_{LBI4_H}$ | <br>8<br>–<br>– | <br>9<br>–<br>0.4 | <br>10<br>10.5<br>– | <br>V<br>V<br>V |
| 5 | P | High Voltage Warning (HBI 1)<br>Assert (Measured on selected pin, rising edge)<br>Deassert (Measured on selected pin, falling edge)<br>Hysteresis (measured on selected pin) | <br>$V_{HBI1_A}$<br>$V_{HBI1_D}$<br>$V_{HBI1_H}$ | <br>15<br>14.5<br>– | <br>16.5<br>–<br>1.0 | <br>18<br>–<br>– | <br>V<br>V<br>V |
| 6 | P | High Voltage Warning (HBI 2)<br>Assert (Measured on selected pin, rising edge)<br>Deassert (Measured on selected pin, falling edge)<br>Hysteresis (measured on selected pin) | <br>$V_{HBI2_A}$<br>$V_{HBI2_D}$<br>$V_{HBI2_H}$ | <br>25<br>24<br>– | <br>27.5<br>–<br>1.0 | <br>29<br>–<br>– | <br>V<br>V<br>V |
| 7 | T | Pin Input Divider Ratio<br>$Ratio_{VSENSE} = V_{SENSE} / V_{ADC}$[3]<br>$Ratio_{VSUP} = V_{SUP} / V_{ADC}$<br><br>5.5V < VSENSE < 29 V; 5.5V < VSUP < 29 V | <br>$Ratio_{VSENSE}$<br>$Ratio_{VSUP}$ | <br>–<br>– | <br>9<br>9 | <br>–<br>– | <br>–<br>– |
| 8 | C | Analog Input Matching<br><br>Absolute Error on $V_{ADC}$<br>- compared to $V_{SENSE}$ / $Ratio_{VSENSE}$<br>- compared to $V_{SUP}$ / $Ratio_{VSUP}$ | $AI_{Matching}$ | – | +-2% | +-5% | – |
| 9 | D | VSENSE Series Resistor<br><br>Required to be placed externally at VSENSE pin. | $R_{VSENSE_R}$ | 9.5 | 10 | 10.5 | kΩ |
| 10 | D | VSENSE Impedance<br><br>If path to ground is enabled. | $R_{VSEN_IMP}$ | – | 350 | – | kΩ |

[1] $T_J$: Junction Temperature

2 $T_A$: Ambient Temperature
3 $V_{ADC}$: Voltage accessible at the ADC input channel

## H.3 Dynamic Electrical Characteristics

**Table H-3. Dynamic Electrical Characteristics - Supply Voltage Sense - (BATS).**

| Characteristics noted under conditions 5.5V ≤ VSUP ≤ 18 V, -40˚C ≤ $T_J$ ≤ 150˚C[1] unless otherwise noted. Typical values noted reflect the approximate parameter mean at $T_A$ = 25˚C[2] under nominal conditions unless otherwise noted. | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Ratings** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | D | Enable Stabilisation Time | $T_{EN_UNC}$ | – | 1 | – | µs |
| 2 | D | Voltage Warning Low Pass Filter | $f_{VWLP_filter}$ | – | 0.5 | – | MHz |

1 $T_J$: Junction Temperature
2 $T_A$: Ambient Temperature

# Appendix I VREG Electrical Specifications

**Table I-1. Voltage Regulator Electrical Characteristics**

| -40$^{o}$C <= TJ <= 150$^{o}$C unless noted otherwise, VDDA, VDDM and VDDX must be shorted on the application board. | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Characteristic** | **Symbol** | **Min** | **Typical** | **Max** | **Unit** |
| 1 | P | Input Voltages | $V_{SUP}$ | 3.5 | — | 40 | V |
| 2 | P | Output Voltage VDDX<br>Full Performance Mode $V_{SUP}$ > =6V<br>Full Performance Mode 5.5V <= $V_{SUP}$ <=6V<br>Full Performance Mode 3.5V <= $V_{SUP}$ <=5.5V<br>Reduced Performance Mode (stopmode)<br>$V_{SUP}$ > =3.5V | $V_{DDX}$ | <br>4.85<br>4.50<br>3.13<br>2.5 | <br>5.0<br>5.0<br>-<br>5.5 | <br>5.15<br>5.25<br>5.25<br>5.75 | <br>V<br>V<br>V<br>V |
| 3 | P | Load Current VDDX(1),(2) (3)without external PNP<br>Full Performance Mode $V_{SUP}$ > =6V<br>Full Performance Mode 3.5V <= $V_{SUP}$ <=6V<br>Reduced Performance Mode (stopmode) | $I_{DDX}$ | <br>0<br>0<br>0 | <br>-<br>-<br>- | <br>70<br>25<br>5 | <br>mA<br>mA<br>mA |
| 4 | P | Low Voltage Interrupt Assert Level(4)<br>Low Voltage Interrupt Deassert Level | $V_{LVIA}$<br>$V_{LVID}$ | 4.04<br>4.19 | 4.23<br>4.38 | 4.40<br>4.49 | V<br>V |
| 5a | P | VDDX Low Voltage Reset deassert(5) | $V_{LVRXD}$ | — | — | 3.13 | V |
| 5b | P | VDDX Low Voltage Reset assert | $V_{LVRXA}$ | 2.97 | 3.02 | — | V |
| 6 | C | Trimmed ACLK output frequency | $f_{ACLK}$ | | 20 | | KHz |
| 7 | C | Trimmed ACLK internal clock $\Delta f / f_{nominal}$ (6) | $df_{ACLK}$ | - 5% | — | + 5% | — |
| 8 | D | The first period after enabling the counter by APIFE might be reduced by API start up delay | $t_{sdel}$ | — | — | 100 | µs |
| 9 | T | Temperature Sensor Slope | $dV_{HT}$ | 5.05 | 5.25 | 5.45 | mV/$^{o}$C |
| 10 | T | Temperature Sensor Output Voltage | $V_{HT}$ | - | 2.4 | - | V |
| 11 | T | High Temperature Interrupt Assert(7)<br>High Temperature Interrupt Deassert | $T_{HTIA}$<br>$T_{HTID}$ | 120<br>110 | 132<br>122 | 144<br>134 | $^{o}$C<br>$^{o}$C |
| 12 | P | Bandgap output voltage | $V_{BG}$ | 1.13 | 1.22 | 1.32 | V |
| 13 | P | Base Current For External PNP(VDDX) | $I_{BCTL}$ | 1.5 | — | — | mA |
| 14 | D | Recover time from STOP | $t_{STP_REC}$ | — | 23 | — | µs |

1. For the given maximum load currents and $V_{SUP}$ input voltages, the MCU will stay out of reset.
2. Load current without the use of external PNP transistor
3. Please note that the core current is derived from VDDX
4. LVI is monitored on the VDDA supply domain
5. LVRX is monitored on the VDDX supply domain only active during full performance mode. During reduced performance mode (stopmode) voltage supervision is solely performed by the POR block monitoring core VDD.
6. The ACLK trimming must be set that the minimum period equals to 0.2ms
7. VREGHTTR=0x88

**NOTE**

The LVR monitors the voltages VDD, VDDF and VDDX. If the voltage drops on these supplies to a level which could prohibit the correct function (e.g. code execution) of the microcontroller, the LVR triggers.

# Appendix J Electrical Characteristics for the Oscillator (OSCLCPcr)

## J.1 OSCLCP Electrical characteristics

**Table J-1. XOSCLCP Characteristics**

| Conditions are shown in **Table J-1.** unless otherwise noted | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Num** | **C** | **Rating** | **Symbol** | **Min** | **Typ** | **Max** | **Unit** |
| 1 | C | Nominal crystal or resonator frequency | $f_{OSC}$ | 4.0 | | 20 | MHz |
| 2 | P | Startup Current | $i_{OSC}$ | 100 | | | µA |
| 3a | C | Oscillator start-up time (4MHz)[1] | $t_{UPOSC}$ | — | 2 | 10 | ms |
| 3b | C | Oscillator start-up time (8MHz)[1] | $t_{UPOSC}$ | — | 1.6 | 8 | ms |
| 3c | C | Oscillator start-up time (16MHz)[1] | $t_{UPOSC}$ | — | 1 | 5 | ms |
| 3d | C | Oscillator start-up time (20MHz)[1] | $t_{UPOSC}$ | — | 1 | 4 | ms |
| 4 | P | Clock Monitor Failure Assert Frequency | $f_{CMFA}$ | 200 | 450 | 1200 | kHz |
| 5 | D | Input Capacitance (EXTAL, XTAL pins) | $C_{IN}$ | | 7 | | pF |
| 6 | C | EXTAL Pin Input Hysteresis | $V_{HYS,EXTAL}$ | — | 120 | — | mV |
| 7 | C | EXTAL Pin oscillation amplitude (loop controlled Pierce) | $V_{PP,EXTAL}$ | — | 0.9 | — | V |
| 8 | C | EXTAL Pin oscillation amplitude (full swing Pierce) | $V_{PP,EXTAL}$ | — | 1.8 | — | V |
| 9 | D | EXTAL Pin oscillation required amplitude[2] | $V_{PP,EXTAL}$ | 0.8 | — | 1.5 | V |
| 10 | D | EXTAL Pin oscillation required amplitude[3] | $V_{PP,EXTAL}$ | 0.8 | — | 2.4 | V |

1. These values apply for carefully designed PCB layouts with capacitors that match the crystal/resonator requirements.
2. Needs to be measured at room temperature on the application board using a probe with very low (<=5pF) input capacitance.
3. Needs to be measured at room temperature on the application board using a probe with very low (<=5pF) input capacitance.

# Appendix K OSC32K Electrical Specifications

## K.1 DC Electrical Specifications

**Table K-1. OSC32K DC Electrical Specifications**

| Num | C | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | C | OSC32K Supply Current $V_{DDA}$=5.0 | $I_{DDOSC}$ | | 40 | 60 | μA |
| 2 | D | Allowed Impedence on 32K_EXTAL/XTAL pins | $I_{lk_ext}$ | 10 | — | — | MΩ |
| 3 | — | 32K_EXTAL Load Capacitance<br>32K_XTAL Load Capacitance | $C_x$<br>$C_y$ | —(1) | | | |
| 4 | — | External Feedback resistor | $R_F$ | | 10 | | MΩ |
| 5 | — | External Series resistor | $R_S$ | | 200 | | KΩ |
| 6 | T | Peak-to-Peak Amplitude of Oscillation | $V_{pp}$ | 0.75*$V_{DDX}$ | $V_{DDX}$ | | V |

1. See crystal or resonator manufacturer's recommendation

## K.2 Frequency Specifications

**Table K-2. OSC32K Frequency Specifications**

| Num | C | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D | Oscillator crystal or resonator frequency | $f_{osc_lo}$ | 32 | — | 40 | kHz |
| 2 | C | Crystal Start-up Time (1), (2) | $t_{cst}$ | — | 1000 | 4000 | ms |

1. This parameter is characterized before qualification rather than 100% tested.
2. Proper PC board layout procedures must be followed to achieve specifications.

# Appendix L SPI Electrical Specifications

This section provides electrical parametrics and ratings for the SPI.

In **Table L-1.** the measurement conditions are listed.

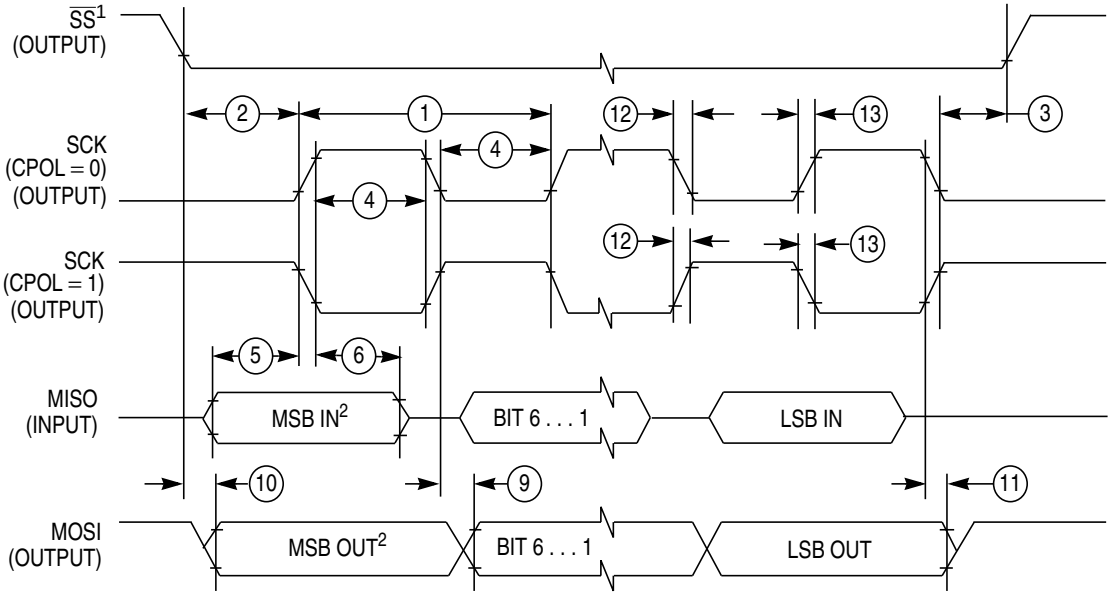
**Table L-1. Measurement Conditions**

| Description | Value | Unit |
|---|---|---|
| Drive mode | full drive mode | — |
| Load capacitance $C_{LOAD}$[(1)], on all outputs | 50 | pF |
| Thresholds for delay measurement points | (35% / 65%) VDDX | V |

1. Timing specified for equal load on all SPI output pins. Avoid asymmetric load.

## L.1 Master Mode

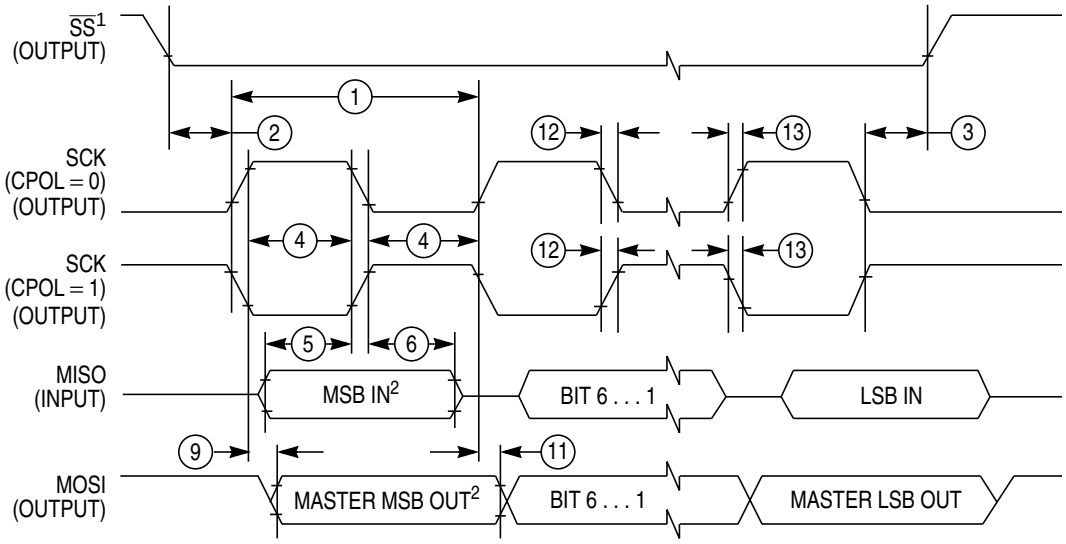
In **Figure L-1.** the timing diagram for master mode with transmission format CPHA=0 is depicted.




1. If enabled.
2. LSBFE = 0. For LSBFE = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure L-1. SPI Master Timing (CPHA=0)**

In **Figure L-2.** the timing diagram for master mode with transmission format CPHA=1 is depicted.




1. If enabled.
2. LSBFE = 0. For LSBFE = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure L-2. SPI Master Timing (CPHA=1)**

In **Table L-2.** the timing characteristics for master mode are listed.

**Table L-2. SPI Master Mode Timing Characteristics**

| Num | C | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D | SCK Frequency | $f_{sck}$ | 1/2048 | — | 1/2(1) | $f_{bus}$ |
| 1 | D | SCK Period | $t_{sck}$ | $2^1$ | — | 2048 | $t_{bus}$ |
| 2 | D | Enable Lead Time | $t_{lead}$ | — | 1/2 | — | $t_{sck}$ |
| 3 | D | Enable Lag Time | $t_{lag}$ | — | 1/2 | — | $t_{sck}$ |
| 4 | D | Clock (SCK) High or Low Time | $t_{wsck}$ | — | 1/2 | — | $t_{sck}$ |
| 5 | D | Data Setup Time (Inputs) | $t_{su}$ | 8 | — | — | ns |
| 6 | D | Data Hold Time (Inputs) | $t_{hi}$ | 8 | — | — | ns |
| 9 | D | Data Valid after SCK Edge | $t_{vsck}$ | — | — | 15 | ns |
| 10 | D | Data Valid after $\overline{SS}$ fall (CPHA=0) | $t_{vss}$ | — | — | 15 | ns |
| 11 | D | Data Hold Time (Outputs) | $t_{ho}$ | 0 | — | — | ns |
| 12 | D | Rise and Fall Time Inputs | $t_{rfi}$ | — | — | 8 | ns |
| 13 | D | Rise and Fall Time Outputs | $t_{rfo}$ | — | — | 8 | ns |

1. pls. see **Figure L-3.**

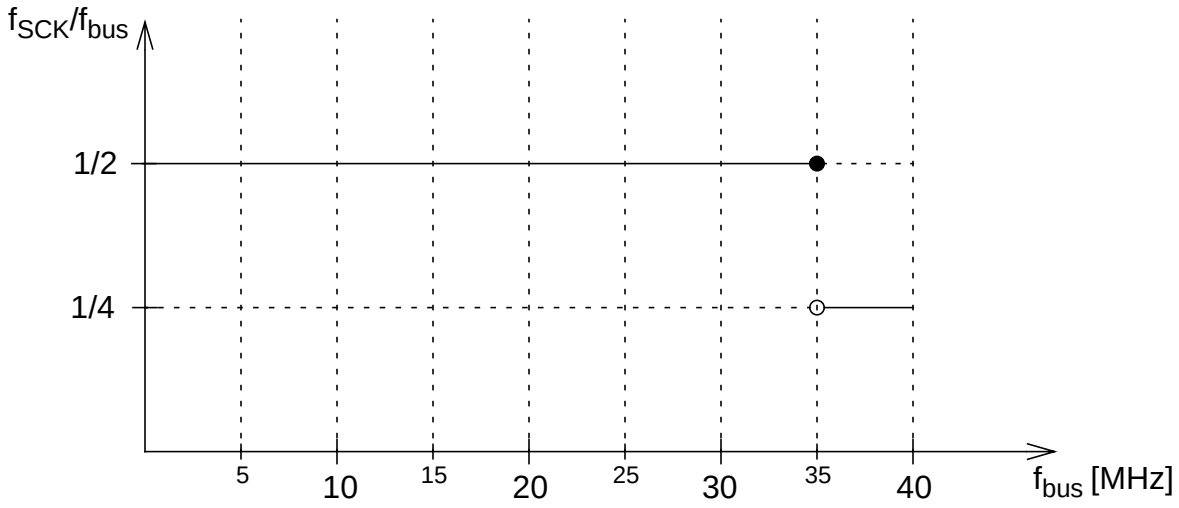



**Figure L-3. Derating of maximum $f_{SCK}$ to $f_{bus}$ ratio in Master Mode**

In Master Mode the allowed maximum $f_{SCK}$ to $f_{bus}$ ratio (= minimum Baud Rate Divisor, pls. see SPI Block Guide) derates with increasing $f_{bus}$, please see **Figure L-3.**.

## L.2 Slave Mode

In **Figure L-4.** the timing diagram for slave mode with transmission format CPHA=0 is depicted.

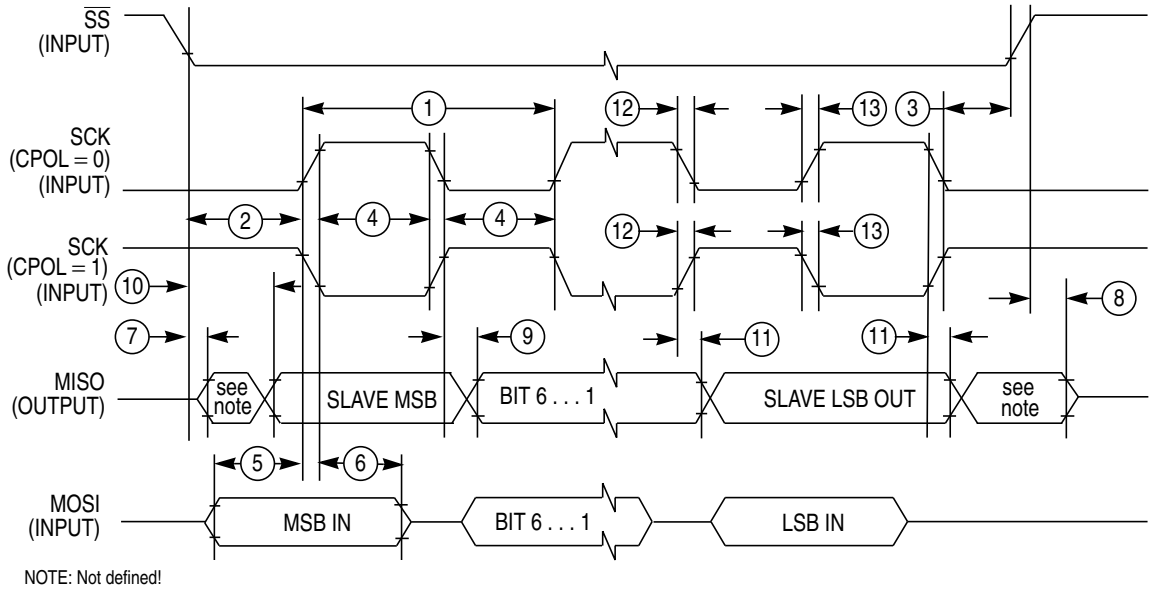



**Figure L-4. SPI Slave Timing (CPHA=0)**

In **Figure L-5.** the timing diagram for slave mode with transmission format CPHA=1 is depicted.

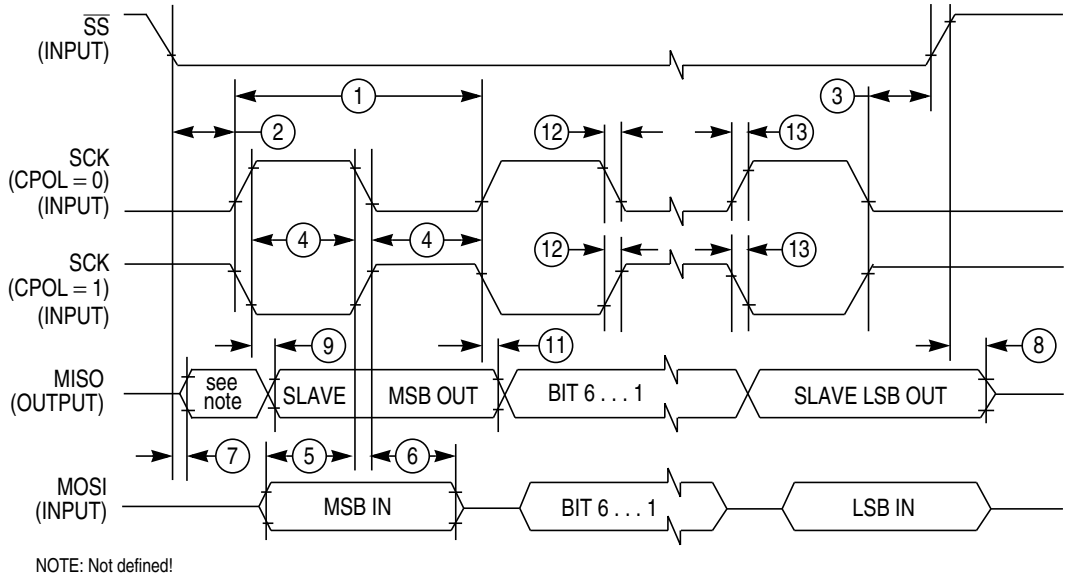



**Figure L-5. SPI Slave Timing (CPHA=1)**

In **Table L-3.** the timing characteristics for slave mode are listed.

**Table L-3. SPI Slave Mode Timing Characteristics**

| Num | C | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D | SCK Frequency | $f_{sck}$ | DC | — | 1/4 | $f_{bus}$ |
| 1 | D | SCK Period | $t_{sck}$ | 4 | — | ∞ | $t_{bus}$ |
| 2 | D | Enable Lead Time | $t_{lead}$ | 4 | — | — | $t_{bus}$ |
| 3 | D | Enable Lag Time | $t_{lag}$ | 4 | — | — | $t_{bus}$ |
| 4 | D | Clock (SCK) High or Low Time | $t_{wsck}$ | 4 | — | — | $t_{bus}$ |
| 5 | D | Data Setup Time (Inputs) | $t_{su}$ | 8 | — | — | ns |
| 6 | D | Data Hold Time (Inputs) | $t_{hi}$ | 8 | — | — | ns |
| 7 | D | Slave Access Time (time to data active) | $t_a$ | — | — | 20 | ns |
| 8 | D | Slave MISO Disable Time | $t_{dis}$ | — | — | 22 | ns |
| 9 | D | Data Valid after SCK Edge | $t_{vsck}$ | — | — | $29 + 0.5 \cdot t_{bus}$ (1) | ns |
| 10 | D | Data Valid after $\overline{SS}$ fall | $t_{vss}$ | — | — | $29 + 0.5 \cdot t_{bus}$ [1] | ns |
| 11 | D | Data Hold Time (Outputs) | $t_{ho}$ | 20 | — | — | ns |
| 12 | D | Rise and Fall Time Inputs | $t_{rfi}$ | — | — | 8 | ns |
| 13 | D | Rise and Fall Time Outputs | $t_{rfo}$ | — | — | 8 | ns |

1. $0.5t_{bus}$ added due to internal synchronization delay

# Appendix M Ordering Information

The following figure provides an ordering partnumber example for the devices covered by this data book. There are two options when ordering a device. Customers must choose between ordering either the mask-specific partnumber or the generic / mask-independent partnumber. Ordering the mask-specific partnumber enables the customer to specify which particular maskset they will receive whereas ordering the generic maskset means that FSL will ship the currently preferred maskset (which may change over time).

In either case, the marking on the device will always show the generic / mask-independent partnumber and the mask set number.

**NOTE**

**The mask identifier suffix and the Tape & Reel suffix are always both omitted from the partnumber which is actually marked on the device.**

For specific partnumbers to order, please contact your local sales office. The below figure illustrates the structure of a typical mask-specific ordering number for the MC9S12ZVHY-Family devices

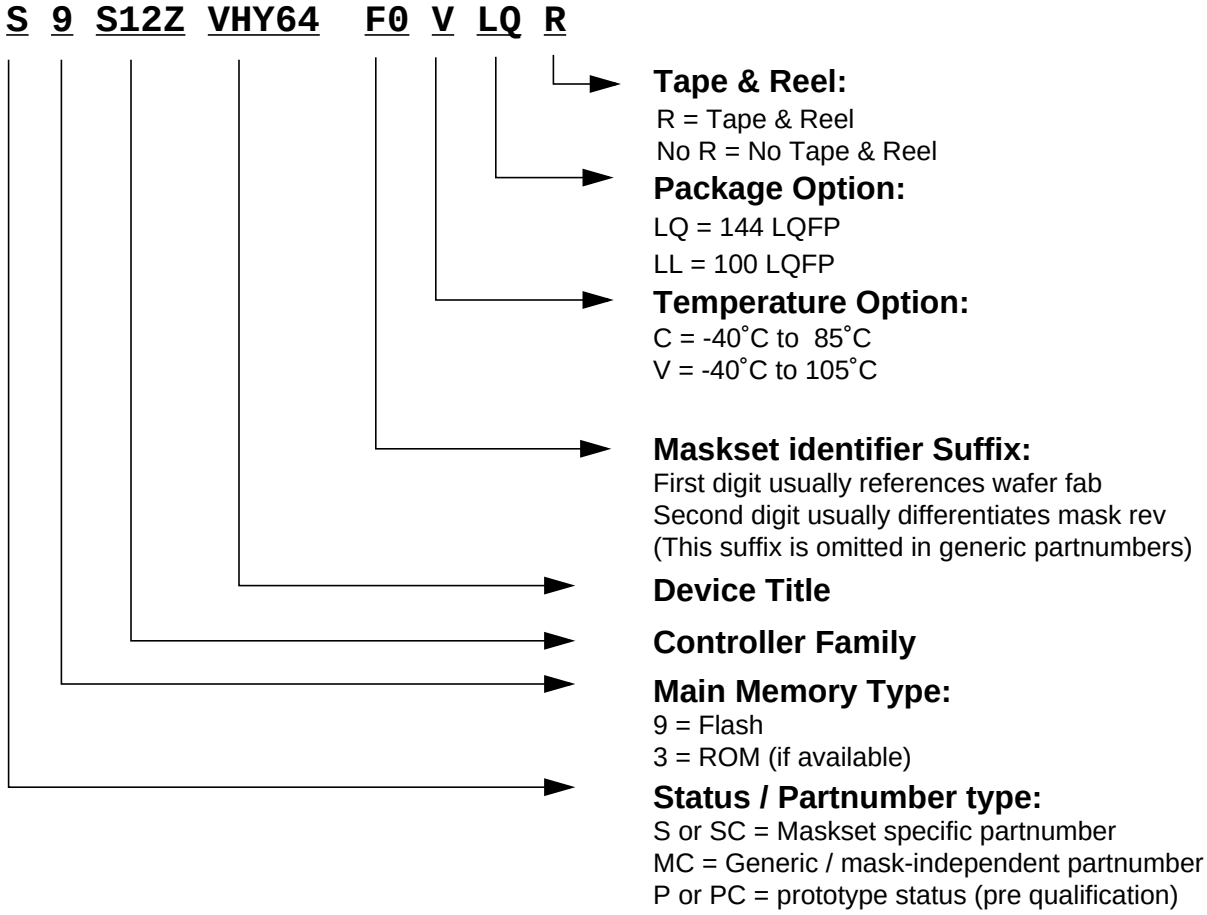
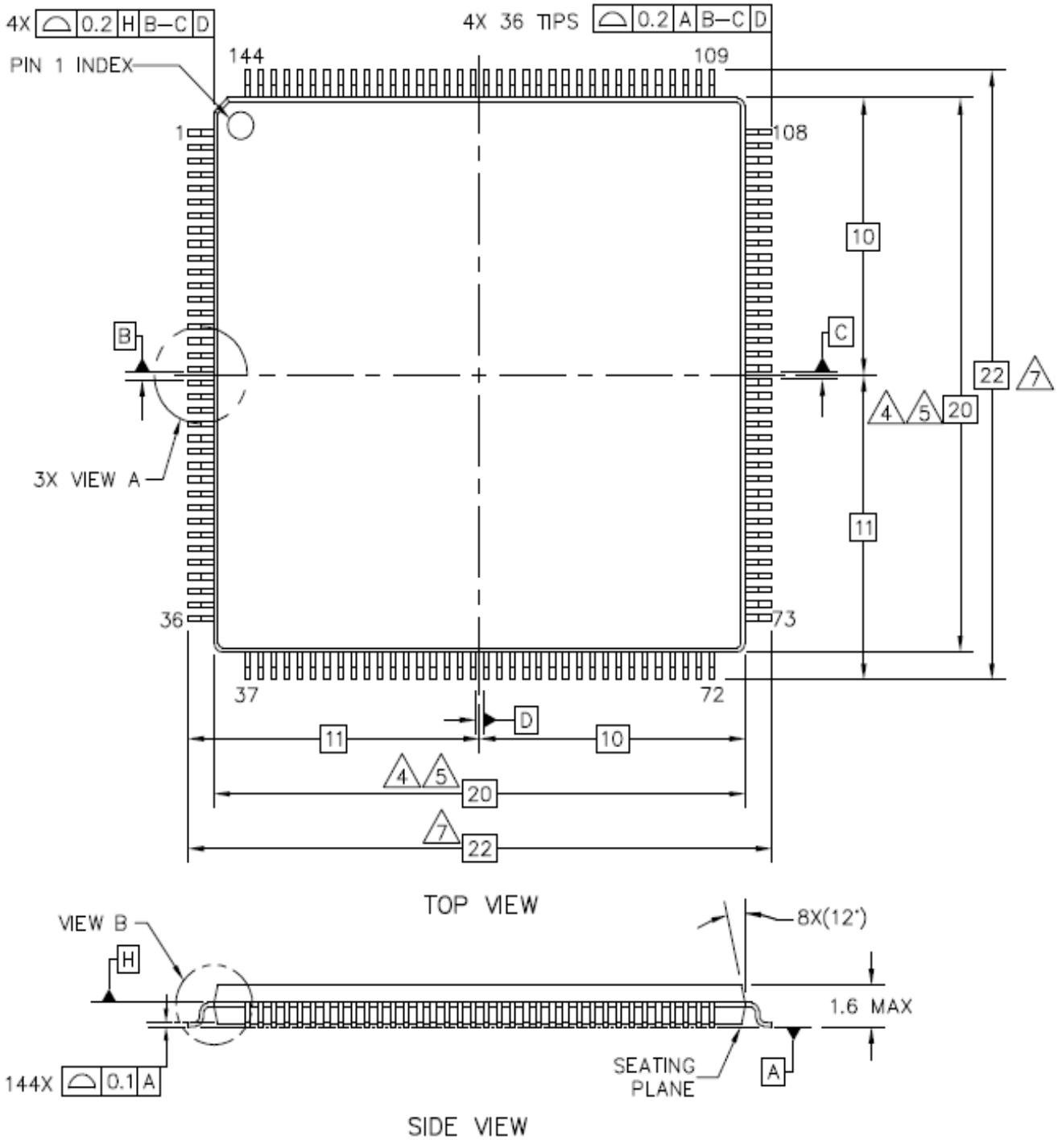


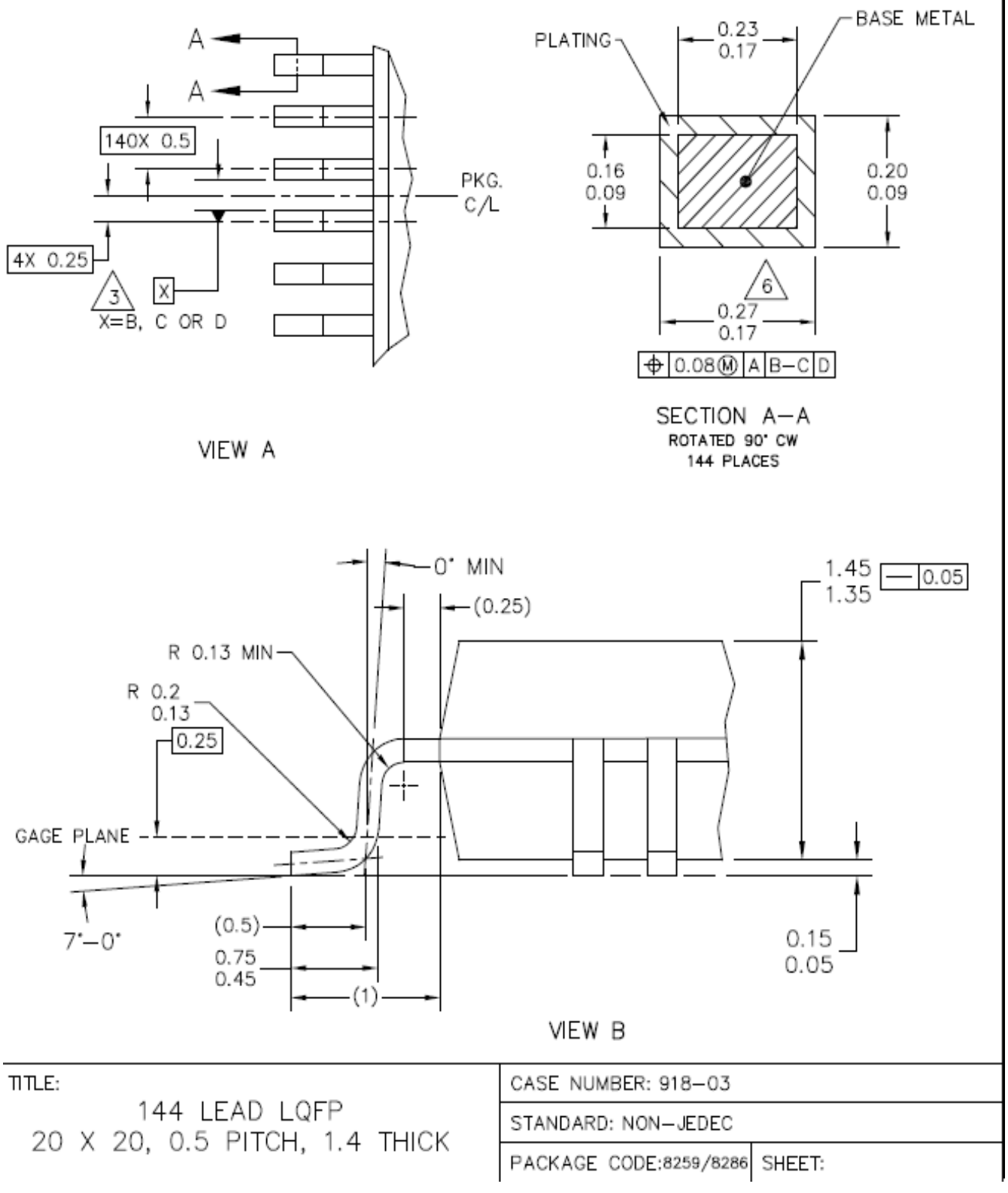

**Figure M-1. Part Number**

# Appendix N  Package Information

# N.1 144 LQFP




| TITLE: | CASE NUMBER: 918–03 | |
|---|---|---|
| 144 LEAD LQFP<br>20 X 20, 0.5 PITCH, 1.4 THICK | STANDARD: NON–JEDEC | |
| | PACKAGE CODE:8259/8286 | SHEET: |




| | |
|---|---|
| TITLE: 144 LEAD LQFP 20 X 20, 0.5 PITCH, 1.4 THICK | CASE NUMBER: 918–03 |
| | STANDARD: NON–JEDEC |
| | PACKAGE CODE:8259/8286 SHEET: |

NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS.

2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M–1994.

3. DATUMS B, C AND D TO BE DETERMINED AT DATUM PLANE H.

4. THE TOP PACKAGE BODY SIZE MAY BE SMALLER THAN THE BOTTOM PACKAGE SIZE BY A MAXIMUM OF 0.1 mm.

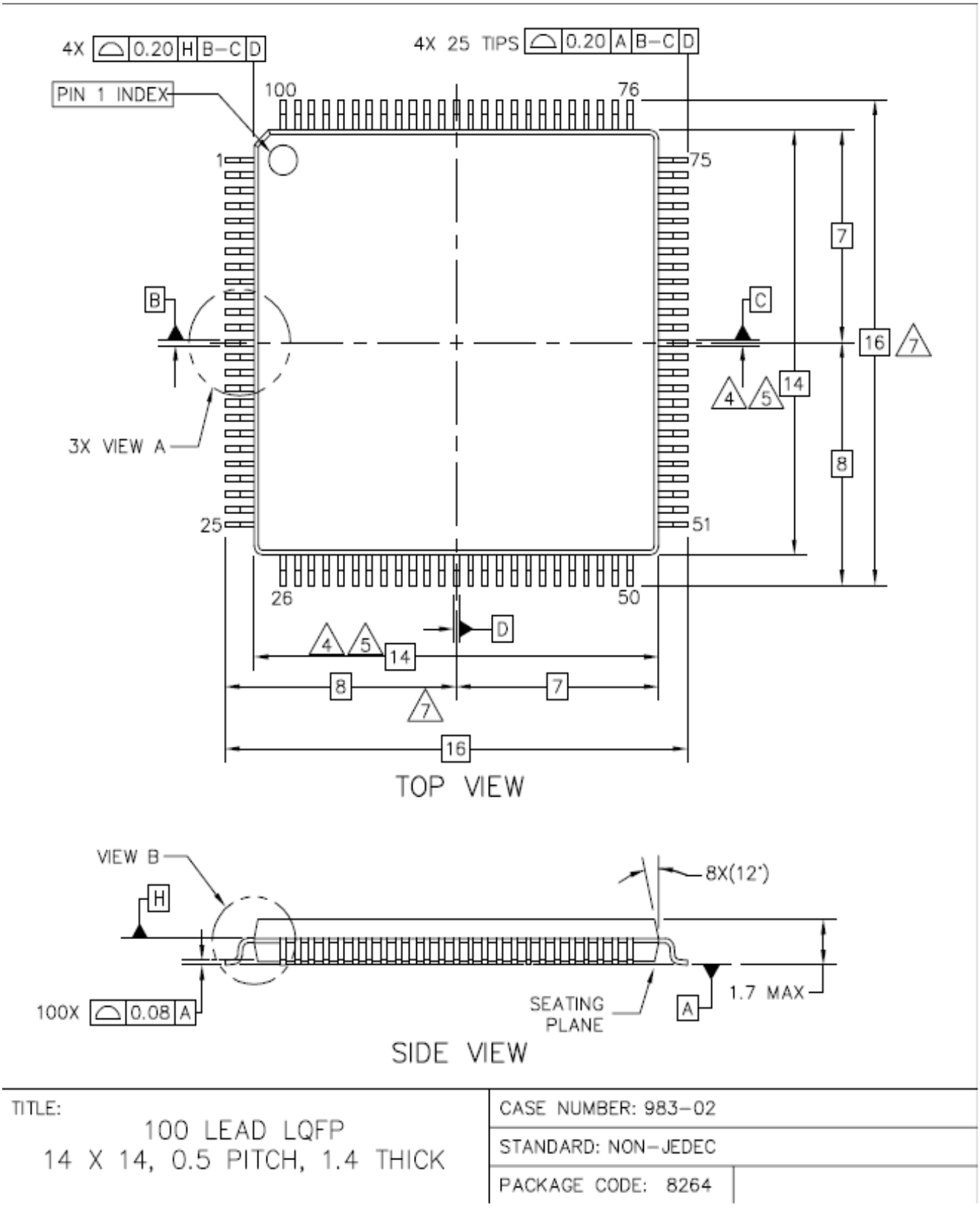
5. THIS DIMENSIONS DO NOT INCLUDE MOLD PROTRUSIONS. THE MAXIMUM ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. THIS DIMENSIONS ARE MAXIMUM BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.

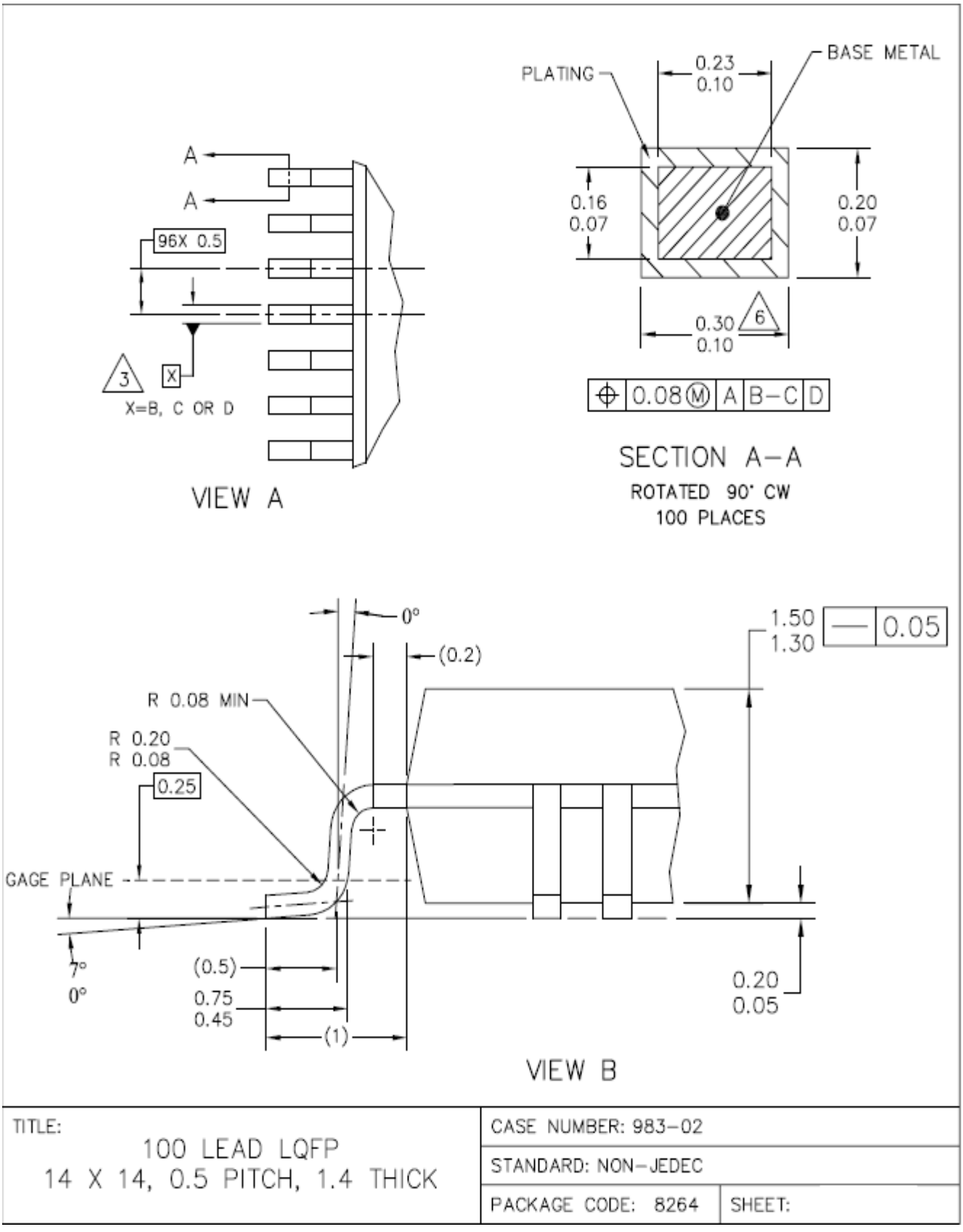
6. THIS DIMENSION DOES NOT INCLUDE DAM BAR PROTRUSION. PROTRUSIONS SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.35. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD SHALL BE 0.07 MM.

7. THIS DIMENSIONS ARE DETERMINED AT THE SEATING PLANE, DATUM A.

| TITLE: | CASE NUMBER: 918–03 | |
|---|---|---|
| 144 LEAD LQFP<br>20 X 20, 0.5 PITCH, 1.4 THICK | STANDARD: NON–JEDEC | |
| | PACKAGE CODE:8259/8286 | SHEET: |

## N.2 100 LQFP


4X 0.20 H B–C D
4X 25 TIPS 0.20 A B–C D
PIN 1 INDEX
100
76
1
75
25
51
26
50
3X VIEW A
TOP VIEW
VIEW B
8X(12°)
100X 0.08 A
SEATING PLANE
1.7 MAX
SIDE VIEW
TITLE:
100 LEAD LQFP
14 X 14, 0.5 PITCH, 1.4 THICK
CASE NUMBER: 983–02
STANDARD: NON–JEDEC
PACKAGE CODE: 8264


A
A
96X 0.5
3
X
X=B, C OR D
VIEW A
PLATING
BASE METAL
0.23
0.10
0.16
0.07
0.20
0.07
0.30
0.10
6
0.08 M A B-C D
SECTION A-A
ROTATED 90° CW
100 PLACES
0°
(0.2)
R 0.08 MIN
R 0.20
R 0.08
0.25
GAGE PLANE
7°
0°
(0.5)
0.75
0.45
(1)
1.50
1.30
0.05
0.20
0.05
VIEW B
TITLE:
100 LEAD LQFP
14 X 14, 0.5 PITCH, 1.4 THICK
CASE NUMBER: 983-02
STANDARD: NON-JEDEC
PACKAGE CODE: 8264
SHEET:

NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS.

2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M–1994.

3. DATUMS B, C AND D TO BE DETERMINED AT DATUM PLANE H.

4. THE TOP PACKAGE BODY SIZE MAY BE SMALLER THAN THE BOTTOM PACKAGE SIZE BY A MAXIMUM OF 0.1 MM.

5. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSIONS. THE MAXIMUM ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. THE DIMENSIONS ARE MAXIMUM BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.

6. DIMENSION DOES NOT INCLUDE DAM BAR PROTRUSION. PROTRUSIONS SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.35. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD SHALL BE 0.07 MM.

7. DIMENSIONS ARE DETERMINED AT THE SEATING PLANE, DATUM A.

| TITLE: | CASE NUMBER: 983–02 |
|---|---|
| 100 LEAD LQFP<br>14 X 14, 0.5 PITCH, 1.4 THICK | STANDARD: NON–JEDEC |
| | PACKAGE CODE: 8264 |

# Appendix O Detailed Register Address Map

## Revision History

| Version Number | Revision Date | Description of Changes |
|---|---|---|
| Rev 0.01 | 08-Mar-2012 | Initial Version |
| Rev 0.02 | 19-Mar-2012 | Update for CPMU register change |
| Rev 0.03 | 19-April-2012 | Update for RTC register change |
| Rev 0.04 | 25-May-2012 | Update for FTMRZ register change |
| Rev 0.05 | 17-Oct-2012 | Update ftmrz FPSTAT for WSTATACK missing<br>Correct PARTID description |
| Rev 0.06 | 21-Nov-2012 | Update SCI from V5 to V6 for 1N39G |

## O.1 Detailed Register Map

The following tables show the detailed register map of the MC9S12ZVHY-Family.

**NOTE**

This is a summary of all register bits implemented on MC9S12ZVHY-Family devices. Each member of the MC9S12ZVHY-Family implements the subset of registers, which is associated with its feature set (see Table 1-2).

### 0x0000–0x0003 PARTID registers

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | PARTID0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | W | | | | | | | | |
| 0x0001 | PARTID1 | R | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| | | W | | | | | | | | |
| 0x0002 | PARTID2 | R | Revision Dependent | | | | | | | |
| | | W | | | | | | | | |
| 0x0003 | PARTID3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

### 0x0004–0x000F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0004-0x000F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0010–0x001F Interrupt Control module(INT)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0010 | IVBR | R | IVB_ADDR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0011 | | R | IVB_ADDR[7:1] | | | | | | | 0 |
| | | W | | | | | | | | |
| 0x0012-0x0015 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0016 | INT_XGPRIO | R | 0 | 0 | 0 | 0 | 0 | XILVL[2:0] | | |
| | | W | | | | | | | | |
| 0x0017 | INT_CFADDR | R | 0 | INT_CFADDR[6:3] | | | | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0018 | INT_CFDATA0 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x0019 | INT_CFDATA1 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x001A | INT_CFDATA2 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x001B | INT_CFDATA3 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x001C | INT_CFDATA4 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x001D | INT_CFDATA5 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x001E | INT_CFDATA6 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |
| 0x001F | INT_CFDATA7 | R | RQST | 0 | 0 | 0 | 0 | PRIOLVL[2:0] | | |
| | | W | | | | | | | | |

## 0x0020–0x006F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0020-0x006F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0070–0x00FF Memory Map Control Module(MMC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0070 | MODE | R | MODC | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0071-0x007F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0080 | MMCECH | R | ITR[3:0] | | | | TGT[2:0] | | | |
| | | W | | | | | | | | |
| 0x0081 | MMCECL | R | ACC[3:0] | | | | ERR[3:0] | | | |
| | | W | | | | | | | | |
| 0x0082 | MMCCCRH | R | CPUU | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0083 | MMCCCRL | R | 0 | CPUX | 0 | CPUI | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0084 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0085 | MMCPCH | R | CPUPC[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0086 | MMCPCM | R | CPUPC[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0087 | MMCPCL | R | CPUPC[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0088-0x00FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0100–0x017F Debug Module (DBG)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0100 | DBGC1 | R | ARM | 0 | reserved | BDMBP | BRKCPU | reserved | EEVE | |
| | | W | | TRIG | | | | | | |
| 0x0101 | DBGC2 | R | 0 | 0 | 0 | 0 | CDCM | | ABCM | |
| | | W | | | | | | | | |
| 0x0102 | DBGTCRH | R | reserved | TSOURCE | TRANGE | | TRCMOD | | TALIGN | |
| | | W | | | | | | | | |
| 0x0103 | DBGTCRL | R | 0 | 0 | 0 | 0 | DSTAMP | PDOE | PROFILE | STAMP |
| | | W | | | | | | | | |
| 0x0104 | DBGTB | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |

## 0x0100–0x017F Debug Module (DBG)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0105 | DBGTB | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0106 | DBGCNT | R | 0 | CNT | | | | | | |
| | | W | | | | | | | | |
| 0x0107 | DBGSCR1 | R | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x0108 | DBGSCR2 | R | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x0109 | DBGSCR3 | R | C3SC1 | C3SC0 | C2SC1 | C2SC0 | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x010A | DBGEFR | R | PTBOVF | TRIGF | 0 | EEVF | ME3 | ME2 | ME1 | ME0 |
| | | W | | | | | | | | |
| 0x010B | DBGSR | R | TBF | 0 | 0 | PTACT | 0 | SSF2 | SSF1 | SSF0 |
| | | W | | | | | | | | |
| 0x010C-0x010F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0110 | DBGACTL | R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| | | W | | | | | | | | |
| 0x0111-0x0114 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0115 | DBGAAH | R | DBGAA[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0116 | DBGAAM | R | DBGAA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0117 | DBGAAL | R | DBGAA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0118 | DBGAD0 | R | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| | | W | | | | | | | | |
| 0x0119 | DBGAD1 | R | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |
| 0x011A | DBGAD2 | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x011B | DBGAD3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x011C | DBGADM0 | R | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| | | W | | | | | | | | |
| 0x011D | DBGADM1 | R | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |

## 0x0100–0x017F Debug Module (DBG)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x011E | DBGADM2 | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x011F | DBGADM3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0120 | DBGBCTL | R | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| | | W | | | | | | | | |
| 0x0121-0x0124 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0125 | DBGBAH | R | DBGBA[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0126 | DBGBAM | R | DBGBA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0127 | DBGBAL | R | DBGBA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0128-0x012F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0130 | DBGCCTL | R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| | | W | | | | | | | | |
| 0x0131-0x0134 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0135 | DBGCAH | R | DBGCA[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0136 | DBGCAM | R | DBGCA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0137 | DBGCAL | R | DBGCA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0138 | DBGCD0 | R | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| | | W | | | | | | | | |
| 0x0139 | DBGCD1 | R | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |
| 0x013A | DBGCD2 | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x013B | DBGCD3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x013C | DBGCDM0 | R | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| | | W | | | | | | | | |
| 0x013D | DBGCDM1 | R | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |

## 0x0100–0x017F Debug Module (DBG)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x013E | DBGCDM2 | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x013F | DBGCDM3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0140 | DBGDCTL | R | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| | | W | | | | | | | | |
| 0x0141-0x0144 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0145 | DBGDAH | R | DBGDA[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0146 | DBGDAM | R | DBGDA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0147 | DBGDAL | R | DBGDA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0148-0x017F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0180–0x01FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0180-0x01FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0200 | MODRR0 | R | 0 | 0 | 0 | 0 | C0RR | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0201 | MODRR1 | R | 0 | 0 | 0 | 0 | PWM6RR | PWM4RR | PWM2RR | PWM0RR |
| | | W | | | | | | | | |
| 0x0202 | MODRR2 | R | 0 | 0 | SCI1RR | IIC0RR | 0 | 0 | T1IC0RR1 | T1IC0RR0 |
| | | W | | | | | | | | |
| 0x02030x0204–0x0207 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0208 | ECLKCTL | R | NECLK | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0209 | IRQCR | R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x020A | PIMMISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CALCLKEN |
| | | W | | | | | | | | |
| 0x020B–0x020D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x020E | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x020F | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x0210–0x021F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0220 | PTA | R | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | W | | | | | | | | |
| 0x0221 | PTB | R | 0 | 0 | 0 | 0 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | W | | | | | | | | |
| 0x0222 | PTIA | R | PTIA7 | PTIA6 | PTIA5 | PTIA4 | PTIA3 | PTIA2 | PTIA1 | PTIA0 |
| | | W | | | | | | | | |
| 0x0223 | PTIB | R | 0 | 0 | 0 | 0 | PTIB3 | PTIB2 | PTIB1 | PTIB0 |
| | | W | | | | | | | | |
| 0x0224 | DDRA | R | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | W | | | | | | | | |
| 0x0225 | DDRB | R | 0 | 0 | 0 | 0 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | W | | | | | | | | |
| 0x0226 | PERA | R | PERA7 | PERA6 | PERA5 | PERA4 | PERA3 | PERA2 | PERA1 | PERA0 |
| | | W | | | | | | | | |
| 0x0227 | PERB | R | 0 | 0 | 0 | 0 | PERB3 | PERB2 | PERB1 | PERB0 |
| | | W | | | | | | | | |
| 0x0228 | PPSA | R | PPSA7 | PPSA6 | PPSA5 | PPSA4 | PPSA3 | PPSA2 | PPSA1 | PPSA0 |
| | | W | | | | | | | | |
| 0x0229 | PPSB | R | 0 | 0 | 0 | 0 | PPSB3 | PPSB2 | PPSB1 | PPSB0 |
| | | W | | | | | | | | |
| 0x022A–0x023D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x023E | WOMA | R | 0 | 0 | 0 | 0 | WOMA3 | WOMA2 | 0 | 0 |
| | | W | | | | | | | | |
| 0x023F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x0240 | PTC | R | PTC7 | PTC6 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | W | | | | | | | | |
| 0x0241 | PTD | R | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| | | W | | | | | | | | |
| 0x0242 | PTIC | R | PTIC7 | PTIC6 | PTIC5 | PTIC4 | PTIC3 | PTIC2 | PTIC1 | PTIC0 |
| | | W | | | | | | | | |
| 0x0243 | PTID | R | PTID7 | PTID6 | PTID5 | PTID4 | PTID3 | PTID2 | PTID1 | PTID0 |
| | | W | | | | | | | | |
| 0x0244 | DDRC | R | DDRC7 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | W | | | | | | | | |
| 0x0245 | DDRD | R | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| | | W | | | | | | | | |
| 0x0246 | PERC | R | PERC7 | PERC6 | PERC5 | PERC4 | PERC3 | PERC2 | PERC1 | PERC0 |
| | | W | | | | | | | | |
| 0x0247 | PERD | R | PERD7 | PERD6 | PERD5 | PERD4 | PERD3 | PERD2 | PERD1 | PERD0 |
| | | W | | | | | | | | |
| 0x0248 | PPSC | R | PPSC7 | PPSC6 | PPSC5 | PPSC4 | PPSC3 | PPSC2 | PPSC1 | PPSC0 |
| | | W | | | | | | | | |
| 0x0249 | PPSD | R | PPSD7 | PPSD6 | PPSD5 | PPSD4 | PPSD3 | PPSD2 | PPSD1 | PPSD0 |
| | | W | | | | | | | | |
| 0x024A–0x025F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0260 | PTE | R | 0 | 0 | 0 | 0 | PTE3 | PTE2 | PTE1 | PTE0 |
| | | W | | | | | | | | |
| 0x0261 | PTF | R | PTF7 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | | W | | | | | | | | |
| 0x0262 | PTIE | R | 0 | 0 | 0 | 0 | PTIE3 | PTIE2 | PTIE1 | PTIE0 |
| | | W | | | | | | | | |
| 0x0263 | PTIF | R | PTIF7 | PTIF6 | PTIF5 | PTIF4 | PTIF3 | PTIF2 | PTIF1 | PTIF0 |
| | | W | | | | | | | | |
| 0x0264 | DDRE | R | 0 | 0 | 0 | 0 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| | | W | | | | | | | | |
| 0x0265 | DDRF | R | DDRF7 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | W | | | | | | | | |
| 0x0266 | PERE | R | 0 | 0 | 0 | 0 | PERE3 | PERE2 | PERE1 | PERE0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0267 | PERF | R | PERF7 | PERF6 | PERF5 | PERF4 | PERF3 | PERF2 | PERF1 | PERF0 |
| | | W | | | | | | | | |
| 0x0268 | PPSE | R | 0 | 0 | 0 | 0 | PPSE3 | PPSE2 | PPSE1 | PPSE0 |
| | | W | | | | | | | | |
| 0x0269 | PPSF | R | PPSF7 | PPSF6 | PPSF5 | PPSF4 | PPSF3 | PPSF2 | PPSF1 | PPSF0 |
| | | W | | | | | | | | |
| 0x026A–0x027F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0280 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0281 | PTADL | R | PTADL7 | PTADL6 | PTADL5 | PTADL4 | PTADL3 | PTADL2 | PTADL1 | PTADL0 |
| | | W | | | | | | | | |
| 0x0282 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0283 | PTIADL | R | PTIADL7 | PTIADL6 | PTIADL5 | PTIADL4 | PTIADL3 | PTIADL2 | PTIADL1 | PTIADL0 |
| | | W | | | | | | | | |
| 0x0284 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0285 | DDRADL | R | DDRADL7 | DDRADL6 | DDRADL5 | DDRADL4 | DDRADL3 | DDRADL2 | DDRADL1 | DDRADL0 |
| | | W | | | | | | | | |
| 0x0286 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0287 | PERADL | R | PERADL7 | PERADL6 | PERADL5 | PERADL4 | PERADL3 | PERADL2 | PERADL1 | PERADL0 |
| | | W | | | | | | | | |
| 0x0288 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0289 | PPSADL | R | PPSADL7 | PPSADL6 | PPSADL5 | PPSADL4 | PPSADL3 | PPSADL2 | PPSADL1 | PPSADL0 |
| | | W | | | | | | | | |
| 0x028A–0x028B | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x028C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x028D | PIEADL | R | PIEADL7 | PIEADL6 | PIEADL5 | PIEADL4 | PIEADL3 | PIEADL2 | PIEADL1 | PIEADL0 |
| | | W | | | | | | | | |
| 0x028E | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x028F | PIFADL | R | PIFADL7 | PIFADL6 | PIFADL5 | PIFADL4 | PIFADL3 | PIFADL2 | PIFADL1 | PIFADL0 |
| | | W | | | | | | | | |
| 0x0290–0x0298 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0299 | DIENADL | R | DIENADL7 | DIENADL6 | DIENADL5 | DIENADL4 | DIENADL3 | DIENADL2 | DIENADL1 | DIENADL0 |
| | | W | | | | | | | | |
| 0x029A–0x02BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02C0 | PTT | R | PTT7 | PTT6 | PTT5 | PTT4 | PTT3 | PTT2 | PTT1 | PTT0 |
| | | W | | | | | | | | |
| 0x02C1 | PTIT | R | PTIT7 | PTIT6 | PTIT5 | PTIT4 | PTIT3 | PTIT2 | PTIT1 | PTIT0 |
| | | W | | | | | | | | |
| 0x02C2 | DDRT | R | DDRT7 | DDRT6 | DDRT5 | DDRT4 | DDRT3 | DDRT2 | DDRT1 | DDRT0 |
| | | W | | | | | | | | |
| 0x02C3 | PERT | R | PERT7 | PERT6 | PERT5 | PERT4 | PERT3 | PERT2 | PERT1 | PERT0 |
| | | W | | | | | | | | |
| 0x02C4 | PPST | R | PPST7 | PPST6 | PPST5 | PPST4 | PPST3 | PPST2 | PPST1 | PPST0 |
| | | W | | | | | | | | |
| 0x02C5 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02C6 | PIET | R | PIET7 | PIET6 | PIET5 | PIET4 | PIET3 | PIET2 | PIET1 | PIET0 |
| | | W | | | | | | | | |
| 0x02C7 | PIFT | R | PIFT7 | PIFT6 | PIFT5 | PIFT4 | PIFT3 | PIFT2 | PIFT1 | PIFT0 |
| | | W | | | | | | | | |
| 0x02C8 – 0x02CF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02D0 | PTS | R | PTS7 | PTS6 | PTS5 | PTS4 | PTS3 | PTS2 | PTS1 | PTS0 |
| | | W | | | | | | | | |
| 0x02D1 | PTIS | R | PTIS7 | PTIS6 | PTIS5 | PTIS4 | PTIS3 | PTIS2 | PTIS1 | PTIS0 |
| | | W | | | | | | | | |
| 0x02D2 | DDRS | R | DDRS7 | DDRS6 | DDRS5 | DDRS4 | DDRS3 | DDRS2 | DDRS1 | DDRS0 |
| | | W | | | | | | | | |
| 0x02D3 | PERS | R | PERS7 | PERS6 | PERS5 | PERS4 | PERS3 | PERS2 | PERS1 | PERS0 |
| | | W | | | | | | | | |
| 0x02D4 | PPSS | R | PPSS7 | PPSS6 | PPSS5 | PPSS4 | PPSS3 | PPSS2 | PPSS1 | PPSS0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02D5 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02D6 | PIES | R | PIES7 | PIES6 | PIES5 | PIES4 | PIES3 | PIES2 | PIES1 | PIES0 |
| | | W | | | | | | | | |
| 0x02D7 | PIFS | R | PIFS7 | PIFS6 | PIFS5 | PIFS4 | PIFS3 | PIFS2 | PIFS1 | PIFS0 |
| | | W | | | | | | | | |
| 0x02D8 – 0x02DE | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02DF | WOMS | R | WOMS7 | WOMS6 | WOMS5 | WOMS4 | WOMS3 | WOMS2 | WOMS1 | WOMS0 |
| | | W | | | | | | | | |
| 0x02E0– 0x02EF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x02F0 | PTP | R | PTP7 | PTP6 | PTP5 | PTP4 | PTP3 | PTP2 | PTP1 | PTP0 |
| | | W | | | | | | | | |
| 0x02F1 | PTIP | R | PTIP7 | PTIP6 | PTIP5 | PTIP4 | PTIP3 | PTIP2 | PTIP1 | PTIP0 |
| | | W | | | | | | | | |
| 0x02F2 | DDRP | R | DDRP7 | DDRP6 | DDRP5 | DDRP4 | DDRP3 | DDRP2 | DDRP1 | DDRP0 |
| | | W | | | | | | | | |
| 0x02F3 | PERP | R | PERP7 | PERP6 | PERP5 | PERP4 | PERP3 | PERP2 | PERP1 | PERP0 |
| | | W | | | | | | | | |
| 0x02F4 | PPSP | R | PPSP7 | PPSP6 | PPSP5 | PPSP4 | PPSP3 | PPSP2 | PPSP1 | PPSP0 |
| | | W | | | | | | | | |
| 0x02F5– 0x02FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0300 | PTH | R | PTH7 | PTH6 | PTH5 | PTH4 | PTH3 | PTH2 | PTH1 | PTH0 |
| | | W | | | | | | | | |
| 0x0301 | PTIH | R | PTIH7 | PTIH6 | PTIH5 | PTIH4 | PTIH3 | PTIH2 | PTIH1 | PTIH0 |
| | | W | | | | | | | | |
| 0x0302 | DDRH | R | DDRH7 | DDRH6 | DDRH5 | DDRH4 | DDRH3 | DDRH2 | DDRH1 | DDRH0 |
| | | W | | | | | | | | |
| 0x0303 | PERH | R | PERH7 | PERH6 | PERH5 | PERH4 | PERH3 | PERH2 | PERH1 | PERH0 |
| | | W | | | | | | | | |
| 0x0304 | PPSH | R | PPSH7 | PPSH6 | PPSH5 | PPSH4 | PPSH3 | PPSH2 | PPSH1 | PPSH0 |
| | | W | | | | | | | | |
| 0x0305– 0x030F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x0310 | PTJ | R | 0 | 0 | 0 | 0 | PTJ3 | PTJ2 | PTJ1 | PTJ0 |
| | | W | | | | | | | | |
| 0x0311 | PTIJ | R | 0 | 0 | 0 | 0 | PTIJ3 | PTIJ2 | PTIJ1 | PTIJ0 |
| | | W | | | | | | | | |
| 0x0312 | DDRJ | R | 0 | 0 | 0 | 0 | DDRJ3 | DDRJ2 | DDRJ1 | DDRJ0 |
| | | W | | | | | | | | |
| 0x0313 | PERJ | R | 0 | 0 | 0 | 0 | PERJ3 | PERJ2 | PERJ1 | PERJ0 |
| | | W | | | | | | | | |
| 0x0314 | PPSJ | R | 0 | 0 | 0 | 0 | PPSJ3 | PPSJ2 | PPSJ1 | PPSJ0 |
| | | W | | | | | | | | |
| 0x0315–0x031F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0320 | PTG | R | PTG7 | PTG6 | PTG5 | PTG4 | PTG3 | PTG2 | PTG1 | PTG0 |
| | | W | | | | | | | | |
| 0x0321 | PTIG | R | PTIG7 | PTIG6 | PTIG5 | PTIG4 | PTIG3 | PTIG2 | PTIG1 | PTIG0 |
| | | W | | | | | | | | |
| 0x0322 | DDRG | R | DDRG7 | DDRG6 | DDRG5 | DDRG4 | DDRG3 | DDRG2 | DDRG1 | DDRG0 |
| | | W | | | | | | | | |
| 0x0323 | PERG | R | PERG7 | PERG6 | PERG5 | PERG4 | PERG3 | PERG2 | PERG1 | PERG0 |
| | | W | | | | | | | | |
| 0x0324 | PPSG | R | PPSG7 | PPSG6 | PPSG5 | PPSG4 | PPSG3 | PPSG2 | PPSG1 | PPSG0 |
| | | W | | | | | | | | |
| 0x0325–0x032F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0350 | PTU | R | PTU7 | PTU6 | PTU5 | PTU4 | PTU3 | PTU2 | PTU1 | PTU0 |
| | | W | | | | | | | | |
| 0x0351 | PTIU | R | PTIU7 | PTIU6 | PTIU5 | PTIU4 | PTIU3 | PTIU2 | PTIU1 | PTIU0 |
| | | W | | | | | | | | |
| 0x0352 | DDRU | R | DDRU7 | DDRU6 | DDRU5 | DDRU4 | DDRU3 | DDRU2 | DDRU1 | DDRU0 |
| | | W | | | | | | | | |
| 0x0353 | PERU | R | PERU7 | PERU6 | PERU5 | PERU4 | PERU3 | PERU2 | PERU1 | PERU0 |
| | | W | | | | | | | | |
| 0x0354 | PPSU | R | PPSU7 | PPSU6 | PPSU5 | PPSU4 | PPSU3 | PPSU2 | PPSU1 | PPSU0 |
| | | W | | | | | | | | |
| 0x0355–0x035D | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0200–0x037F Port Integration Module(PIM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x035E | SRRU | R | SRRU7 | SRRU6 | SRRU5 | SRRU4 | SRRU3 | SRRU2 | SRRU1 | SRRU0 |
| | | W | | | | | | | | |
| 0x035F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0360–0x037F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0380–0x039F Flash Module(FTMRZ)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0380 | FCLKDIV | R | FDIVLD | FDIVLCK | FDIV5 | FDIV4 | FDIV3 | FDIV2 | FDIV1 | FDIV0 |
| | | W | | | | | | | | |
| 0x0381 | FSEC | R | KEYEN1 | KEYEN0 | RNV5 | RNV4 | RNV3 | RNV2 | SEC1 | SEC0 |
| | | W | | | | | | | | |
| 0x0382 | FCCOBIX | R | 0 | 0 | 0 | 0 | 0 | CCOBIX2 | CCOBIX1 | CCOBIX0 |
| | | W | | | | | | | | |
| 0x0383 | FPSTAT | R | FPOVRD | 0 | 0 | 0 | 0 | 0 | 0 | WSTATACK |
| | | W | | | | | | | | |
| 0x0384 | FCNFG | R | CCIE | 0 | ERSAREQ | IGNSF | WSTAT[1:0] | | FDFD | FSFD |
| | | W | | | | | | | | |
| 0x0385 | FERCNFG | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SFDIE |
| | | W | | | | | | | | |
| 0x0386 | FSTAT | R | CCIF | 0 | ACCERR | FPVIOL | MGBUSY | RSVD | MGSTAT1 | MGSTAT0 |
| | | W | | | | | | | | |
| 0x0387 | FERSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | DFDIF | SFDIF |
| | | W | | | | | | | | |
| 0x0388 | FPROT | R | FPOPEN | RNV6 | FPHDIS | FPHS1 | FPHS0 | FPLDIS | FPLS1 | FPLS0 |
| | | W | | | | | | | | |
| 0x0389 | DFPROT | R | DPOPEN | DPS6 | DPS5 | DPS4 | DPS3 | DPS2 | DPS1 | DPS0 |
| | | W | | | | | | | | |
| 0x038A | FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | | W | | | | | | | | |
| 0x038B | FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x038C | FCCOB0HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |

## 0x0380–0x039F Flash Module(FTMRZ)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x038D | FCCOB0LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |
| 0x038E | FCCOB1HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |
| 0x038F | FCCOB1LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |
| 0x0390 | FCCOB2HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |
| 0x0391 | FCCOB2LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |
| 0x0392 | FCCOB3HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |
| 0x0393 | FCCOB3LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |
| 0x0394 | FCCOB4HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |
| 0x0395 | FCCOB4LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |
| 0x0396 | FCCOB5HI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | | W | | | | | | | | |
| 0x0397 | FCCOB5LO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | | W | | | | | | | | |
| 0x0398- 0x039F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x03A0–0x03BF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03A0- 0x03BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x03C0–0x03CF SRAM ECC Generator(SRAM_ECC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03C0 | ECCSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RDY |
| | | W | | | | | | | | |
| 0x03C1 | ECCIE | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIE |
| | | W | | | | | | | | |

## 0x03C0–0x03CF SRAM ECC Generator(SRAM_ECC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03C2 | ECCIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SBEEIF |
| | | W | | | | | | | | |
| 0x03C3-0x03C6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03C7 | ECCDPTRH | R | DPTR[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x03C8 | ECCDPTRM | R | DPTR[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03C9 | ECCDPTRL | R | DPTR[7:1] | | | | | | | 0 |
| | | W | | | | | | | | |
| 0x03CA-0x3CB | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x03CC | ECCDDH | R | DDATA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x03CD | ECCDDL | R | DDATA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x03CE | ECCDE | R | 0 | 0 | DECC[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x03CF | ECCDCMD | R | ECCDRR | 0 | 0 | 0 | 0 | 0 | ECCDW | ECCDR |
| | | W | | | | | | | | |

## 0x03D0–0x03FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03D0-0x03FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0400–0x042F Timer Module (TIM1)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0400 | TIM1TIOS | R | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | | W | | | | | | | | |
| 0x0401 | TIM1CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x0402 | TIM1OC7M | R | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| | | W | | | | | | | | |

## 0x0400–0x042F Timer Module (TIM1)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0403 | TIM1OC7D | R<br>W | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| 0x0404 | TIM1TCNTH | R<br>W | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| 0x0405 | TIM1TCNTL | R<br>W | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| 0x0406 | TIM1TSCR1 | R<br>W | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| 0x0407 | TIM1TTOV | R<br>W | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| 0x0408 | TIM1TCTL1 | R<br>W | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| 0x0409 | TIM1TCTL2 | R<br>W | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| 0x040A | TIM1TCTL3 | R<br>W | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| 0x040B | TIM1TCTL4 | R<br>W | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| 0x040C | TIM1TIE | R<br>W | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| 0x040D | TIM1TSCR2 | R<br>W | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| 0x040E | TIM1TFLG1 | R<br>W | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| 0x040F | TIM1TFLG2 | R<br>W | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0410<br>–<br>0x041F | TIM1TCxH –<br>TIM1TCxL | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x0420 | TIM1PACTL | R<br>W | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| 0x0421 | TIM1PAFLG | R<br>W | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| 0x0422 | TIM1PACNTH | R<br>W | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |

## 0x0400–0x042F Timer Module (TIM1)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0423 | TIM1PACNTL | R | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| | | W | | | | | | | | |
| 0x0424-0x042B | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |
| 0x042C | TIM1OCPD | R | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| | | W | | | | | | | | |
| 0x042D | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |
| 0x042E | TIM1PTPSR | R | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| | | W | | | | | | | | |
| 0x042F | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |

## 0x0430–0x047F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0430-0x047F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0480–0x04A7 Pulse-Width-Modulator (PWM)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0480 | PWME | R | PWME7 | PWME6 | PWME5 | PWME4 | PWME3 | PWME2 | PWME1 | PWME0 |
| | | W | | | | | | | | |
| 0x0481 | PWMPOL | R | PPOL7 | PPOL6 | PPOL5 | PPOL4 | PPOL3 | PPOL2 | PPOL1 | PPOL0 |
| | | W | | | | | | | | |
| 0x0482 | PWMCLK | R | PCLK7 | PCLKL6 | PCLK5 | PCLK4 | PCLK3 | PCLK2 | PCLK1 | PCLK0 |
| | | W | | | | | | | | |
| 0x0483 | PWMPRCLK | R | 0 | PCKB2 | PCKB1 | PCKB0 | 0 | PCKA2 | PCKA1 | PCKA0 |
| | | W | | | | | | | | |
| 0x0484 | PWMCAE | R | CAE7 | CAE6 | CAE5 | CAE4 | CAE3 | CAE2 | CAE1 | CAE0 |
| | | W | | | | | | | | |
| 0x0485 | PWMCTL | R | CON67 | CON45 | CON23 | CON01 | PSWAI | PFRZ | 0 | 0 |
| | | W | | | | | | | | |
| 0x0486 | PWMCLKAB | R | PCLKAB7 | PCLKAB6 | PCLKAB5 | PCLKAB4 | PCLKAB3 | PCLKAB2 | PCLKAB1 | PCLKAB0 |
| | | W | | | | | | | | |
| 0x0487 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0480–0x04A7 Pulse-Width-Modulator (PWM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0488 | PWMSCLA | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0489 | PWMSCLB | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x048A - 0x048B | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x048C | PWMCNT0 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x048D | PWMCNT1 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x048E | PWMCNT2 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x048F | PWMCNT3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0490 | PWMCNT4 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0491 | PWMCNT5 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0492 | PWMCNT6 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0493 | PWMCNT7 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0494 | PWMPER0 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0495 | PWMPER1 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0496 | PWMPER2 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0497 | PWMPER3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0498 | PWMPER4 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0499 | PWMPER5 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x049A | PWMPER6 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |

## 0x0480–0x04A7 Pulse-Width-Modulator (PWM)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x049B | PWMPER7 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x049C | PWMDTY0 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x049D | PWMDTY1 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x049E | PWMDTY2 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x049F | PWMDTY3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x04A0 | PWMDTY4 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x04A1 | PWMDTY5 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x04A2 | PWMDTY6 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x04A3 | PWMDTY7 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x04A4-<br>0x04A7 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x04A8–0x05BF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x04A8-<br>0x05BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x05C0–0x05EF Timer Module (TIM0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x05C0 | TIM0TIOS | R | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | | W | | | | | | | | |
| 0x05C1 | TIM0CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | FOC7 | FOC6 | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x05C2 | TIM0OC7M | R | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| | | W | | | | | | | | |
| 0x05C3 | TIM0OC7D | R | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| | | W | | | | | | | | |

## 0x05C0–0x05EF Timer Module (TIM0)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x05C4 | TIM0TCNTH | R | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| | | W | | | | | | | | |
| 0x05C5 | TIM0TCNTL | R | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| | | W | | | | | | | | |
| 0x05C6 | TIM0TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x05C7 | TIM0TTOV | R | TOV7 | TOV6 | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | | W | | | | | | | | |
| 0x05C8 | TIM0TCTL1 | R | OM7 | OL7 | OM6 | OL6 | OM5 | OL5 | OM4 | OL4 |
| | | W | | | | | | | | |
| 0x05C9 | TIM0TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | | W | | | | | | | | |
| 0x05CA | TIM0TCTL3 | R | EDG7B | EDG7A | EDG6B | EDG6A | EDG5B | EDG5A | EDG4B | EDG4A |
| | | W | | | | | | | | |
| 0x05CB | TIM0TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | | W | | | | | | | | |
| 0x05CC | TIM0TIE | R | C7I | C6I | C5I | C4I | C3I | C2I | C1I | C0I |
| | | W | | | | | | | | |
| 0x05CD | TIM0TSCR2 | R | TOI | 0 | 0 | 0 | TCRE | PR2 | PR1 | PR0 |
| | | W | | | | | | | | |
| 0x05CE | TIM0TFLG1 | R | C7F | C6F | C5F | C4F | C3F | C2F | C1F | C0F |
| | | W | | | | | | | | |
| 0x05CF | TIM0TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x05D0 – 0x05DF | TIM0TCxH – TIM0TCxL | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| | | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x05E0 | TIM0PACTL | R | 0 | PAEN | PAMOD | PEDGE | CLK1 | CLK0 | PAOVI | PAI |
| | | W | | | | | | | | |
| 0x05E1 | TIM0PAFLG | R | 0 | 0 | 0 | 0 | 0 | 0 | PAOVF | PAIF |
| | | W | | | | | | | | |
| 0x05E2 | TIM0PACNTH | R | PACNT15 | PACNT14 | PACNT13 | PACNT12 | PACNT11 | PACNT10 | PACNT9 | PACNT8 |
| | | W | | | | | | | | |
| 0x05E3 | TIM0PACNTL | R | PACNT7 | PACNT6 | PACNT5 | PACNT4 | PACNT3 | PACNT2 | PACNT1 | PACNT0 |
| | | W | | | | | | | | |
| 0x05E4- 0x05EB | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |

## 0x05C0–0x05EF Timer Module (TIM0)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x05EC | TIM0OCPD | R | OCPD7 | OCPD6 | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| | | W | | | | | | | | |
| 0x05ED | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |
| 0x05EE | TIM0PTPSR | R | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| | | W | | | | | | | | |
| 0x05EF | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |

## 0x05F0–0x05FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x05F0-0x05FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0600–0x063F Analog to Digital Converter (ADC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0600 | ADC0CTL_0 | R | ADC_EN | ADC_SR | FRZ_MOD | SWAI | ACC_CFG[1:0] | | STR_SEQA | MOD_CFG |
| | | W | | | | | | | | |
| 0x0601 | ADC0CTL_1 | R | CSL_BMOD | RVL_BMOD | SMOD_ACC | AUT_RSTA | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0602 | ADC0STS | R | CSL_SEL | RVL_SEL | DBECC_ERR | Reserved | READY | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0603 | ADC0TIM | R | 0 | PRS[6:0] | | | | | | |
| | | W | | | | | | | | |
| 0x0604 | ADC0FMT | R | DJM | 0 | 0 | 0 | 0 | SRES[2:0] | | |
| | | W | | | | | | | | |
| 0x0605 | ADC0FLWCTL | R | SEQA | TRIG | RSTA | LDOK | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0606 | ADC0EIE | R | IA_EIE | CMD_EIE | EOL_EIE | Reserved | TRIG_EIE | RSTAR_EIE | LDOK_EIE | 0 |
| | | W | | | | | | | | |
| 0x0607 | ADC0IE | R | SEQAD_IE | CONIF_OIE | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0608 | ADC0EiF | R | IA_EIF | CMD_EIF | EOL_EIF | Reserved | TRIG_EIF | RSTAR_EIF | LDOK_EIF | 0 |
| | | W | | | | | | | | |
| 0x0609 | ADC0IF | R | SEQAD_IF | CONIF_OIF | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0600–0x063F Analog to Digital Converter (ADC)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x060A | ADC0CONIE_0 | R | CON_IE[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x060B | ADC0CONIE_1 | R | CON_IE[7:1] | | | | | | | EOL_IE |
| | | W | | | | | | | | |
| 0x060C | ADC0CONIF_0 | R | CON_IF[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x060D | ADC0CONIF_1 | R | CON_IF[7:1] | | | | | | | EOL_IF |
| | | W | | | | | | | | |
| 0x060E | ADC0IMDRI_0 | R | CSL_IMD | RVL_IMD | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x060F | ADC0IMDRI_1 | R | 0 | RIDX_IMD | | | | | | |
| | | W | | | | | | | | |
| 0x0610 | ADC0EOLRI | R | CSL_EOL | RVL_EOL | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0611 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0612 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0613 | Reserved | R | Reserved | Reserved | | | | | 0 | 0 |
| | | W | | | | | | | | |
| 0x0614 | ADC0CMD_0 | R | CMD_SEL | | 0 | 0 | INTFLG_SEL[3:0] | | | |
| | | W | | | | | | | | |
| 0x0615 | ADC0CMD_1 | R | VRH_SEL | VRL_SEL | CH_SEL[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x0616 | ADC0CMD_2 | R | SMP[4:0] | | | | | 0 | 0 | Reserved |
| | | W | | | | | | | | |
| 0x0617 | ADC0CMD_3 | R | Reserved | Reserved | Reserved | | | | | |
| | | W | | | | | | | | |
| 0x0618 | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x0619 | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x061A | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |
| 0x061B | Reserved | R | Reserved | | | | | | | |
| | | W | | | | | | | | |

## 0x0600–0x063F Analog to Digital Converter (ADC)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x061C | ADC0CIDX | R | 0 | 0 | CMD_IDX[5:0] | | | | |
| | | W | | | | | | | |
| 0x061D | ADC0CBP_0 | R | CMD_PTR[23:16] | | | | | | |
| | | W | | | | | | | |
| 0x061E | ADC0CBP_1 | R | CMD_PTR[15:8] | | | | | | |
| | | W | | | | | | | |
| 0x061F | ADC0CBP_2 | R | CMD_PTR[7:2] | | | | | 0 | 0 |
| | | W | | | | | | | |
| 0x0620 | ADC0RIDX | R | 0 | 0 | RES_IDX[5:0] | | | | |
| | | W | | | | | | | |
| 0x0621 | ADC0RBP_0 | R | 0 | 0 | 0 | 0 | RES_PTR[19:16] | | |
| | | W | | | | | | | |
| 0x0622 | ADC0RBP_1 | R | RES_PTR[15:8] | | | | | | |
| | | W | | | | | | | |
| 0x0623 | ADC0RBP_2 | R | RES_PTR[7:2] | | | | | 0 | 0 |
| | | W | | | | | | | |
| 0x0624 | ADC0CROFF0 | R | 0 | CMDRES_OFF0[6:0] | | | | | |
| | | W | | | | | | | |
| 0x0625 | ADC0CROFF1 | R | 0 | CMDRES_OFF1[6:0] | | | | | |
| | | W | | | | | | | |
| 0x0626 | Reserved | R | 0 | 0 | 0 | 0 | Reserved | | |
| | | W | | | | | | | |
| 0x0627 | Reserved | R | Reserved | | | | | | |
| | | W | | | | | | | |
| 0x0628 | Reserved | R | Reserved | | | | | 0 | 0 |
| | | W | | | | | | | |
| 0x0629 | Reserved | R | Reserved | 0 | Reserved | | | | |
| | | W | | | | | | | |

## 0x0640–0x06BF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0640-0x06BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x06C0–0x06DF Clock and Power Management (CPMU_UHV)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x06C0 | CPMU RESERVED00 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06C1 | CPMU RESERVED01 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06C2 | CPMU RESERVED02 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06C3 | CPMURFLG | R | 0 | PORF | LVRF | 0 | COPRF | 0 | OMRF | PMRF |
| | | W | | | | | | | | |
| 0x06C4 | CPMU SYNR | R | VCOFRQ[1:0] | | SYNDIV[5:0] | | | | | |
| | | W | | | | | | | | |
| 0x06C5 | CPMU REFDIV | R | REFFRQ[1:0] | | 0 | 0 | REFDIV[3:0] | | | |
| | | W | | | | | | | | |
| 0x06C6 | CPMU POSTDIV | R | 0 | 0 | 0 | POSTDIV[4:0] | | | | |
| | | W | | | | | | | | |
| 0x06C7 | CPMUIFLG | R | RTIF | 0 | 0 | LOCKIF | LOCK | 0 | OSCIF | UPOSC |
| | | W | | | | | | | | |
| 0x06C8 | CPMUINT | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | OSCIE | 0 |
| | | W | | | | | | | | |
| 0x06C9 | CPMUCLKS | R | PLLSEL | PSTP | CSAD | COP OSCSEL1 | PRE | PCE | RTI OSCSEL | COP OSCSEL0 |
| | | W | | | | | | | | |
| 0x06CA | CPMUPLL | R | 0 | 0 | FM1 | FM0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06CB | CPMURTI | R | RTDEC | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 |
| | | W | | | | | | | | |
| 0x06CC | CPMUCOP | R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 |
| | | W | | | WRTMASK | | | | | |
| 0x06CD | RESERVED CPMUTEST0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06CE | RESERVED CPMUTEST1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06CF | CPMU ARMCOP | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x06D0 | CPMU HTCTL | R | 0 | 0 | VSEL | 0 | HTE | HTDS | HTIE | HTIF |
| | | W | | | | | | | | |

## 0x06C0–0x06DF Clock and Power Management (CPMU_UHV)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x06D1 | CPMU LVCTL | R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| | | W | | | | | | | | |
| 0x06D2 | CPMU APICTL | R | APICLK | 0 | 0 | APIES | APIEA | APIFE | APIE | APIF |
| | | W | | | | | | | | |
| 0x06D3 | CPMUACLKTR | R | ACLKTR5 | ACLKTR4 | ACLKTR3 | ACLKTR2 | ACLKTR1 | ACLKTR0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06D4 | CPMUAPIRH | R | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| | | W | | | | | | | | |
| 0x06D5 | CPMUAPIRL | R | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| | | W | | | | | | | | |
| 0x06D6 | RESERVED CPMUTEST3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06D7 | CPMUHTTR | R | HTOE | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| | | W | | | | | | | | |
| 0x06D8 | CPMU IRCTRIMH | R | TCTRIM[4:0] | | | | | 0 | IRCTRIM[9:8] | |
| | | W | | | | | | | | |
| 0x06D9 | CPMU IRCTRIML | R | IRCTRIM[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x06DA | CPMUOSC | R | OSCE | 0 | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06DB | CPMUPROT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PROT |
| | | W | | | | | | | | |
| 0x06DC | RESERVED CPMUTEST2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x06DD | CPMU VREGCTL | R | 0 | 0 | 0 | 0 | 0 | 0 | EXTXON | INTXON |
| | | W | | | | | | | | |
| 0x06DE | CPMUOSC2 | R | 0 | 0 | 0 | 0 | 0 | 0 | OMRE | OSCMOD |
| | | W | | | | | | | | |
| 0x06DF | CPMU RESERVED1F | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x06E0–0x06EF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x06E0-<br>0x06EF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x06F0–0x06F7 Supply Voltage Sensor (BATS)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x06F0 | BATE | R | 0 | BVHS | BVLS[1:0] | | BSUAE | BSUSE | BSEAE | BSESE |
| | | W | | | | | | | | |
| 0x06F1 | BATSR | R | 0 | 0 | 0 | 0 | 0 | 0 | BVHC | BVLC |
| | | W | | | | | | | | |
| 0x06F2 | BATIE | R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIE | BVLIE |
| | | W | | | | | | | | |
| 0x06F3 | BATIF | R | 0 | 0 | 0 | 0 | 0 | 0 | BVHIF | BVLIF |
| | | W | | | | | | | | |
| 0x06F4-<br>0x06F5 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x06F8–0x06FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x06F8-<br>0x06FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0700–0x0707 Serial Communication Interface (SCI0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0700 | SCI0BDH[1] | R | SBR15 | SBR14 | SBR13 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x0701 | SCI0BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x0702 | SCI0CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x0700 | SCI0ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x0701 | SCI0ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |

## 0x0700–0x0707 Serial Communication Interface (SCI0)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0702 | SCI0ACR2[2] | R | IREN | TNP1 | TNP0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x0703 | SCI0CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |
| 0x0704 | SCI0SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x0705 | SCI0SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x0706 | SCI0DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0707 | SCI0DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1 These registers are accessible if the AMAP bit in the SCI0SR2 register is set to zero.
2 These registers are accessible if the AMAP bit in the SCI0SR2 register is set to one.

## 0x0708–0x070F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0708-0x070F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0710–0x0717 Serial Communication Interface (SCI1)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0710 | SCI1BDH[1] | R | SBR15 | SBR14 | SBR13 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| | | W | | | | | | | | |
| 0x0711 | SCI1BDL[1] | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| | | W | | | | | | | | |
| 0x0712 | SCI1CR1[1] | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| | | W | | | | | | | | |
| 0x0710 | SCI1ASR1[2] | R | RXEDGIF | 0 | 0 | 0 | 0 | BERRV | BERRIF | BKDIF |
| | | W | | | | | | | | |
| 0x0711 | SCI1ACR1[2] | R | RXEDGIE | 0 | 0 | 0 | 0 | 0 | BERRIE | BKDIE |
| | | W | | | | | | | | |
| 0x0712 | SCI1ACR2[2] | R | IREN | TNP1 | TNP0 | 0 | 0 | BERRM1 | BERRM0 | BKDFE |
| | | W | | | | | | | | |
| 0x0713 | SCI1CR2 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| | | W | | | | | | | | |

## 0x0710–0x0717 Serial Communication Interface (SCI1)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x0714 | SCI1SR1 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| | | W | | | | | | | | |
| 0x0715 | SCI1SR2 | R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| | | W | | | | | | | | |
| 0x0716 | SCI1DRH | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0717 | SCI1DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

1 These registers are accessible if the AMAP bit in the SCI1SR2 register is set to zero.
2 These registers are accessible if the AMAP bit in the SCI1SR2 register is set to one.

## 0x0718–0x077F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0718-0x077F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0780–0x0787 Serial Peripheral Interface (SPI0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0780 | SPI0CR1 | R | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| | | W | | | | | | | | |
| 0x0781 | SPI0CR2 | R | 0 | XFRW | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| | | W | | | | | | | | |
| 0x0782 | SPI0BR | R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| | | W | | | | | | | | |
| 0x0783 | SPI0SR | R | SPIF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0784 | SPI0DRH | R | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |
| | | | T15 | T14 | T13 | T12 | T11 | T10 | T9 | T8 |
| 0x0785 | SPI0DRL | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| 0x0786-0x0787 | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |

## 0x0788–0x07BF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0788-0x07BF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x07C0–0x07C7 Inter IC Bus (IIC0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x07C0 | IIC0IBAD | R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | 0 |
| | | W | | | | | | | | |
| 0x07C1 | IIC0IBFD | R | IBC7 | IBC6 | IBC5 | IBC4 | IBC3 | IBC2 | IBC1 | IBC0 |
| | | W | | | | | | | | |
| 0x07C2 | IIC0IBCR | R | IBEN | IBIE | MS/$\overline{SL}$ | TX/$\overline{RX}$ | TXAK | 0 | 0 | IBSWAI |
| | | W | | | | | | RSTA | | |
| 0x07C3 | IIC0IBSR | R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| | | W | | | | | | | | |
| 0x07C4 | IIC0IBDR | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D 0 |
| | | W | | | | | | | | |
| 0x07C5 | IIC0IBCR2 | R | GCEN | ADTYPE | 0 | 0 | 0 | ADR10 | ADR9 | ADR8 |
| | | W | | | | | | | | |
| 0x07C6 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x07C7 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x07C8–0x07FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x07C8-0x07FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0800–0x083F CAN Controller (MSCAN0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0800 | CAN0CTL0 | R | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| | | W | | | | | | | | |
| 0x0801 | CAN0CTL1 | R | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| | | W | | | | | | | | |
| 0x0802 | CAN0BTR0 | R | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | W | | | | | | | | |
| 0x0803 | CAN0BTR1 | R | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | W | | | | | | | | |
| 0x0804 | CAN0RFLG | R | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| | | W | | | | | | | | |
| 0x0805 | CAN0RIER | R | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| | | W | | | | | | | | |
| 0x0806 | CAN0TFLG | R | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| | | W | | | | | | | | |
| 0x0807 | CAN0TIER | R | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | W | | | | | | | | |
| 0x0808 | CAN0TARQ | R | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| | | W | | | | | | | | |
| 0x0809 | CAN0TAAK | R | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| | | W | | | | | | | | |
| 0x080A | CAN0TBSEL | R | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| | | W | | | | | | | | |
| 0x080B | CAN0IDAC | R | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | | W | | | | | | | | |
| 0x080C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x080D | CAN0MISC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| | | W | | | | | | | | |
| 0x080E | CAN0RXERR | R | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | W | | | | | | | | |
| 0x080F | CAN0TXERR | R | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | W | | | | | | | | |
| 0x0810-0x0813 | CAN0IDAR0–3 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |

## 0x0800–0x083F CAN Controller (MSCAN0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0814-0x0817 | CAN0IDMRx | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0818-0x081B | CAN0IDAR4–7 | R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | W | | | | | | | | |
| 0x081C-0x081F | CAN0IDMR4–7 | R | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | W | | | | | | | | |
| 0x0820-0x082F | CAN0RXFG | R | See 11.3.3” | | | | | | | |
| | | W | | | | | | | | |
| 0x0830-0x083F | CAN0TXFG | R | See 11.3.3” | | | | | | | |
| | | W | | | | | | | | |

## 0x0840–0x09FF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0840-0x09FF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0A00–0x0A1F Liquid Crystal Display (LCD)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A00 | LCDCR0 | R | LCDEN | 0 | 0 | LCLK1 | LCLK0 | BIAS | DUTY1 | DUTY0 |
| | | W | | | | | | | | |
| 0x0A01 | LCDCR1 | R | 0 | 0 | 0 | 0 | 0 | 0 | LCDSWAI | LCDRSTP |
| | | W | | | | | | | | |
| 0x0A02 | LCDFPENR0 | R | FP7EN | FP6EN | FP5EN | FP4EN | FP3EN | FP2EN | FP1EN | FP0EN |
| | | W | | | | | | | | |
| 0x0A03 | LCDFPENR1 | R | FP15EN | FP14EN | FP13EN | FP12EN | FP11EN | FP10EN | FP9EN | FP8EN |
| | | W | | | | | | | | |
| 0x0A04 | LCDFPENR2 | R | FP23EN | FP22EN | FP21EN | FP20EN | FP19EN | FP18EN | FP17EN | FP16EN |
| | | W | | | | | | | | |
| 0x0A05 | LCDFPENR3 | R | FP31EN | FP30EN | FP29EN | FP28EN | FP27EN | FP26EN | FP25EN | FP24EN |
| | | W | | | | | | | | |
| 0x0A06 | LCDFPENR4 | R | FP39EN | FP38EN | FP37EN | FP36EN | FP35EN | FP34EN | FP33EN | FP32EN |
| | | W | | | | | | | | |
| 0x0A07 | Reserved | R | | | | | | | | |
| | | W | | | | | | | | |

## 0x0A00–0x0A1F Liquid Crystal Display (LCD)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A08 | LCDRAM0 | R<br>W | FP1BP3 | FP1BP2 | FP1BP1 | FP1BP0 | FP0BP3 | FP0BP2 | FP0BP1 | FP0BP0 |
| 0x0A09 | LCDRAM1 | R<br>W | FP3BP3 | FP3BP2 | FP3BP1 | FP3BP0 | FP2BP3 | FP2BP2 | FP2BP1 | FP2BP0 |
| 0x0A0A | LCDRAM2 | R<br>W | FP5BP3 | FP5BP2 | FP5BP1 | FP5BP0 | FP4BP3 | FP4BP2 | FP4BP1 | FP4BP0 |
| 0x0A0B | LCDRAM3 | R<br>W | FP7BP3 | FP7BP2 | FP7BP1 | FP7BP0 | FP6BP3 | FP6BP2 | FP6BP1 | FP6BP0 |
| 0x0A0C | LCDRAM4 | R<br>W | FP9BP3 | FP9BP2 | FP9BP1 | FP9BP0 | FP8BP3 | FP8BP2 | FP8BP1 | FP8BP0 |
| 0x0A0D | LCDRAM5 | R<br>W | FP11BP3 | FP11BP2 | FP11BP1 | FP11BP0 | FP10BP3 | FP10BP2 | FP10BP1 | FP10BP0 |
| 0x0A0E | LCDRAM6 | R<br>W | FP13BP3 | FP13BP2 | FP13BP1 | FP13BP0 | FP12BP3 | FP12BP2 | FP12BP1 | FP12BP0 |
| 0x0A0F | LCDRAM7 | R<br>W | FP15BP3 | FP15BP2 | FP15BP1 | FP15BP0 | FP14BP3 | FP14BP2 | FP14BP1 | FP14BP0 |
| 0x0A10 | LCDRAM8 | R<br>W | FP17BP3 | FP17BP2 | FP17BP1 | FP17BP0 | FP16BP3 | FP16BP2 | FP16BP1 | FP16BP0 |
| 0x0A11 | LCDRAM9 | R<br>W | FP19BP3 | FP19BP2 | FP19BP1 | FP19BP0 | FP18BP3 | FP18BP2 | FP18BP1 | FP18BP0 |
| 0x0A12 | LCDRAM10 | R<br>W | FP21BP3 | FP21BP2 | FP21BP1 | FP21BP0 | FP20BP3 | FP20BP2 | FP20BP1 | FP20BP0 |
| 0x0A13 | LCDRAM11 | R<br>W | FP23BP3 | FP23BP2 | FP23BP1 | FP23BP0 | FP22BP3 | FP22BP2 | FP22BP1 | FP22BP0 |
| 0x0A14 | LCDRAM12 | R<br>W | FP25BP3 | FP25BP2 | FP25BP1 | FP25BP0 | FP24BP3 | FP24BP2 | FP24BP1 | FP24BP0 |
| 0x0A15 | LCDRAM13 | R<br>W | FP27BP3 | FP27BP2 | FP27BP1 | FP27BP0 | FP26BP3 | FP26BP2 | FP26BP1 | FP26BP0 |
| 0x0A16 | LCDRAM14 | R<br>W | FP29BP3 | FP29BP2 | FP29BP1 | FP29BP0 | FP28BP3 | FP28BP2 | FP28BP1 | FP28BP0 |
| 0x0A17 | LCDRAM15 | R<br>W | FP31BP3 | FP31BP2 | FP31BP1 | FP31BP0 | FP30BP3 | FP30BP2 | FP30BP1 | FP30BP0 |
| 0x0A18 | LCDRAM16 | R<br>W | FP33BP3 | FP33BP2 | FP33BP1 | FP33BP0 | FP32BP3 | FP32BP2 | FP32BP1 | FP32BP0 |
| 0x0A19 | LCDRAM17 | R<br>W | FP35BP3 | FP35BP2 | FP35BP1 | FP35BP0 | FP34BP3 | FP34BP2 | FP34BP1 | FP34BP0 |

## 0x0A00–0x0A1F Liquid Crystal Display (LCD)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A1A | LCDRAM18 | R | FP37BP3 | FP37BP2 | FP37BP1 | FP37BP0 | FP36BP3 | FP36BP2 | FP36BP1 | FP36BP0 |
| | | W | | | | | | | | |
| 0x0A1B | LCDRAM19 | R | FP39BP3 | FP39BP2 | FP39BP1 | FP39BP0 | FP38BP3 | FP38BP2 | FP38BP1 | FP38BP0 |
| | | W | | | | | | | | |
| 0x0A1C-0x0A1F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0A20–0x0A3F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A20-0x0A3F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0A40–0x0A7F Motor Control (MC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A40 | MCCTL0 | R | 0 | MCPRE1 | MCPRE0 | MCSWAI | FAST | DITH | 0 | MCTOIF |
| | | W | | | | | | | | |
| 0x0A41 | MCCTL1 | R | RECIRC | 0 | 0 | 0 | 0 | 0 | 0 | MCTOIE |
| | | W | | | | | | | | |
| 0x0A42 | MCPERH | R | 0 | 0 | 0 | 0 | 0 | P10 | P9 | P8 |
| | | W | | | | | | | | |
| 0x0A43 | MCPERL | R | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| | | W | | | | | | | | |
| 0x0A44–0x0A4F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0A50 | MCCC0 | R | MCOM1 | MCOM0 | MCAM1 | MCAM0 | 0 | 0 | CD1 | CD0 |
| | | W | | | | | | | | |
| 0x0A51 | MCCC1 | R | MCOM1 | MCOM0 | MCAM1 | MCAM0 | 0 | 0 | CD1 | CD0 |
| | | W | | | | | | | | |
| 0x0A52 | MCCC2 | R | MCOM1 | MCOM0 | MCAM1 | MCAM0 | 0 | 0 | CD1 | CD0 |
| | | W | | | | | | | | |
| 0x0A53 | MCCC3 | R | MCOM1 | MCOM0 | MCAM1 | MCAM0 | 0 | 0 | CD1 | CD0 |
| | | W | | | | | | | | |
| 0x0A54–0x0A5F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0A60 | MCDC0H | R | S | S | S | S | S | D10 | D9 | D8 |
| | | W | | | | | | | | |

## 0x0A40–0x0A7F Motor Control (MC)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A61 | MCDC0L | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | | W | | | | | | | | |
| 0x0A62 | MCDC1H | R | S | S | S | S | S | D10 | D9 | D8 |
| | | W | | | | | | | | |
| 0x0A63 | MCDC1L | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | | W | | | | | | | | |
| 0x0A64 | MCDC2H | R | S | S | S | S | S | D10 | D9 | D8 |
| | | W | | | | | | | | |
| 0x0A65 | MCDC2L | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | | W | | | | | | | | |
| 0x0A66 | MCDC3H | R | S | S | S | S | S | D10 | D9 | D8 |
| | | W | | | | | | | | |
| 0x0A67 | MCDC3L | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | | W | | | | | | | | |
| 0x0A68–0x0A7F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0A80–0x0A87 Stepper Stall Detector (SSD0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A80 | RTZ0CTL | R | ITG | DCOIL | RCIR | POL | 0 | 0 | STEP | |
| | | W | | | | | | | | |
| 0x0A81 | MDC0CTL | R | MCZIE | MODMC | RDMCL | PRE | 0 | MCEN | 0 | AOVIE |
| | | W | | | | | FLMC | | | |
| 0x0A82 | SSD0CTL | R | RTZE | SDCPU | SSDWAI | FTST | 0 | 0 | ACLKS | |
| | | W | | | | | | | | |
| 0x0A83 | SSD0FLG | R | MCZIF | 0 | 0 | 0 | 0 | 0 | 0 | AOVIF |
| | | W | | | | | | | | |
| 0x0A84 | MDC0CNTH | R | MDCCNT[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0A85 | MDC0CNTL | R | MDCCNT[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0A86 | ITG0ACCH | R | ITGACC[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0A87 | ITG0ACCL | R | ITGACC[7:0] | | | | | | | |
| | | W | | | | | | | | |

## 0x0A88–0x0A8F Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A88-0x0A8F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0A90–0x0A97 Stepper Stall Detector (SSD1)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A90 | RTZ1CTL | R | ITG | DCOIL | RCIR | POL | 0 | 0 | STEP | |
| | | W | | | | | | | | |
| 0x0A91 | MDC1CTL | R | MCZIE | MODMC | RDMCL | PRE | 0 | MCEN | 0 | AOVIE |
| | | W | | | | | FLMC | | | |
| 0x0A92 | SSD1CTL | R | RTZE | SDCPU | SSDWAI | FTST | 0 | 0 | ACLKS | |
| | | W | | | | | | | | |
| 0x0A93 | SSD1FLG | R | MCZIF | 0 | 0 | 0 | 0 | 0 | 0 | AOVIF |
| | | W | | | | | | | | |
| 0x0A94 | MDC1CNTH | R | MDCCNT[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0A95 | MDC1CNTL | R | MDCCNT[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0A96 | ITG1ACCH | R | ITGACC[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0A97 | ITG1ACCL | R | ITGACC[7:0] | | | | | | | |
| | | W | | | | | | | | |

## 0x0A98–0x0ADF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A98-0x0ADF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0AE0–0x0AEF Real Time Clock (RTC)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0AE0 | RTCCTL1 | R | RTCEN | 0 | 0 | 0 | 0 | 0 | COMPE | 0 |
| | | W | | | | | | | | |
| 0x0AE1 | RTCCTL2 | R | CLKSRC1 | CLKSRC0 | 0 | 0 | RTCPS3 | RTCPS2 | RTCPS1 | RTCPS0 |
| | | W | | | | | | | | |
| 0x0AE2 | RTCCTL3 | R | | | 0 | 0 | FRZ | 0 | CALS | 0 |
| | | W | RTCWE1 | RTCWE0 | | | | | | |

## 0x0AE0–0x0AEF Real Time Clock (RTC)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x0AE3 | RTCCTL4 | R | 0 | 0 | HRIE | MINIE | SECIE | COMPIE | 0 | TB0IE |
| | | W | | | | | | | | |
| 0x0AE4 | RTCS1 | R | CDLC | 0 | HRF | MINF | SECF | COMPF | 0 | TB0F |
| | | W | | | | | | | | |
| 0x0AE5 | RTCCCR | R | CCS | Q | | | | | | |
| | | W | | | | | | | | |
| 0x0AE6 | RTCMODH | R | RTCMODH | | | | | | | |
| | | W | | | | | | | | |
| 0x0AE7 | RTCMODL | R | RTCMODL | | | | | | | |
| | | W | | | | | | | | |
| 0x0AE8 | RTCCNTH | R | RTCCNTH | | | | | | | |
| | | W | | | | | | | | |
| 0x0AE9 | RTCCNTL | R | RTCCNTL | | | | | | | |
| | | W | | | | | | | | |
| 0x0AEA–0x0AEC | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0AED | RTCSEC | R | 0 | 0 | SEC5 | SEC4 | SEC3 | SEC2 | SEC1 | SEC0 |
| | | W | | | | | | | | |
| 0x0AEE | RTCMIN | R | 0 | 0 | MIN5 | MIN4 | MIN3 | MIN2 | MIN1 | MIN0 |
| | | W | | | | | | | | |
| 0x0AEF | RTCHR | R | 0 | 0 | 0 | HR4 | HR3 | HR2 | HR1 | HR0 |
| | | W | | | | | | | | |

## 0x0AF0–0x0AFF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0AF0-0x0AFF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0B00–0x0B17 Simple Sound Generator(SSG0)

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0B00 | SSG0CR | R | SSGE | 0 | 0 | 0 | 0 | OMS | RDR | STP |
| | | W | | | | | | | | |
| 0x0B01 | SSG0ADC | R | ADE | 0 | 0 | 0 | 0 | ADM[1:0] | | ADS |
| | | W | | | | | | | | |
| 0x0B02 | SSG0PSH | R | 0 | 0 | 0 | 0 | 0 | PS10 | PS9 | PS8 |
| | | W | | | | | | | | |

## 0x0B00–0x0B17 Simple Sound Generator(SSG0)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0B03 | SSG0PSL | R | PS7 | PS6 | PS5 | PS4 | PS3 | PS2 | PS1 | PS0 |
| | | W | | | | | | | | |
| 0x0B04 | SSG0TONEH | R | 0 | 0 | 0 | 0 | 0 | 0 | TONE9 | TONE8 |
| | | W | | | | | | | | |
| 0x0B05 | SSG0TONEL | R | TONE7 | TONE6 | TONE5 | TONE4 | TONE3 | TONE2 | TONE1 | TONE0 |
| | | W | | | | | | | | |
| 0x0B06 | SSG0AMPH | R | 0 | 0 | 0 | 0 | 0 | AMP10 | AMP9 | AMP8 |
| | | W | | | | | | | | |
| 0x0B07 | SSG0AMPL | R | AMP7 | AMP6 | AMP5 | AMP4 | AMP3 | AMP2 | AMP1 | AMP0 |
| | | W | | | | | | | | |
| 0x0B08 | SSG0AAH | R | 0 | 0 | 0 | 0 | 0 | AA10 | AA9 | AA8 |
| | | W | | | | | | | | |
| 0x0B09 | SSG0AAL | R | AA7 | AA6 | AA5 | AA4 | AA3 | AA2 | AA1 | AA0 |
| | | W | | | | | | | | |
| 0x0B0A | SSG0ATH | R | 0 | 0 | 0 | 0 | 0 | AT10 | AT9 | AT8 |
| | | W | | | | | | | | |
| 0x0B0B | SSG0ATL | R | AT7 | AT6 | AT5 | AT4 | AT3 | AT2 | AT1 | AT0 |
| | | W | | | | | | | | |
| 0x0B0C | SSG0DUR | R | DUR7 | DUR6 | DUR5 | DUR4 | DUR3 | DUR2 | DUR1 | DUR0 |
| | | W | | | | | | | | |
| 0x0B0D | SSG0IE | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RNDIE |
| | | W | | | | | | | | |
| 0xC0E | SSG0IF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RNDI |
| | | W | | | | | | | | |
| 0x0B0F | RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0B10 | SSG0AMPBH | R | 0 | 0 | 0 | 0 | 0 | AMPB10 | AMPB9 | AMPB8 |
| | | W | | | | | | | | |
| 0x0B11 | SSG0AMPBL | R | AMPB7 | AMPB6 | AMPB5 | AMPB4 | AMPB3 | AMPB2 | AMP1 | AMPB0 |
| | | W | | | | | | | | |
| 0x0B12 | SSG0DCNT | R | DCNT7 | DCNT6 | DCNT5 | DCNT4 | DCNT3 | DCNT2 | DCNT1 | DCNT0 |
| | | W | | | | | | | | |
| 0x0B13-0x0B17 | RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

## 0x0B18–0x0FFF Reserved

| Address | Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0B18-0x0FFF | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

***How to Reach Us:***

**Home Page:**
freescale.com

**Web Support:**
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. MagniV is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.

Document Number: MC9S12ZVHYRMV1
Rev. 1.00
09/2013

freescale™