# THE DATASHEET OF

# S9S08DZ16F1CLC

# MC9S08DZ60
# MC9S08DZ48
# MC9S08DZ32
# MC9S08DZ16

Data Sheet

***HCS08***
***Microcontrollers***

MC9S08DZ60
Rev. 4
6/2008

freescale.com

# MC9S08DZ60 Series Features

## 8-Bit HCS08 Central Processor Unit (CPU)

- 40-MHz HCS08 CPU (20-MHz bus)
- HC08 instruction set with added BGND instruction
- Support for up to 32 interrupt/reset sources

## On-Chip Memory

- Flash read/program/erase over full operating voltage and temperature
  - MC9S08DZ60 = 60K
  - MC9S08DZ48 = 48K
  - MC9S08DZ32 = 32K
  - MC9S08DZ16 = 16K
- Up to 2K EEPROM in-circuit programmable memory; 8-byte single-page or 4-byte dual-page erase sector; Program and Erase while executing Flash; Erase abort
- Up to 4K random-access memory (RAM)

## Power-Saving Modes

- Two very low power stop modes
- Reduced power wait mode
- Very low power real time interrupt for use in run, wait, and stop

## Clock Source Options

- Oscillator (XOSC) — Loop-control Pierce oscillator; Crystal or ceramic resonator range of 31.25 kHz to 38.4 kHz or 1 MHz to 16 MHz
- Multi-purpose Clock Generator (MCG) — PLL and FLL modes (FLL capable of 1.5% deviation using internal temperature compensation); Internal reference clock with trim adjustment (trimmed at factory, with trim value stored in flash); External reference with oscillator/resonator options

## System Protection

- Watchdog computer operating properly (COP) reset with option to run from backup dedicated 1-kHz internal clock source or bus clock
- Low-voltage detection with reset or interrupt; selectable trip points
- Illegal opcode detection with reset
- Illegal address detection with reset
- Flash block protect
- Loss-of-lock protection

## Development Support

- Single-wire background debug interface
- On-chip, in-circuit emulation (ICE) with real-time bus capture

## Peripherals

- **ADC** — 24-channel, 12-bit resolution, 2.5 µs conversion time, automatic compare function, temperature sensor, internal bandgap reference channel
- **ACMPx** — Two analog comparators with selectable interrupt on rising, falling, or either edge of comparator output; compare option to fixed internal bandgap reference voltage
- **MSCAN** — CAN protocol - Version 2.0 A, B; standard and extended data frames; Support for remote frames; Five receive buffers with FIFO storage scheme; Flexible identifier acceptance filters programmable as: 2 x 32-bit, 4 x 16-bit, or 8 x 8-bit
- **SCIx** — Two SCIs supporting LIN 2.0 Protocol and SAE J2602 protocols; Full duplex non-return to zero (NRZ); Master extended break generation; Slave extended break detection; Wakeup on active edge
- **SPI** — Full-duplex or single-wire bidirectional; Double-buffered transmit and receive; Master or Slave mode; MSB-first or LSB-first shifting
- **IIC** — Up to 100 kbps with maximum bus loading; Multi-master operation; Programmable slave address; General Call Address; Interrupt driven byte-by-byte data transfer
- **TPMx** — One 6-channel (TPM1) and one 2-channel (TPM2); Selectable input capture, output compare, or buffered edge-aligned PWM on each channel
- **RTC** — (Real-time counter) 8-bit modulus counter with binary or decimal based prescaler; Real-time clock capabilities using external crystal and RTC for precise time base, time-of-day, calendar or task scheduling functions; Free running on-chip low power oscillator (1 kHz) for cyclic wake-up without external components

## Input/Output

- 53 general-purpose input/output (I/O) pins and 1 input-only pin
- 24 interrupt pins with selectable polarity on each pin
- Hysteresis and configurable pull device on all input pins.
- Configurable slew rate and drive strength on all output pins.

## Package Options

- 64-pin low-profile quad flat-pack (LQFP) — 10x10 mm
- 48-pin low-profile quad flat-pack (LQFP) — 7x7 mm
- 32-pin low-profile quad flat-pack (LQFP) — 7x7 mm

# MC9S08DZ60 Data Sheet

Covers MC9S08DZ60
MC9S08DZ48
MC9S08DZ32
MC9S08DZ16

MC9S08DZ60
Rev. 4
6/2008

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.

© Freescale Semiconductor, Inc., 2007-2008. All rights reserved.

freescale™
semiconductor

# Revision History

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:

http://freescale.com/

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

| Revision Number | Revision Date | Description of Changes |
|---|---|---|
| 1 | 6/2006 | Advance Information for alpha samples customers |
| 2 | 9/2007 | Product Launch. Removed the 64-pin QFN package. Changed from standard to extended mode for MSCAN registers in register summary. Corrected Block diagrams for SCI. Updated the latest Temp Sensor information. Made FTSTMOD reserved. Updated device to use the ADC 12-bit module. Revised the MCG module. Updated the CPU Instruction Set table. Updated the TPM block module to version 3. Added the TPM block module version 2 as an appendix for devices using 3M05C (or earlier) mask sets. Heavily revised the Electricals appendix. |
| 3 | 10/2007 | Removed two tables that were inadvertently included in the MC9S08DZ60 version of the book. |
| 4 | 6/2008 | Sustaining update. Incorporated PS Issues # 2765, 3177, 3236, 3292, 3311, 3312, 3326, 3335, 3345, 3382, 2795, 3382 and 3386 PLL Jitter Spec update. Also, added internal reference clock trim adjustment statement to Features page. Updated the TPM module to the latest version. Adjusted values in Table A-13 Control Timing row 2 and in Table A-6 DC Characteristics row 24 so that it references 5.0 V instead of 3.0 V. |

© Freescale Semiconductor, Inc., 2007-2008. All rights reserved.

This product incorporates SuperFlash® Technology licensed from SST.

# List of Chapters


| Chapter | Title | Page |
|---|---|---|
| Chapter 1 | Device Overview | 21 |
| Chapter 2 | Pins and Connections | 27 |
| Chapter 3 | Modes of Operation | 35 |
| Chapter 4 | Memory | 41 |
| Chapter 5 | Resets, Interrupts, and General System Control | 69 |
| Chapter 6 | Parallel Input/Output Control | 85 |
| Chapter 7 | Central Processor Unit (S08CPUV3) | 115 |
| Chapter 8 | Multi-Purpose Clock Generator (S08MCGV1) | 135 |
| Chapter 9 | Analog Comparator (S08ACMPV3) | 167 |
| Chapter 10 | Analog-to-Digital Converter (S08ADC12V1) | 173 |
| Chapter 11 | Inter-Integrated Circuit (S08IICV2) | 199 |
| Chapter 12 | Freescale Controller Area Network (S08MSCANV1) | 219 |
| Chapter 13 | Serial Peripheral Interface (S08SPIV3) | 273 |
| Chapter 14 | Serial Communications Interface (S08SCIV4) | 289 |
| Chapter 15 | Real-Time Counter (S08RTCV1) | 309 |
| Chapter 16 | Timer Pulse-Width Modulator (S08TPMV3) | 319 |
| Chapter 17 | Development Support | 347 |
| Appendix A | Electrical Characteristics | 369 |
| Appendix B | Timer Pulse-Width Modulator (TPMV2) | 391 |
| Appendix C | Ordering Information and Mechanical Drawings | 405 |

# Contents

**Section Number** **Title** **Page**


## Chapter 1
## Device Overview

1.1 Devices in the MC9S08DZ60 Series....21
1.2 MCU Block Diagram....22
1.3 System Clock Distribution....24

## Chapter 2
## Pins and Connections

2.1 Device Pin Assignment....27
2.2 Recommended System Connections....30
2.2.1 Power....31
2.2.2 Oscillator....31
2.2.3 $\overline{\text{RESET}}$....31
2.2.4 Background / Mode Select (BKGD/MS)....32
2.2.5 ADC Reference Pins ($V_{REFH}$, $V_{REFL}$)....32
2.2.6 General-Purpose I/O and Peripheral Ports....32

## Chapter 3
## Modes of Operation

3.1 Introduction....35
3.2 Features....35
3.3 Run Mode....35
3.4 Active Background Mode....35
3.5 Wait Mode....36
3.6 Stop Modes....37
3.6.1 Stop3 Mode....37
3.6.2 Stop2 Mode....38
3.6.3 On-Chip Peripheral Modules in Stop Modes....39

## Chapter 4
## Memory

4.1 MC9S08DZ60 Series Memory Map....41
4.2 Reset and Interrupt Vector Assignments....42
4.3 Register Addresses and Bit Assignments....44
4.4 RAM....52
4.5 Flash and EEPROM....52
4.5.1 Features....52

4.5.2 Program and Erase Times ....53
4.5.3 Program and Erase Command Execution ....53
4.5.4 Burst Program Execution ....55
4.5.5 Sector Erase Abort ....57
4.5.6 Access Errors ....58
4.5.7 Block Protection ....59
4.5.8 Vector Redirection ....59
4.5.9 Security ....59
4.5.10 EEPROM Mapping ....61
4.5.11 Flash and EEPROM Registers and Control Bits ....61

## Chapter 5
## Resets, Interrupts, and General System Control

5.1 Introduction ....69
5.2 Features ....69
5.3 MCU Reset ....69
5.4 Computer Operating Properly (COP) Watchdog ....70
5.5 Interrupts ....71
5.5.1 Interrupt Stack Frame ....72
5.5.2 External Interrupt Request (IRQ) Pin ....72
5.5.3 Interrupt Vectors, Sources, and Local Masks ....73
5.6 Low-Voltage Detect (LVD) System ....75
5.6.1 Power-On Reset Operation ....75
5.6.2 Low-Voltage Detection (LVD) Reset Operation ....75
5.6.3 Low-Voltage Warning (LVW) Interrupt Operation ....75
5.7 MCLK Output ....75
5.8 Reset, Interrupt, and System Control Registers and Control Bits ....76
5.8.1 Interrupt Pin Request Status and Control Register (IRQSC) ....77
5.8.2 System Reset Status Register (SRS) ....78
5.8.3 System Background Debug Force Reset Register (SBDFR) ....79
5.8.4 System Options Register 1 (SOPT1) ....80
5.8.5 System Options Register 2 (SOPT2) ....81
5.8.6 System Device Identification Register (SDIDH, SDIDL) ....82
5.8.7 System Power Management Status and Control 1 Register (SPMSC1) ....83
5.8.8 System Power Management Status and Control 2 Register (SPMSC2) ....84

## Chapter 6
## Parallel Input/Output Control

6.1 Port Data and Data Direction ....85
6.2 Pull-up, Slew Rate, and Drive Strength ....86
6.3 Pin Interrupts ....87
6.3.1 Edge Only Sensitivity ....87

6.3.2 Edge and Level Sensitivity ....88
6.3.3 Pull-up/Pull-down Resistors ....88
6.3.4 Pin Interrupt Initialization ....88
6.4 Pin Behavior in Stop Modes....88
6.5 Parallel I/O and Pin Control Registers ....89
6.5.1 Port A Registers ....90
6.5.2 Port B Registers ....94
6.5.3 Port C Registers ....98
6.5.4 Port D Registers ....101
6.5.5 Port E Registers ....105
6.5.6 Port F Registers ....108
6.5.7 Port G Registers ....111

## Chapter 7
## Central Processor Unit (S08CPUV3)

7.1 Introduction ....115
7.1.1 Features ....115
7.2 Programmer’s Model and CPU Registers ....116
7.2.1 Accumulator (A) ....116
7.2.2 Index Register (H:X) ....116
7.2.3 Stack Pointer (SP) ....117
7.2.4 Program Counter (PC) ....117
7.2.5 Condition Code Register (CCR) ....117
7.3 Addressing Modes....119
7.3.1 Inherent Addressing Mode (INH) ....119
7.3.2 Relative Addressing Mode (REL) ....119
7.3.3 Immediate Addressing Mode (IMM) ....119
7.3.4 Direct Addressing Mode (DIR) ....119
7.3.5 Extended Addressing Mode (EXT) ....120
7.3.6 Indexed Addressing Mode ....120
7.4 Special Operations....121
7.4.1 Reset Sequence ....121
7.4.2 Interrupt Sequence ....121
7.4.3 Wait Mode Operation ....122
7.4.4 Stop Mode Operation ....122
7.4.5 BGND Instruction ....123
7.5 HCS08 Instruction Set Summary ....124

## Chapter 8
## Multi-Purpose Clock Generator (S08MCGV1)

8.1 Introduction ....135
8.1.1 Features ....137

8.1.2 Modes of Operation ....139
8.2 External Signal Description ....139
8.3 Register Definition ....140
8.3.1 MCG Control Register 1 (MCGC1) ....140
8.3.2 MCG Control Register 2 (MCGC2) ....141
8.3.3 MCG Trim Register (MCGTRM) ....142
8.3.4 MCG Status and Control Register (MCGSC) ....143
8.3.5 MCG Control Register 3 (MCGC3) ....144
8.4 Functional Description ....146
8.4.1 Operational Modes ....146
8.4.2 Mode Switching ....150
8.4.3 Bus Frequency Divider ....151
8.4.4 Low Power Bit Usage ....151
8.4.5 Internal Reference Clock ....151
8.4.6 External Reference Clock ....151
8.4.7 Fixed Frequency Clock ....152
8.5 Initialization / Application Information ....152
8.5.1 MCG Module Initialization Sequence ....152
8.5.2 MCG Mode Switching ....153
8.5.3 Calibrating the Internal Reference Clock (IRC) ....164

## Chapter 9
## Analog Comparator (S08ACMPV3)

9.1 Introduction ....167
9.1.1 ACMP Configuration Information ....167
9.1.2 Features ....169
9.1.3 Modes of Operation ....169
9.1.4 Block Diagram ....170
9.2 External Signal Description ....170
9.3 Memory Map/Register Definition ....171
9.3.1 ACMPx Status and Control Register (ACMPxSC) ....171
9.4 Functional Description ....172

## Chapter 10
## Analog-to-Digital Converter (S08ADC12V1)

10.1 Introduction ....173
10.1.1 Analog Power and Ground Signal Names ....173
10.1.2 Channel Assignments ....173
10.1.3 Alternate Clock ....174
10.1.4 Hardware Trigger ....174
10.1.5 Temperature Sensor ....175
10.1.6 Features ....177

10.1.7 ADC Module Block Diagram ....177
10.2 External Signal Description ....178
10.2.1 Analog Power ($V_{DDAD}$) ....179
10.2.2 Analog Ground ($V_{SSAD}$) ....179
10.2.3 Voltage Reference High ($V_{REFH}$) ....179
10.2.4 Voltage Reference Low ($V_{REFL}$) ....179
10.2.5 Analog Channel Inputs (ADx) ....179
10.3 Register Definition ....179
10.3.1 Status and Control Register 1 (ADCSC1) ....179
10.3.2 Status and Control Register 2 (ADCSC2) ....181
10.3.3 Data Result High Register (ADCRH) ....181
10.3.4 Data Result Low Register (ADCRL) ....182
10.3.5 Compare Value High Register (ADCCVH) ....182
10.3.6 Compare Value Low Register (ADCCVL) ....183
10.3.7 Configuration Register (ADCCFG) ....183
10.3.8 Pin Control 1 Register (APCTL1) ....184
10.3.9 Pin Control 2 Register (APCTL2) ....185
10.3.10Pin Control 3 Register (APCTL3) ....186
10.4 Functional Description ....187
10.4.1 Clock Select and Divide Control ....188
10.4.2 Input Select and Pin Control ....188
10.4.3 Hardware Trigger ....188
10.4.4 Conversion Control ....188
10.4.5 Automatic Compare Function ....191
10.4.6 MCU Wait Mode Operation ....191
10.4.7 MCU Stop3 Mode Operation ....192
10.4.8 MCU Stop2 Mode Operation ....192
10.5 Initialization Information ....193
10.5.1 ADC Module Initialization Example ....193
10.6 Application Information ....195
10.6.1 External Pins and Routing ....195
10.6.2 Sources of Error ....196

## Chapter 11
## Inter-Integrated Circuit (S08IICV2)

11.1 Introduction ....199
11.1.1 Features ....201
11.1.2 Modes of Operation ....201
11.1.3 Block Diagram ....202
11.2 External Signal Description ....202
11.2.1 SCL — Serial Clock Line ....202
11.2.2 SDA — Serial Data Line ....202

11.3 Register Definition ....202
11.3.1 IIC Address Register (IICA) ....203
11.3.2 IIC Frequency Divider Register (IICF) ....203
11.3.3 IIC Control Register (IICC1) ....206
11.3.4 IIC Status Register (IICS) ....207
11.3.5 IIC Data I/O Register (IICD) ....208
11.3.6 IIC Control Register 2 (IICC2) ....208
11.4 Functional Description ....209
11.4.1 IIC Protocol ....209
11.4.2 10-bit Address ....213
11.4.3 General Call Address ....214
11.5 Resets ....214
11.6 Interrupts ....214
11.6.1 Byte Transfer Interrupt ....214
11.6.2 Address Detect Interrupt ....214
11.6.3 Arbitration Lost Interrupt ....214
11.7 Initialization/Application Information ....216

## Chapter 12
## Freescale Controller Area Network (S08MSCANV1)

12.1 Introduction ....219
12.1.1 Features ....221
12.1.2 Modes of Operation ....221
12.1.3 Block Diagram ....222
12.2 External Signal Description ....222
12.2.1 RXCAN — CAN Receiver Input Pin ....222
12.2.2 TXCAN — CAN Transmitter Output Pin ....222
12.2.3 CAN System ....222
12.3 Register Definition ....223
12.3.1 MSCAN Control Register 0 (CANCTL0) ....223
12.3.2 MSCAN Control Register 1 (CANCTL1) ....226
12.3.3 MSCAN Bus Timing Register 0 (CANBTR0) ....227
12.3.4 MSCAN Bus Timing Register 1 (CANBTR1) ....228
12.3.5 MSCAN Receiver Interrupt Enable Register (CANRIER) ....231
12.3.6 MSCAN Transmitter Flag Register (CANTFLG) ....232
12.3.7 MSCAN Transmitter Interrupt Enable Register (CANTIER) ....233
12.3.8 MSCAN Transmitter Message Abort Request Register (CANTARQ) ....234
12.3.9 MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK) ....235
12.3.10MSCAN Transmit Buffer Selection Register (CANTBSEL) ....235
12.3.11MSCAN Identifier Acceptance Control Register (CANIDAC) ....236
12.3.12MSCAN Miscellaneous Register (CANMISC) ....237
12.3.13MSCAN Receive Error Counter (CANRXERR) ....238

12.3.14MSCAN Transmit Error Counter (CANTXERR) ....239
12.3.15MSCAN Identifier Acceptance Registers (CANIDAR0-7) ....239
12.3.16MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7) ....240
12.4 Programmer's Model of Message Storage ....241
12.4.1 Identifier Registers (IDR0–IDR3) ....244
12.4.2 IDR0–IDR3 for Standard Identifier Mapping ....246
12.4.3 Data Segment Registers (DSR0-7) ....247
12.4.4 Data Length Register (DLR) ....248
12.4.5 Transmit Buffer Priority Register (TBPR) ....249
12.4.6 Time Stamp Register (TSRH–TSRL) ....249
12.5 Functional Description ....250
12.5.1 General ....250
12.5.2 Message Storage ....251
12.5.3 Identifier Acceptance Filter ....254
12.5.4 Modes of Operation ....261
12.5.5 Low-Power Options ....262
12.5.6 Reset Initialization ....268
12.5.7 Interrupts ....268
12.6 Initialization/Application Information ....270
12.6.1 MSCAN initialization ....270
12.6.2 Bus-Off Recovery ....271

## Chapter 13
## Serial Peripheral Interface (S08SPIV3)

13.1 Introduction ....273
13.1.1 Features ....275
13.1.2 Block Diagrams ....275
13.1.3 SPI Baud Rate Generation ....277
13.2 External Signal Description ....278
13.2.1 SPSCK — SPI Serial Clock ....278
13.2.2 MOSI — Master Data Out, Slave Data In ....278
13.2.3 MISO — Master Data In, Slave Data Out ....278
13.2.4 $\overline{SS}$ — Slave Select ....278
13.3 Modes of Operation ....279
13.3.1 SPI in Stop Modes ....279
13.4 Register Definition ....279
13.4.1 SPI Control Register 1 (SPIC1) ....279
13.4.2 SPI Control Register 2 (SPIC2) ....280
13.4.3 SPI Baud Rate Register (SPIBR) ....281
13.4.4 SPI Status Register (SPIS) ....282
13.4.5 SPI Data Register (SPID) ....283
13.5 Functional Description ....284

13.5.1 SPI Clock Formats ....284
13.5.2 SPI Interrupts ....287
13.5.3 Mode Fault Detection ....287

## Chapter 14
## Serial Communications Interface (S08SCIV4)

14.1 Introduction ....289
14.1.1 SCI2 Configuration Information ....289
14.1.2 Features ....291
14.1.3 Modes of Operation ....291
14.1.4 Block Diagram ....292
14.2 Register Definition ....294
14.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL) ....294
14.2.2 SCI Control Register 1 (SCIxC1) ....295
14.2.3 SCI Control Register 2 (SCIxC2) ....296
14.2.4 SCI Status Register 1 (SCIxS1) ....297
14.2.5 SCI Status Register 2 (SCIxS2) ....299
14.2.6 SCI Control Register 3 (SCIxC3) ....300
14.2.7 SCI Data Register (SCIxD) ....301
14.3 Functional Description ....301
14.3.1 Baud Rate Generation ....301
14.3.2 Transmitter Functional Description ....302
14.3.3 Receiver Functional Description ....303
14.3.4 Interrupts and Status Flags ....305
14.3.5 Additional SCI Functions ....306

## Chapter 15
## Real-Time Counter (S08RTCV1)

15.1 Introduction ....309
15.1.1 RTC Clock Signal Names ....309
15.1.2 Features ....311
15.1.3 Modes of Operation ....311
15.1.4 Block Diagram ....312
15.2 External Signal Description ....312
15.3 Register Definition ....312
15.3.1 RTC Status and Control Register (RTCSC) ....313
15.3.2 RTC Counter Register (RTCCNT) ....314
15.3.3 RTC Modulo Register (RTCMOD) ....314
15.4 Functional Description ....314
15.4.1 RTC Operation Example ....315
15.5 Initialization/Application Information ....316

## Chapter 16
## Timer Pulse-Width Modulator (S08TPMV3)

16.1 Introduction ..... 319
16.1.1 Features ..... 321
16.1.2 Modes of Operation ..... 321
16.1.3 Block Diagram ..... 322
16.2 Signal Description ..... 324
16.2.1 Detailed Signal Descriptions ..... 324
16.3 Register Definition ..... 328
16.3.1 TPM Status and Control Register (TPMxSC) ..... 328
16.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL) ..... 329
16.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL) ..... 330
16.3.4 TPM Channel n Status and Control Register (TPMxCnSC) ..... 331
16.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL) ..... 333
16.4 Functional Description ..... 334
16.4.1 Counter ..... 335
16.4.2 Channel Mode Selection ..... 337
16.5 Reset Overview ..... 340
16.5.1 General ..... 340
16.5.2 Description of Reset Operation ..... 340
16.6 Interrupts ..... 340
16.6.1 General ..... 340
16.6.2 Description of Interrupt Operation ..... 341
16.7 The Differences from TPM v2 to TPM v3 ..... 342

## Chapter 17
## Development Support

17.1 Introduction ..... 347
17.1.1 Forcing Active Background ..... 347
17.1.2 Features ..... 348
17.2 Background Debug Controller (BDC) ..... 348
17.2.1 BKGD Pin Description ..... 349
17.2.2 Communication Details ..... 350
17.2.3 BDC Commands ..... 354
17.2.4 BDC Hardware Breakpoint ..... 356
17.3 On-Chip Debug System (DBG) ..... 357
17.3.1 Comparators A and B ..... 357
17.3.2 Bus Capture Information and FIFO Operation ..... 357
17.3.3 Change-of-Flow Information ..... 358
17.3.4 Tag vs. Force Breakpoints and Triggers ..... 358
17.3.5 Trigger Modes ..... 359
17.3.6 Hardware Breakpoints ..... 361

17.4 Register Definition ....361
17.4.1 BDC Registers and Control Bits ....361
17.4.2 System Background Debug Force Reset Register (SBDFR) ....363
17.4.3 DBG Registers and Control Bits ....364

## Appendix A
## Electrical Characteristics

A.1 Introduction ....369
A.2 Parameter Classification ....369
A.3 Absolute Maximum Ratings ....369
A.4 Thermal Characteristics ....370
A.5 ESD Protection and Latch-Up Immunity ....372
A.6 DC Characteristics ....373
A.7 Supply Current Characteristics ....375
A.8 Analog Comparator (ACMP) Electricals ....376
A.9 ADC Characteristics ....376
A.10 External Oscillator (XOSC) Characteristics ....380
A.11 MCG Specifications ....381
A.12 AC Characteristics ....383
A.12.1 Control Timing ....383
A.12.2 Timer/PWM ....384
A.12.3 MSCAN ....385
A.12.4 SPI ....386
A.13 Flash and EEPROM ....389
A.14 EMC Performance ....390
A.14.1 Radiated Emissions ....390

## Appendix B
## Timer Pulse-Width Modulator (TPMV2)

B.0.1 Features ....391
B.0.2 Block Diagram ....391
B.1 External Signal Description ....393
B.1.1 External TPM Clock Sources ....393
B.1.2 TPMxCHn — TPMx Channel n I/O Pins ....393
B.2 Register Definition ....393
B.2.1 Timer Status and Control Register (TPMxSC) ....394
B.2.2 Timer Counter Registers (TPMxCNTH:TPMxCNTL) ....395
B.2.3 Timer Counter Modulo Registers (TPMxMODH:TPMxMODL) ....396
B.2.4 Timer Channel n Status and Control Register (TPMxCnSC) ....397
B.2.5 Timer Channel Value Registers (TPMxCnVH:TPMxCnVL) ....398
B.3 Functional Description ....399
B.3.1 Counter ....399

B.3.2 Channel Mode Selection ....400
B.3.3 Center-Aligned PWM Mode ....402
B.4 TPM Interrupts ....403
B.4.1 Clearing Timer Interrupt Flags ....403
B.4.2 Timer Overflow Interrupt Description ....403
B.4.3 Channel Event Interrupt Description ....404
B.4.4 PWM End-of-Duty-Cycle Events ....404

## Appendix C
## Ordering Information and Mechanical Drawings

C.1 Ordering Information ....405
C.1.1 MC9S08DZ60 Series Devices ....405
C.2 Mechanical Drawings ....405

# Chapter 1
# Device Overview

MC9S08DZ60 Series devices provide significant value to customers looking to combine Controller Area Network (CAN) and embedded EEPROM in their applications. This combination will provide lower costs, enhanced performance, and higher quality.

## 1.1 Devices in the MC9S08DZ60 Series

This data sheet covers members of the MC9S08DZ60 Series of MCUs:

- MC9S08DZ60
- MC9S08DZ48
- MC9S08DZ32
- MC9S08DZ16

Table 1-1 summarizes the feature set available in the MC9S08DZ60 Series.

**Table 1-1. MC9S08DZ60 Series Features by MCU and Pin Count**

| Feature | MC9S08DZ60 | | | MC9S08DZ48 | | | MC9S08DZ32 | | | MC9S08DZ16 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Flash size (bytes) | 60032 | | | 49152 | | | 33792 | | | 16896 | |
| RAM size (bytes) | 4096 | | | 3072 | | | 2048 | | | 1024 | |
| EEPROM size (bytes) | 2048 | | | 1536 | | | 1024 | | | 512 | |
| Pin quantity | 64 | 48 | 32 | 64 | 48 | 32 | 64 | 48 | 32 | 48 | 32 |
| ACMP1 | yes | | | | | | | | | | |
| ACMP2 | yes | yes[1] | no | yes | yes[1] | no | yes | yes[1] | no | yes[1] | no |
| ADC channels | 24 | 16 | 10 | 24 | 16 | 10 | 24 | 16 | 10 | 16 | 10 |
| DBG | yes | | | | | | | | | | |
| IIC | yes | | | | | | | | | | |
| IRQ | yes | | | | | | | | | | |
| MCG | yes | | | | | | | | | | |
| MSCAN | yes | | | | | | | | | | |
| RTC | yes | | | | | | | | | | |
| SCI1 | yes | | | | | | | | | | |
| SCI2 | yes | | | | | | | | | | |
| SPI | yes | | | | | | | | | | |
| TPM1 channels | 6 | 6 | 4 | 6 | 6 | 4 | 6 | 6 | 4 | 6 | 4 |
| TPM2 channels | 2 | | | | | | | | | | |
| XOSC | yes | | | | | | | | | | |
| COP Watchdog | yes | | | | | | | | | | |

1 ACMP2O is not available.

## 1.2 MCU Block Diagram

Figure 1-1 is the MC9S08DZ60 Series system-level block diagram.

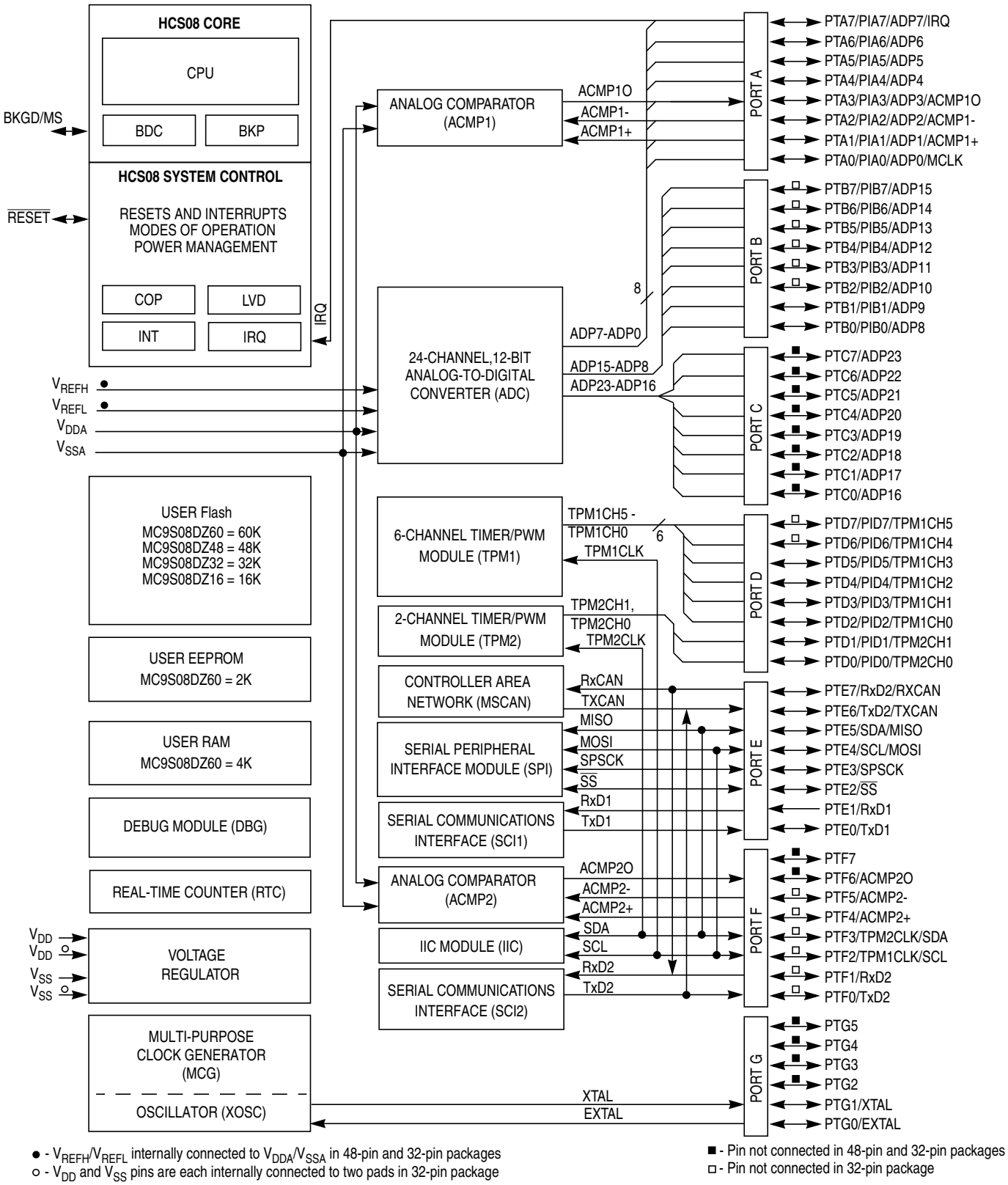



**Figure 1-1. MC9S08DZ60 Block Diagram**

Table 1-2 provides the functional version of the on-chip modules.

**Table 1-2. Module Versions**

| Module | | Version |
|---|---|---|
| Central Processor Unit | (CPU) | 3 |
| Multi-Purpose Clock Generator | (MCG) | 1 |
| Analog Comparator | (ACMP) | 3 |
| Analog-to-Digital Converter | (ADC) | 1 |
| Inter-Integrated Circuit | (IIC) | 2 |
| Freescale's CAN | (MSCAN) | 1 |
| Serial Peripheral Interface | (SPI) | 3 |
| Serial Communications Interface | (SCI) | 4 |
| Real-Time Counter | (RTC) | 1 |
| Timer Pulse Width Modulator | (TPM) | 3[1] |
| Debug Module | (DBG) | 2 |

[1] 3M05C and older masks have TPM version 2.

## 1.3 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs as shown. The clock inputs to the modules indicate the clock(s) that are used to drive the module function.

The following are the clocks used in this MCU:

- BUSCLK — The frequency of the bus is always half of MCGOUT.
- LPO — Independent 1-kHz clock that can be selected as the source for the COP and RTC modules.
- MCGOUT — Primary output of the MCG and is twice the bus frequency.
- MCGLCLK — Development tools can select this clock source to speed up BDC communications in systems where BUSCLK is configured to run at a very slow frequency.
- MCGERCLK — External reference clock can be selected as the RTC clock source. It can also be used as the alternate clock for the ADC and MSCAN.
- MCGIRCLK — Internal reference clock can be selected as the RTC clock source.
- MCGFFCLK — Fixed frequency clock can be selected as clock source for the TPM1 and TPM2.
- TPM1CLK — External input clock source for TPM1.
- TPM2CLK — External input clock source for TPM2.

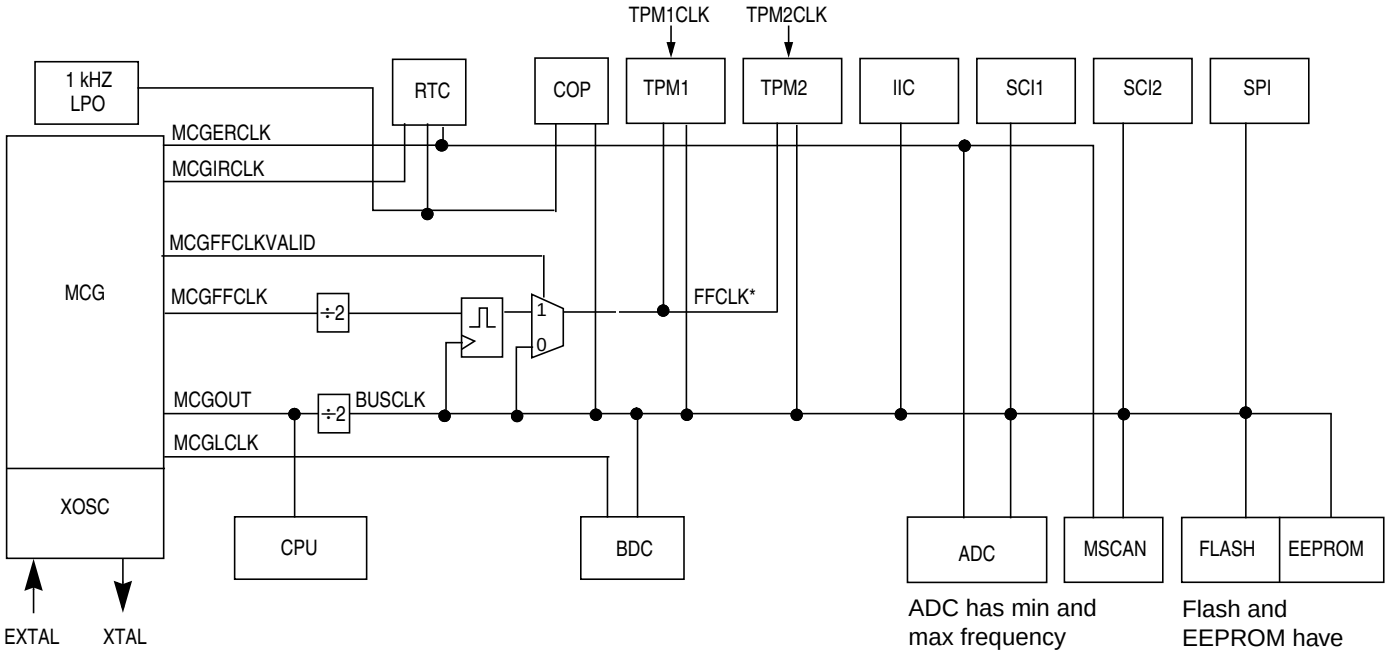



**Figure 1-2. MC9S08DZ60 System Clock Distribution Diagram**

# Chapter 2
# Pins and Connections

This section describes signals that connect to package pins. It includes pinout diagrams, recommended system connections, and detailed discussions of signals.

## 2.1 Device Pin Assignment

This section shows the pin assignments for MC9S08DZ60 Series MCUs in the available packages.

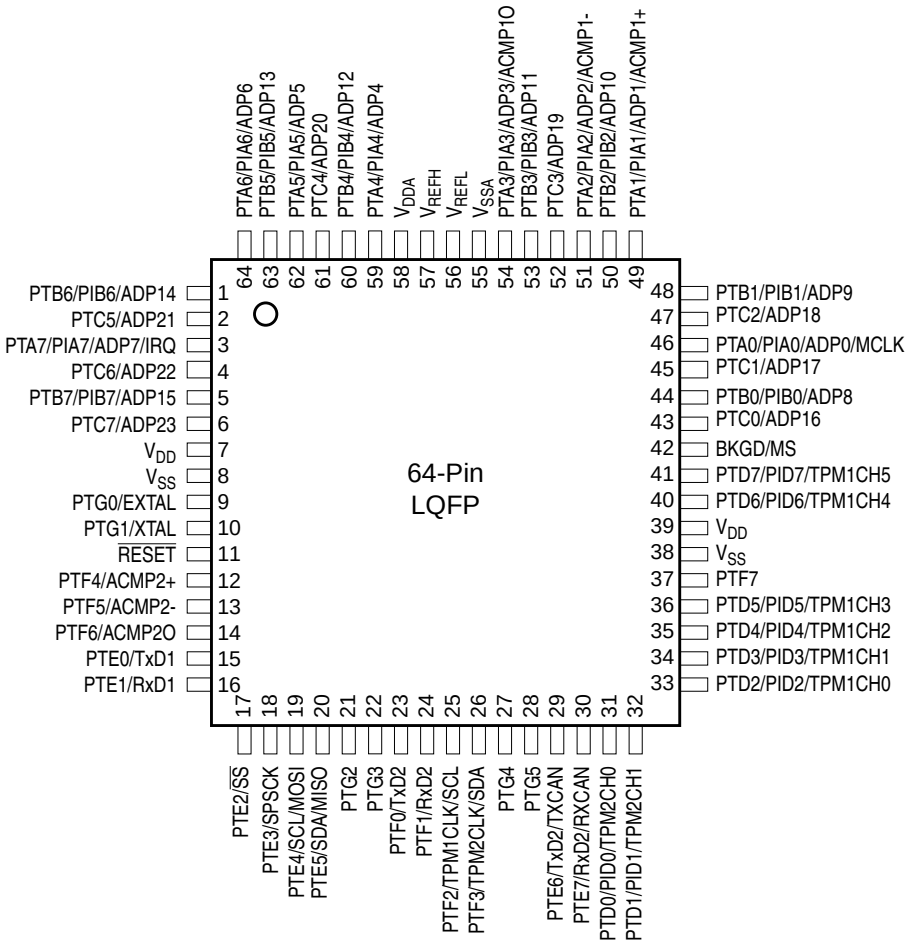
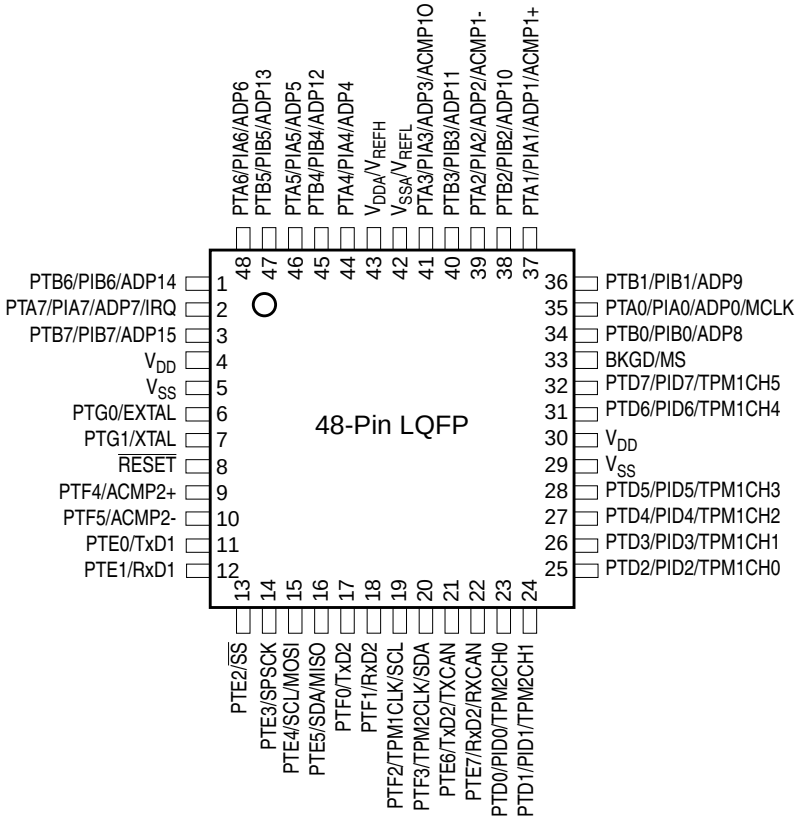
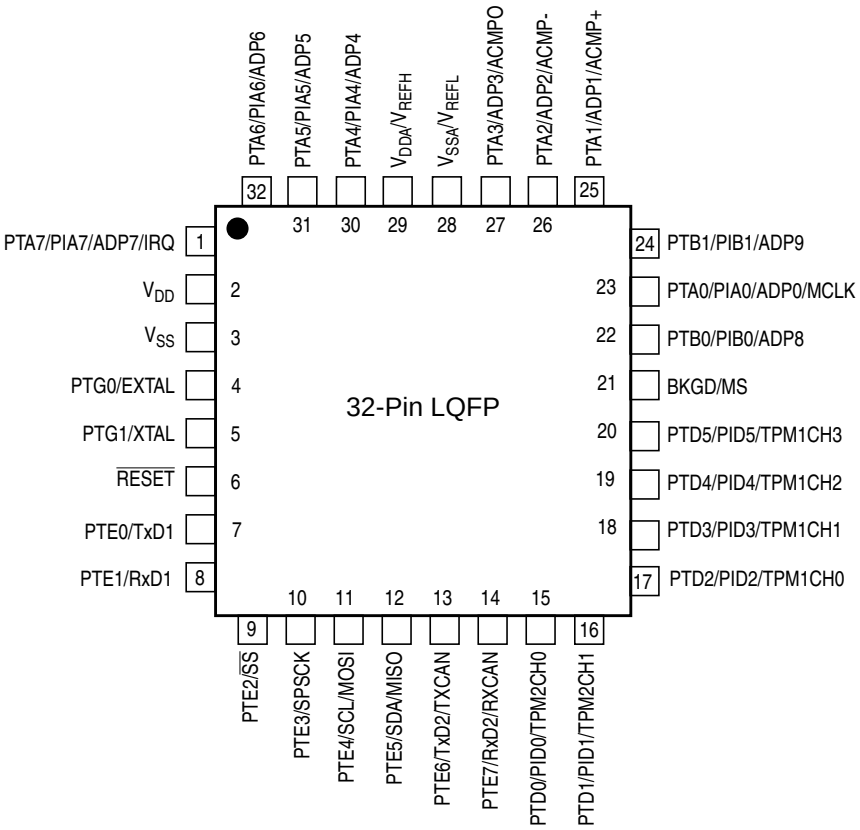



**Figure 2-1. 64-Pin LQFP**




$V_{REFH}$ and $V_{REFL}$ are internally connected to $V_{DDA}$ and $V_{SSA}$, respectively.

**Figure 2-2. 48-Pin LQFP**

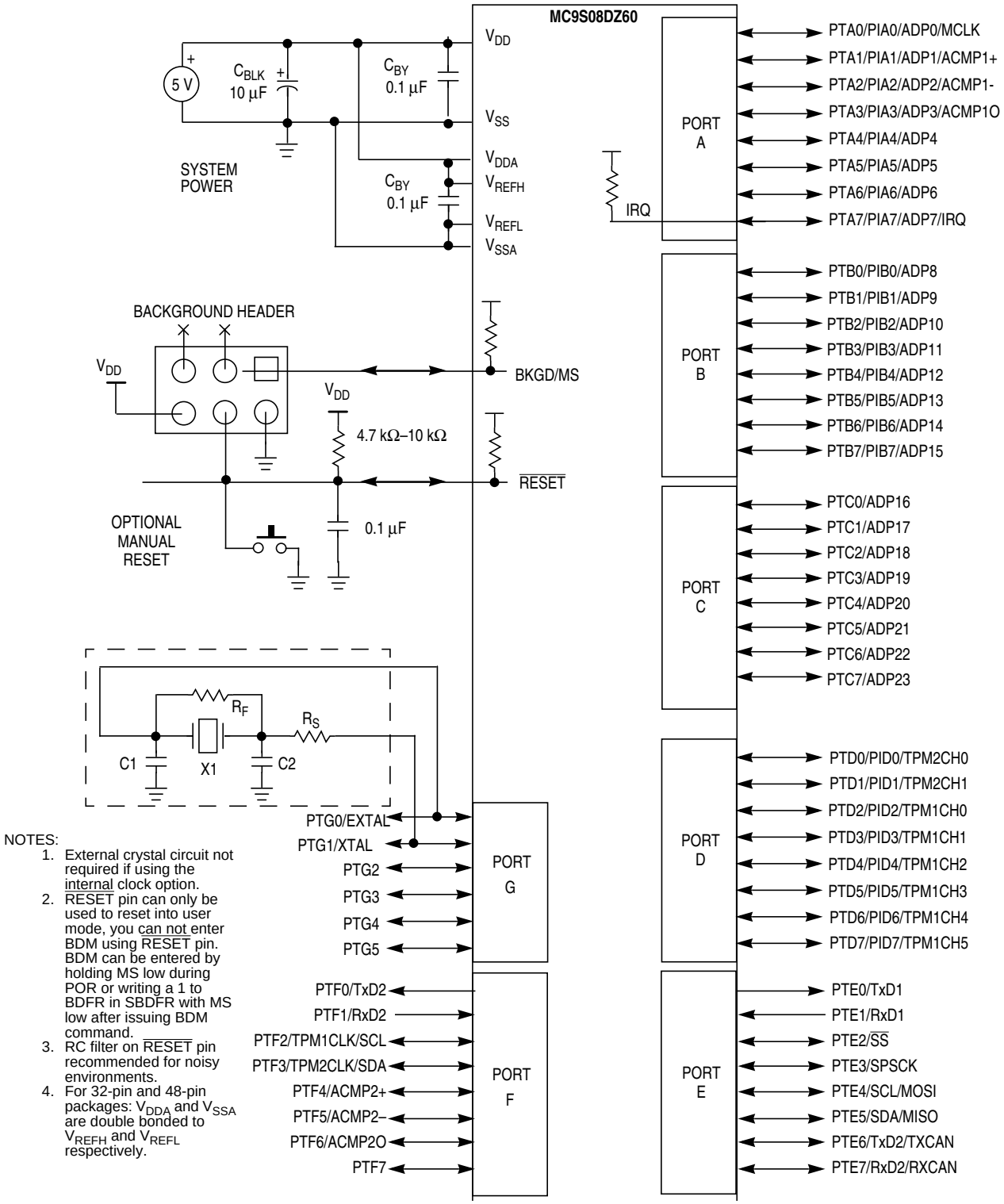



$V_{REFH}$ and $V_{REFL}$ are internally connected to $V_{DDA}$ and $V_{SSA}$, respectively.

**Figure 2-3. 32-Pin LQFP**

## 2.2 Recommended System Connections

Figure 2-4 shows pin connections that are common to MC9S08DZ60 Series application systems.




NOTES:
1. External crystal circuit not required if using the internal clock option.
2. RESET pin can only be used to reset into user mode, you can not enter BDM using RESET pin. BDM can be entered by holding MS low during POR or writing a 1 to BDFR in SBDFR with MS low after issuing BDM command.
3. RC filter on RESET pin recommended for noisy environments.
4. For 32-pin and 48-pin packages: $V_{DDA}$ and $V_{SSA}$ are double bonded to $V_{REFH}$ and $V_{REFL}$ respectively.

**Figure 2-4. Basic System Connections (Shown in 64-Pin Package)**

### 2.2.1 Power

$V_{DD}$ and $V_{SS}$ are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides regulated lower-voltage source to the CPU and other internal circuitry of the MCU.

Typically, application systems have two separate capacitors across the power pins. In this case, there should be a bulk electrolytic capacitor, such as a 10-µF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1-µF ceramic bypass capacitor located as near to the MCU power pins as practical to suppress high-frequency noise. The MC9S08DZ60 Series has two $V_{DD}$ pins except on the 32-pin package. Each pin must have a bypass capacitor for best noise suppression.

$V_{DDA}$ and $V_{SSA}$ are the analog power supply pins for the MCU. This voltage source supplies power to the ADC module. A 0.1-µF ceramic bypass capacitor should be located as near to the MCU power pins as practical to suppress high-frequency noise.

### 2.2.2 Oscillator

Immediately after reset, the MCU uses an internally generated clock provided by the multi-purpose clock generator (MCG) module. For more information on the MCG, see Chapter 8, “Multi-Purpose Clock Generator (S08MCGV1).”

The oscillator (XOSC) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Rather than a crystal or ceramic resonator, an external oscillator can be connected to the EXTAL input pin.

Refer to Figure 2-4 for the following discussion. $R_S$ (when used) and $R_F$ should be low-inductance resistors such as carbon composition resistors. Wire-wound resistors and some metal film resistors have too much inductance. C1 and C2 normally should be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

$R_F$ is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup; its value is not generally critical. Typical systems use 1 MΩ to 10 MΩ. Higher values are sensitive to humidity, and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5-pF to 25-pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to take into account printed circuit board (PCB) capacitance and MCU pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

### 2.2.3 $\overline{\text{RESET}}$

$\overline{\text{RESET}}$ is a dedicated pin with a pull-up device built in. It has input hysteresis, a high current output driver, and no output slew rate control. Internal power-on reset and low-voltage reset circuitry typically make external reset circuitry unnecessary. This pin is normally connected to the standard 6-pin background debug connector so a development system can directly reset the MCU system. If desired, a manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

Whenever any reset is initiated (whether from an external signal or from an internal system), the $\overline{\text{RESET}}$ pin is driven low for about 34 bus cycles. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the system reset status register (SRS).

### 2.2.4 Background / Mode Select (BKGD/MS)

While in reset, the BKGD/MS pin functions as a mode select pin. Immediately after reset rises, the pin functions as the background pin and can be used for background debug communication. While functioning as a background or mode select pin, the pin includes an internal pull-up device, input hysteresis, a standard output driver, and no output slew rate control.

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD low during the rising edge of reset which forces the MCU to active background mode.

The BKGD/MS pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock could be as fast as the bus clock rate, so there should never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD/MS pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speedup pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pull-up device play almost no role in determining rise and fall times on the BKGD/MS pin.

### 2.2.5 ADC Reference Pins ($V_{REFH}$, $V_{REFL}$)

The $V_{REFH}$ and $V_{REFL}$ pins are the voltage reference high and voltage reference low inputs, respectively, for the ADC module.

### 2.2.6 General-Purpose I/O and Peripheral Ports

The MC9S08DZ60 Series series of MCUs support up to 53 general-purpose I/O pins and 1 input-only pin, which are shared with on-chip peripheral functions (timers, serial I/O, ADC, MSCAN, etc.).

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pull-up device. Immediately after reset, all of these pins are configured as high-impedance general-purpose inputs with internal pull-up devices disabled.

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from port data registers even though the peripheral module controls the pin direction by controlling the enable for the pin's output buffer. For information about controlling these pins as general-purpose I/O pins, see Chapter 6, "Parallel Input/Output Control."

**NOTE**

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should either enable on-chip pull-up devices or change the direction of unused or non-bonded pins to outputs so they do not float.

## Table 2-1. Pin Availability by Package Pin-Count

| Pin Number | | | <-- Lowest Priority --> Highest | | | |
|---|---|---|---|---|---|---|
| 64 | 48 | 32 | Port Pin/Interrupt | | Alt 1 | Alt 2 |
| 1 | 1 | — | PTB6 | PIB6 | ADP14 | |
| 2 | — | — | PTC5 | | ADP21 | |
| 3 | 2 | 1 | PTA7 | PIA7 | ADP7 | IRQ |
| 4 | — | — | PTC6 | | ADP22 | |
| 5 | 3 | — | PTB7 | PIB7 | ADP15 | |
| 6 | — | — | PTC7 | | ADP23 | |
| 7 | 4 | 2 | | | | $V_{DD}$ |
| 8 | 5 | 3 | | | | $V_{SS}$ |
| 9 | 6 | 4 | PTG0 | | EXTAL | |
| 10 | 7 | 5 | PTG1 | | XTAL | |
| 11 | 8 | 6 | | | | $\overline{\text{RESET}}$ |
| 12 | 9 | — | PTF4 | | | ACMP2+ |
| 13 | 10 | — | PTF5 | | | ACMP2- |
| 14 | — | — | PTF6 | | | ACMP2O |
| 15 | 11 | 7 | PTE0 | | TxD1 | |
| 16 | 12 | 8 | PTE1[2] | | RxD1[2] | |
| 17 | 13 | 9 | PTE2 | | | $\overline{\text{SS}}$ |
| 18 | 14 | 10 | PTE3 | | | SPSCK |
| 19 | 15 | 11 | PTE4 | | SCL[3] | MOSI |
| 20 | 16 | 12 | PTE5 | | SDA[3] | MISO |
| 21 | — | — | PTG2 | | | |
| 22 | — | — | PTG3 | | | |
| 23 | 17 | — | PTF0 | | | TxD2[4] |
| 24 | 18 | — | PTF1 | | | RxD2[4] |
| 25 | 19 | — | PTF2 | | TPM1CLK | SCL[3] |
| 26 | 20 | — | PTF3 | | TPM2CLK | SDA[3] |
| 27 | — | — | PTG4 | | | |
| 28 | — | — | PTG5 | | | |
| 29 | 21 | 13 | PTE6 | | TxD2[4] | TXCAN |
| 30 | 22 | 14 | PTE7 | | RxD2[4] | RxCAN |
| 31 | 23 | 15 | PTD0 | PID0 | | TPM2CH0 |
| 32 | 24 | 16 | PTD1 | PID1 | | TPM2CH1 |
| 33 | 25 | 17 | PTD2 | PID2 | | TPM1CH0 |
| 34 | 26 | 18 | PTD3 | PID3 | | TPM1CH1 |
| 35 | 27 | 19 | PTD4 | PID4 | | TPM1CH2 |
| 36 | 28 | 20 | PTD5 | PID5 | | TPM1CH3 |
| 37 | — | — | PTF7 | | | |
| 38 | 29 | — | | | | $V_{SS}$ |
| 39 | 30 | — | | | | $V_{DD}$ |
| 40 | 31 | — | PTD6 | PID6 | | TPM1CH4 |
| 41 | 32 | — | PTD7 | PID7 | | TPM1CH5 |
| 42 | 33 | 21 | | | BKGD | MS |
| 43 | — | — | PTC0 | | ADP16 | |
| 44 | 34 | 22 | PTB0 | PIB0 | ADP8 | |
| 45 | — | — | PTC1 | | ADP17 | |
| 46 | 35 | 23 | PTA0 | PIA0 | ADP0 | MCLK |
| 47 | — | — | PTC2 | | ADP18 | |
| 48 | 36 | 24 | PTB1 | PIB1 | ADP9 | |
| 49 | 37 | 25 | PTA1 | PIA1 | ADP1[1] | ACMP1+[1] |
| 50 | 38 | — | PTB2 | PIB2 | ADP10 | |
| 51 | 39 | 26 | PTA2 | PIA2 | ADP2[1] | ACMP1-[1] |
| 52 | — | — | PTC3 | | ADP19 | |
| 53 | 40 | — | PTB3 | PIB3 | ADP11 | |
| 54 | 41 | 27 | PTA3 | PIA3 | ADP3 | ACMP1O |
| 55 | 42 | 28 | | | | $V_{SSA}$ |
| 56 | | | | | | $V_{REFL}$ |
| 57 | 43 | 29 | | | | $V_{REFH}$ |
| 58 | | | | | | $V_{DDA}$ |
| 59 | 44 | 30 | PTA4 | PIA4 | ADP4 | |
| 60 | 45 | — | PTB4 | PIB4 | ADP12 | |
| 61 | — | — | PTC4 | | ADP20 | |
| 62 | 46 | 31 | PTA5 | PIA5 | ADP5 | |
| 63 | 47 | — | PTB5 | PIB5 | ADP13 | |
| 64 | 48 | 32 | PTA6 | PIA6 | ADP6 | |

1. If both of these analog modules are enabled, they both will have access to the pin.
2. Pin does not contain a clamp diode to $V_{DD}$ and should not be driven above $V_{DD}$. The voltage measured on this pin when internal pull-up is enabled may be as low as $V_{DD}$ – 0.7 V. The internal gates connected to this pin are pulled to $V_{DD}$.
3. The IIC module pins can be repositioned using IICPS bit in the SOPT1 register. The default reset locations are on PTF2 and PTF3.
4. The SCI2 module pins can be repositioned using SCI2PS bit in the SOPT1 register. The default reset locations are on PTF0 and PTF1.

# Chapter 3
# Modes of Operation

## 3.1 Introduction

The operating modes of the MC9S08DZ60 Series are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

## 3.2 Features

- Active background mode for code development
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained
- Stop modes — System clocks are stopped and voltage regulator is in standby
  - Stop3 — All internal circuits are powered for fast recovery
  - Stop2 — Partial power down of internal circuits; RAM content is retained

## 3.3 Run Mode

This is the normal operating mode for the MC9S08DZ60 Series. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFE–0xFFFF after reset.

## 3.4 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the HCS08 core. The BDC, together with the on-chip debug module (DBG), provide the means for analyzing MCU operation during software development.

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low at the rising edge of reset
- When a BACKGROUND command is received through the BKGD/MS pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

After entering active background mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user application program.

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD/MS pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory-access-with-status commands
  - BDC register access commands
  - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the Flash program memory before the MCU is operated in run mode for the first time. When the MC9S08DZ60 Series is shipped from the Freescale Semiconductor factory, the Flash program memory is erased by default unless specifically noted so there is no program that could be executed in run mode until the Flash memory is initially programmed. The active background mode can also be used to erase and reprogram the Flash memory after it has been previously programmed.

For additional information about the active background mode, refer to the Development Support chapter.

## 3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available when the MCU is in wait mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

## 3.6 Stop Modes

One of two stop modes is entered upon execution of a STOP instruction when the STOPE bit in SOPT1 register is set. In both stop modes, all internal clocks are halted. The MCG module can be configured to leave the reference clocks running. See Chapter 8, “Multi-Purpose Clock Generator (S08MCGV1),” for more information.

Table 3-1 shows all of the control bits that affect stop mode selection and the mode selected under various conditions. The selected mode is entered following the execution of a STOP instruction.

**Table 3-1. Stop Mode Selection**

| STOPE | ENBDM [1] | LVDE | LVDSE | PPDC | Stop Mode |
|---|---|---|---|---|---|
| 0 | x | x | | x | Stop modes disabled; illegal opcode reset if STOP instruction executed |
| 1 | 1 | x | | x | Stop3 with BDM enabled [2] |
| 1 | 0 | Both bits must be 1 | | x | Stop3 with voltage regulator active |
| 1 | 0 | Either bit a 0 | | 0 | Stop3 |
| 1 | 0 | Either bit a 0 | | 1 | Stop2 |

1 ENBDM is located in the BDCSCR, which is only accessible through BDC commands, see Section 17.4.1.1, “BDC Status and Control Register (BDCSCR)”.

2 When in Stop3 mode with BDM enabled, The $S_{IDD}$ will be near $R_{IDD}$ levels because internal clocks are enabled.

### 3.6.1 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained.

Exit from stop3 is done by asserting $\overline{\text{RESET}}$ or an asynchronous interrupt pin. The asynchronous interrupt pins are IRQ, PIA0–PIA7, PIB0–PIB7, and PID0–PID7. Exit from stop3 can also be done by the low-voltage detect (LVD) reset, low-voltage warning (LVW) interrupt, ADC conversion complete interrupt, real-time clock (RTC) interrupt, MSCAN wake-up interrupt, or SCI receiver interrupt.

If stop3 is exited by means of the $\overline{\text{RESET}}$ pin, the MCU will be reset and operation will resume after fetching the reset vector. Exit by means of an interrupt will result in the MCU fetching the appropriate interrupt vector.

#### 3.6.1.1 LVD Enabled in Stop3 Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) at the time the CPU executes a STOP instruction, then the voltage regulator remains active during stop mode.

For the ADC to operate the LVD must be left enabled when entering stop3.

#### 3.6.1.2 Active BDM Enabled in Stop3 Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in Chapter 17, "Development Support." If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation.

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After entering background debug mode, all background commands are available.

### 3.6.2 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting $\overline{\text{RESET}}$. On 3M05C or older masksets only, exit from stop2 can also be performed by asserting PTA7/ADP7/IRQ.

**NOTE**

On 3M05C or older masksets only, PTA7/ADP7/IRQ is an active low wake-up and must be configured as an input prior to executing a STOP instruction to avoid an immediate exit from stop2. PTA7/ADP7/IRQ can be disabled as a wake-up if it is configured as a high driven output. For lowest power consumption in stop2, this pin should not be left open when configured as input (enable the internal pullup; or tie an external pullup/down device; or set pin as output).

In addition, the real-time counter (RTC) can wake the MCU from stop2, if enabled.

Upon wake-up from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset
- The LVD reset function is enabled and the MCU remains in the reset state if $V_{DD}$ is below the LVD trip point (low trip point selected due to POR)
- The CPU takes the reset vector

In addition to the above, upon waking up from stop2, the PPDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to PPDACK in SPMSC2.

To maintain I/O states for pins that were configured as general-purpose I/O before entering stop2, the user must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the pins will switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, the user must reconfigure the peripheral module that interfaces to the pin before writing to the PPDACK bit. If the peripheral module is not enabled before writing to PPDACK, the pins will be controlled by their associated port control registers when the I/O latches are opened.

### 3.6.3 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to Section 3.6.2, “Stop2 Mode” and Section 3.6.1, “Stop3 Mode” for specific information on system behavior in stop modes.

**Table 3-2. Stop Mode Behavior**

| Peripheral | Mode | |
|---|---|---|
| | Stop2 | Stop3 |
| CPU | Off | Standby |
| RAM | Standby | Standby |
| Flash/EEPROM | Off | Standby |
| Parallel Port Registers | Off | Standby |
| ACMP | Off | Off |
| ADC | Off | Optionally On[1] |
| IIC | Off | Standby |
| MCG | Off | Optionally On[2] |
| MSCAN | Off | Standby |
| RTC | Optionally On[3] | Optionally On[3] |
| SCI | Off | Standby |
| SPI | Off | Standby |
| TPM | Off | Standby |
| Voltage Regulator | Off | Optionally On[4] |
| XOSC | Off | Optionally On[5] |
| I/O Pins | States Held | States Held |
| BDM | Off[6] | Optionally On |
| LVD/LVW | Off[7] | Optionally On |

1 Requires the asynchronous ADC clock and LVD to be enabled, else in standby.
2 IRCLKEN and IREFSTEN set in MCGC1, else in standby.
3 Requires the RTC to be enabled, else in standby.
4 Requires the LVD or BDC to be enabled.

5 ERCLKEN and EREFSTEN set in MCGC2 for, else in standby. For high frequency range (RANGE in MCGC2 set) requires the LVD to also be enabled in stop3.

6 If ENBDM is set when entering stop2, the MCU will actually enter stop3.

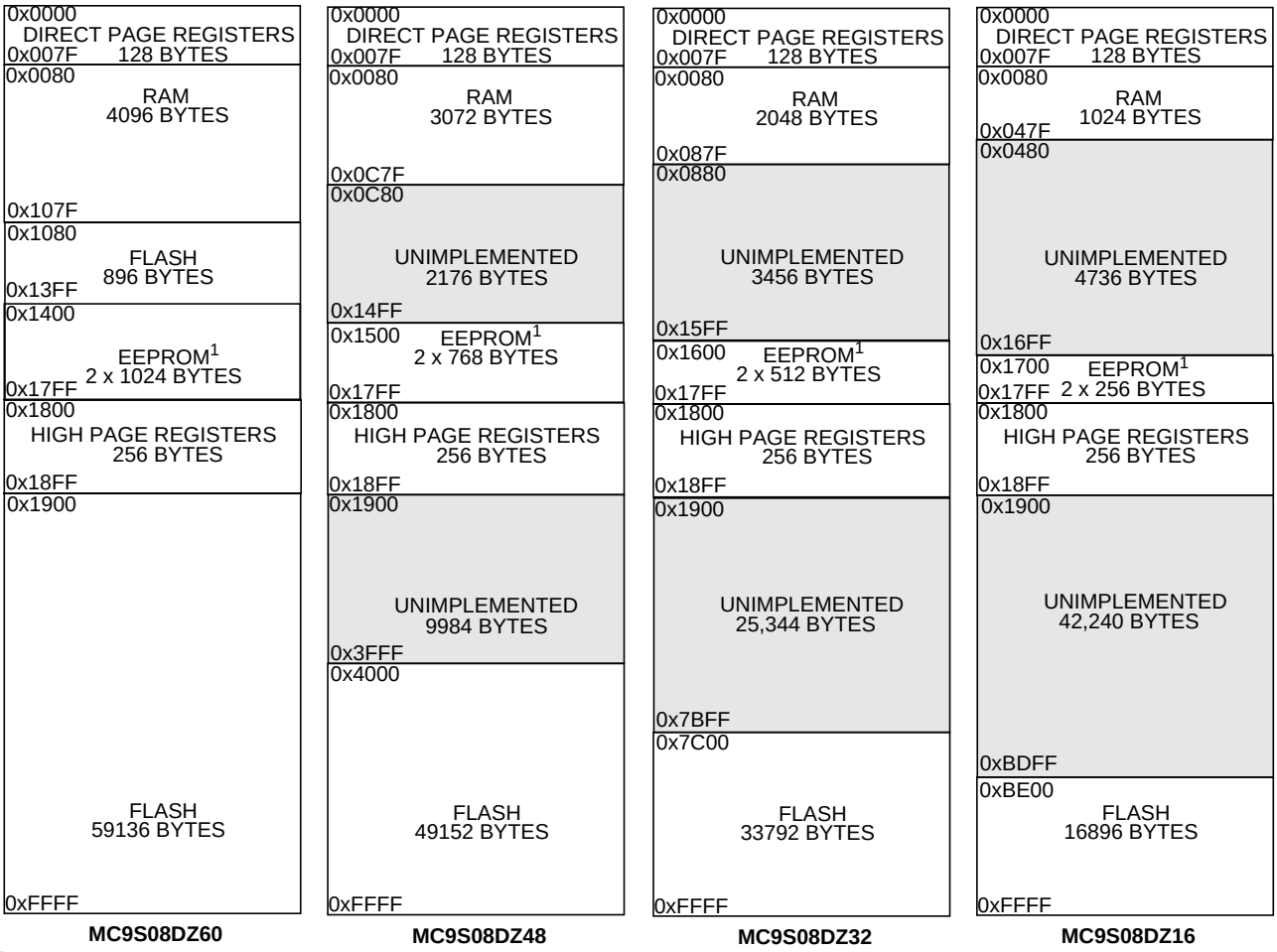
7 If LVDSE is set when entering stop2, the MCU will actually enter stop3.

# Chapter 4
# Memory

## 4.1 MC9S08DZ60 Series Memory Map

On-chip memory in the MC9S08DZ60 Series consists of RAM, EEPROM, and Flash program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x007F)
- High-page registers (0x1800 through 0x18FF)
- Nonvolatile registers (0xFFB0 through 0xFFBF)




1 EEPROM address range shows half the total EEPROM. See Section 4.5.10, “EEPROM Mapping” for more details.

**Figure 4-1. MC9S08DZ60 Memory Map**

## 4.2 Reset and Interrupt Vector Assignments

Table 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the MC9S08DZ60 Series equate file provided by Freescale Semiconductor.

**Table 4-1. Reset and Interrupt Vectors**

| Address (High/Low) | Vector | Vector Name |
|---|---|---|
| 0xFFC0:0xFFC1 | ACMP2 | Vacmp2 |
| 0xFFC2:0xFFC3 | ACMP1 | Vacmp1 |
| 0xFFC4:0xFFC5 | MSCAN Transmit | Vcantx |
| 0xFFC6:0xFFC7 | MSCAN Receive | Vcanrx |
| 0xFFC8:0xFFC9 | MSCAN errors | Vcanerr |
| 0xFFCA:0xFFCB | MSCAN wake up | Vcanwu |

**Table 4-1. Reset and Interrupt Vectors**

| Address (High/Low) | Vector | Vector Name |
|---|---|---|
| 0xFFCC:0xFFCD | RTC | Vrtc |
| 0xFFCE:0xFFCF | IIC | Viic |
| 0xFFD0:0xFFD1 | ADC Conversion | Vadc |
| 0xFFD2:0xFFD3 | Port A, Port B, Port D | Vport |
| 0xFFD4:0xFFD5 | SCI2 Transmit | Vsci2tx |
| 0xFFD6:0xFFD7 | SCI2 Receive | Vsci2rx |
| 0xFFD8:0xFFD9 | SCI2 Error | Vsci2err |
| 0xFFDA:0xFFDB | SCI1 Transmit | Vsci1tx |
| 0xFFDC:0xFFDD | SCI1 Receive | Vsci1rx |
| 0xFFDE:0xFFDF | SCI1 Error | Vsci1err |
| 0xFFE0:0xFFE1 | SPI | Vspi |
| 0xFFE2:0xFFE3 | TPM2 Overflow | Vtpm2ovf |
| 0xFFE4:0xFFE5 | TPM2 Channel 1 | Vtpm2ch1 |
| 0xFFE6:0xFFE7 | TPM2 Channel 0 | Vtpm2ch0 |
| 0xFFE8:0xFFE9 | TPM1 Overflow | Vtpm1ovf |
| 0xFFEA:0xFFEB | TPM1 Channel 5 | Vtpm1ch5 |
| 0xFFEC:0xFFED | TPM1 Channel 4 | Vtpm1ch4 |
| 0xFFEE:0xFFEF | TPM1 Channel 3 | Vtpm1ch3 |
| 0xFFF0:0xFFF1 | TPM1 Channel 2 | Vtpm1ch2 |
| 0xFFF2:0xFFF3 | TPM1 Channel 1 | Vtpm1ch1 |
| 0xFFF4:0xFFF5 | TPM1 Channel 0 | Vtpm1ch0 |
| 0xFFF6:0xFFF7 | MCG Loss of lock | Vlol |
| 0xFFF8:0xFFF9 | Low-Voltage Detect | Vlvd |
| 0xFFFA:0xFFFB | IRQ | Virq |
| 0xFFFC:0xFFFD | SWI | Vswi |
| 0xFFFE:0xFFFF | Reset | Vreset |

## 4.3 Register Addresses and Bit Assignments

The registers in the MC9S08DZ60 Series are divided into these groups:

- Direct-page registers are located in the first 128 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and RAM.
- The nonvolatile register area consists of a block of 16 locations in Flash memory at 0xFFB0–0xFFBF. Nonvolatile register locations include:
  - NVPROT and NVOPT are loaded into working registers at reset
  - An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory

  Because the nonvolatile register locations are Flash memory, they must be erased and programmed like other Flash memory locations.

Direct-page registers can be accessed with efficient direct addressing mode instructions. Bit manipulation instructions can be used to access any bit in any direct-page register. Table 4-2 is a summary of all user-accessible direct-page registers and control bits.

The direct page registers in Table 4-2 can use the more efficient direct addressing mode, which requires only the lower byte of the address. Because of this, the lower byte of the address in column one is shown in bold text. In Table 4-3 and Table 4-5, the whole address in column one is shown in bold. In Table 4-2, Table 4-3, and Table 4-5, the register names in column two are shown in bold to set them apart from the bit names to the right. Cells that are not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s.

**Table 4-2. Direct-Page Register Summary (Sheet 1 of 3)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | PTAD | PTAD7 | PTAD6 | PTAD5 | PTAD4 | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| 0x0001 | PTADD | PTADD7 | PTADD6 | PTADD5 | PTADD4 | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| 0x0002 | PTBD | PTBD7 | PTBD6 | PTBD5 | PTBD4 | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| 0x0003 | PTBDD | PTBDD7 | PTBDD6 | PTBDD5 | PTBDD4 | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| 0x0004 | PTCD | PTCD7 | PTCD6 | PTCD5 | PTCD4 | PTCD3 | PTCD2 | PTCD1 | PTCD0 |
| 0x0005 | PTCDD | PTCDD7 | PTCDD6 | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| 0x0006 | PTDD | PTDD7 | PTDD6 | PTDD5 | PTDD4 | PTDD3 | PTDD2 | PTDD1 | PTDD0 |
| 0x0007 | PTDDD | PTDDD7 | PTDDD6 | PTDDD5 | PTDDD4 | PTDDD3 | PTDDD2 | PTDDD1 | PTDDD0 |
| 0x0008 | PTED | PTED7 | PTED6 | PTED5 | PTED4 | PTED3 | PTED2 | PTED1 | PTED0 |
| 0x0009 | PTEDD | PTEDD7 | PTEDD6 | PTEDD5 | PTEDD4 | PTEDD3 | PTEDD2 | PTEDD1 | PTEDD0 |
| 0x000A | PTFD | PTFD7 | PTFD6 | PTFD5 | PTFD4 | PTFD3 | PTFD2 | PTFD1 | PTFD0 |
| 0x000B | PTFDD | PTFDD7 | PTFDD6 | PTFDD5 | PTFDD4 | PTFDD3 | PTFDD2 | PTFDD1 | PTFDD0 |
| 0x000C | PTGD | 0 | 0 | PTGD5 | PTGD4 | PTGD3 | PTGD2 | PTGD1 | PTGD0 |
| 0x000D | PTGDD | 0 | 0 | PTGDD5 | PTGDD4 | PTGDD3 | PTGDD2 | PTGDD1 | PTGDD0 |
| 0x000E | ACMP1SC | ACME | ACBGS | ACF | ACIE | ACO | ACOPE | ACMOD1 | ACMOD0 |
| 0x000F | ACMP2SC | ACME | ACBGS | ACF | ACIE | ACO | ACOPE | ACMOD1 | ACMOD0 |
| 0x0010 | ADCSC1 | COCO | AIEN | ADCO | ADCH | | | | |
| 0x0011 | ADCSC2 | ADACT | ADTRG | ACFE | ACFGT | 0 | 0 | — | — |
| 0x0012 | ADCRH | 0 | 0 | 0 | 0 | ADR11 | ADR10 | ADR9 | ADR8 |
| 0x0013 | ADCRL | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |
| 0x0014 | ADCCVH | 0 | 0 | 0 | 0 | ADCV11 | ADCV10 | ADCV9 | ADCV8 |
| 0x0015 | ADCCVL | ADCV7 | ADCV6 | ADCV5 | ADCV4 | ADCV3 | ADCV2 | ADCV1 | ADCV0 |
| 0x0016 | ADCCFG | ADLPC | ADIV | | ADLSMP | MODE | | ADICLK | |
| 0x0017 | APCTL1 | ADPC7 | ADPC6 | ADPC5 | ADPC4 | ADPC3 | ADPC2 | ADPC1 | ADPC0 |
| 0x0018 | APCTL2 | ADPC15 | ADPC14 | ADPC13 | ADPC12 | ADPC11 | ADPC10 | ADPC9 | ADPC8 |
| 0x0019 | APCTL3 | ADPC23 | ADPC22 | ADPC21 | ADPC20 | ADPC19 | ADPC18 | ADPC17 | ADPC16 |
| 0x001A–<br>0x001B | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x001C | IRQSC | 0 | IRQPDD | IRQEDG | IRQPE | IRQF | IRQACK | IRQIE | IRQMOD |
| 0x001D–<br>0x001F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x0020 | TPM1SC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x0021 | TPM1CNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0022 | TPM1CNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0023 | TPM1MODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0024 | TPM1MODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0025 | TPM1C0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x0026 | TPM1C0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0027 | TPM1C0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

**Table 4-2. Direct-Page Register Summary (Sheet 2 of 3)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0028 | TPM1C1SC | CH1F | CH1IE | MS1B | MS1A | ELS1B | ELS1A | 0 | 0 |
| 0x0029 | TPM1C1VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x002A | TPM1C1VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x002B | TPM1C2SC | CH2F | CH2IE | MS2B | MS2A | ELS2B | ELS2A | 0 | 0 |
| 0x002C | TPM1C2VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x002D | TPM1C2VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x002E | TPM1C3SC | CH3F | CH3IE | MS3B | MS3A | ELS3B | ELS3A | 0 | 0 |
| 0x002F | TPM1C3VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0030 | TPM1C3VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0031 | TPM1C4SC | CH4F | CH4IE | MS4B | MS4A | ELS4B | ELS4A | 0 | 0 |
| 0x0032 | TPM1C4VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0033 | TPM1C4VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0034 | TPM1C5SC | CH5F | CH5IE | MS5B | MS5A | ELS5B | ELS5A | 0 | 0 |
| 0x0035 | TPM1C5VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0036 | TPM1C5VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0037 | Reserved | — | — | — | — | — | — | — | — |
| 0x0038 | SCI1BDH | LBKDIE | RXEDGIE | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| 0x0039 | SCI1BDL | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| 0x003A | SCI1C1 | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| 0x003B | SCI1C2 | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| 0x003C | SCI1S1 | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| 0x003D | SCI1S2 | LBKDIF | RXEDGIF | 0 | RXINV | RWUID | BRK13 | LBKDE | RAF |
| 0x003E | SCI1C3 | R8 | T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE |
| 0x003F | SCI1D | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0040 | SCI2BDH | LBKDIE | RXEDGIE | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| 0x0041 | SCI2BDL | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| 0x0042 | SCI2C1 | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| 0x0043 | SCI2C2 | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| 0x0044 | SCI2S1 | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| 0x0045 | SCI2S2 | LBKDIF | RXEDGIF | 0 | RXINV | RWUID | BRK13 | LBKDE | RAF |
| 0x0046 | SCI2C3 | R8 | T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE |
| 0x0047 | SCI2D | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0048 | MCGC1 | CLKS | | RDIV | | | IREFS | IRCLKEN | IREFSTEN |
| 0x0049 | MCGC2 | BDIV | | RANGE | HGO | LP | EREFS | ERCLKEN | EREFSTEN |
| 0x004A | MCGTRM | TRIM | | | | | | | |
| 0x004B | MCGSC | LOLS | LOCK | PLLST | IREFST | CLKST | | OSCINIT | FTRIM |
| 0x004C | MCGC3 | LOLIE | PLLS | CME | 0 | VDIV | | | |
| 0x004D–0x004F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |

**Table 4-2. Direct-Page Register Summary (Sheet 3 of 3)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0050 | **SPIC1** | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| 0x0051 | **SPIC2** | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| 0x0052 | **SPIBR** | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| 0x0053 | **SPIS** | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| 0x0054 | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0055 | **SPID** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0056–<br>0x0057 | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x0058 | **IICA** | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | 0 |
| 0x0059 | **IICF** | MULT | | ICR | | | | | |
| 0x005A | **IICC1** | IICEN | IICIE | MST | TX | TXAK | RSTA | 0 | 0 |
| 0x005B | **IICS** | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| 0x005C | **IICD** | DATA | | | | | | | |
| 0x005D | **IICC2** | GCAEN | ADEXT | 0 | 0 | 0 | AD10 | AD9 | AD8 |
| 0x005E–<br>0x005F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x0060 | **TPM2SC** | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x0061 | **TPM2CNTH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0062 | **TPM2CNTL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0063 | **TPM2MODH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0064 | **TPM2MODL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0065 | **TPM2C0SC** | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x0066 | **TPM2C0VH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0067 | **TPM2C0VL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0068 | **TPM2C1SC** | CH1F | CH1IE | MS1B | MS1A | ELS1B | ELS1A | 0 | 0 |
| 0x0069 | **TPM2C1VH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x006A | **TPM2C1VL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x006B | Reserved | — | — | — | — | — | — | — | — |
| 0x006C | **RTCSC** | RTIF | RTCLKS | | RTIE | RTCPS | | | |
| 0x006D | **RTCCNT** | RTCCNT | | | | | | | |
| 0x006E | **RTCMOD** | RTCMOD | | | | | | | |
| 0x006F | Reserved | — | — | — | — | — | — | — | — |
| 0x0070–<br>0x007F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |

High-page registers, shown in Table 4-3, are accessed much less often than other I/O and control registers so they have been located outside the direct addressable memory space, starting at 0x1800.

**Table 4-3. High-Page Register Summary (Sheet 1 of 3)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1800 | SRS | POR | PIN | COP | ILOP | ILAD | LOCS | LVD | 0 |
| 0x1801 | SBDFR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BDFR |
| 0x1802 | SOPT1 | COPT | | STOPE | SCI2PS | IICPS | 0 | 0 | 0 |
| 0x1803 | SOPT2 | COPCLKS | COPW | 0 | ADHTS | 0 | MCSEL | | |
| 0x1804 – 0x1805 | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| 0x1806 | SDIDH | — | — | — | — | ID11 | ID10 | ID9 | ID8 |
| 0x1807 | SDIDL | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 0x1808 | Reserved | — | — | — | — | — | — | — | — |
| 0x1809 | SPMSC1 | LVWF | LVWACK | LVWIE | LVDRE | LVDSE | LVDE | 0 | BGBE |
| 0x180A | SPMSC2 | 0 | 0 | LVDV | LVWV | PPDF | PPDACK | 0 | PPDC |
| 0x180B– 0x180F | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| 0x1810 | DBGCAH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1811 | DBGCAL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1812 | DBGCBH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1813 | DBGCBL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1814 | DBGFH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1815 | DBGFL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1816 | DBGC | DBGEN | ARM | TAG | BRKEN | RWA | RWAEN | RWB | RWBEN |
| 0x1817 | DBGT | TRGSEL | BEGIN | 0 | 0 | TRG3 | TRG2 | TRG1 | TRG0 |
| 0x1818 | DBGS | AF | BF | ARMF | 0 | CNT3 | CNT2 | CNT1 | CNT0 |
| 0x1819– 0x181F | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| 0x1820 | FCDIV | DIVLD | PRDIV8 | DIV | | | | | |
| 0x1821 | FOPT | KEYEN | FNORED | EPGMOD | 0 | 0 | 0 | SEC | |
| 0x1822 | Reserved | — | — | — | — | — | — | — | — |
| 0x1823 | FCNFG | 0 | EPGSEL | KEYACC | Reserved[1] | 0 | 0 | 0 | 1 |
| 0x1824 | FPROT | EPS | | FPS | | | | | |
| 0x1825 | FSTAT | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| 0x1826 | FCMD | FCMD | | | | | | | |
| 0x1827– 0x183F | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| 0x1840 | PTAPE | PTAPE7 | PTAPE6 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| 0x1841 | PTASE | PTASE7 | PTASE6 | PTASE5 | PTASE4 | PTASE3 | PTASE2 | PTASE1 | PTASE0 |
| 0x1842 | PTADS | PTADS7 | PTADS6 | PTADS5 | PTADS4 | PTADS3 | PTADS2 | PTADS1 | PTADS0 |
| 0x1843 | Reserved | — | — | — | — | — | — | — | — |
| 0x1844 | PTASC | 0 | 0 | 0 | 0 | PTAIF | PTAACK | PTAIE | PTAMOD |
| 0x1845 | PTAPS | PTAPS7 | PTAPS6 | PTAPS5 | PTAPS4 | PTAPS3 | PTAPS2 | PTAPS1 | PTAPS0 |
| 0x1846 | PTAES | PTAES7 | PTAES6 | PTAES5 | PTAES4 | PTAES3 | PTAES2 | PTAES1 | PTAES0 |

**Table 4-3. High-Page Register Summary (Sheet 2 of 3)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1847 | Reserved | — | — | — | — | — | — | — | — |
| 0x1848 | **PTBPE** | PTBPE7 | PTBPE6 | PTBPE5 | PTBPE4 | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| 0x1849 | **PTBSE** | PTBSE7 | PTBSE6 | PTBSE5 | PTBSE4 | PTBSE3 | PTBSE2 | PTBSE1 | PTBSE0 |
| 0x184A | **PTBDS** | PTBDS7 | PTBDS6 | PTBDS5 | PTBDS4 | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| 0x184B | Reserved | — | — | — | — | — | — | — | — |
| 0x184C | **PTBSC** | 0 | 0 | 0 | 0 | PTBIF | PTBACK | PTBIE | PTBMOD |
| 0x184D | **PTBPS** | PTBPS7 | PTBPS6 | PTBPS5 | PTBPS4 | PTBPS3 | PTBPS2 | PTBPS1 | PTBPS0 |
| 0x184E | **PTBES** | PTBES7 | PTBES6 | PTBES5 | PTBES4 | PTBES3 | PTBES2 | PTBES1 | PTBES0 |
| 0x184F | Reserved | — | — | — | — | — | — | — | — |
| 0x1850 | **PTCPE** | PTCPE7 | PTCPE6 | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| 0x1851 | **PTCSE** | PTCSE7 | PTCSE6 | PTCSE5 | PTCSE4 | PTCSE3 | PTCSE2 | PTCSE1 | PTCSE0 |
| 0x1852 | **PTCDS** | PTCDS7 | PTCDS6 | PTCDS5 | PTCDS4 | PTCDS3 | PTCDS2 | PTCDS1 | PTCDS0 |
| 0x1853–<br>0x1857 | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1858 | **PTDPE** | PTDPE7 | PTDPE6 | PTDPE5 | PTDPE4 | PTDPE3 | PTDPE2 | PTDPE1 | PTDPE0 |
| 0x1859 | **PTDSE** | PTDSE7 | PTDSE6 | PTDSE5 | PTDSE4 | PTDSE3 | PTDSE2 | PTDSE1 | PTDSE0 |
| 0x185A | **PTDDS** | PTDDS7 | PTDDS6 | PTDDS5 | PTDDS4 | PTDDS3 | PTDDS2 | PTDDS1 | PTDDS0 |
| 0x185B | Reserved | — | — | — | — | — | — | — | — |
| 0x185C | **PTDSC** | 0 | 0 | 0 | 0 | PTDIF | PTDACK | PTDIE | PTDMOD |
| 0x185D | **PTDPS** | PTDPS7 | PTDPS6 | PTDPS5 | PTDPS4 | PTDPS3 | PTDPS2 | PTDPS1 | PTDPS0 |
| 0x185E | **PTDES** | PTDES7 | PTDES6 | PTDES5 | PTDES4 | PTDES3 | PTDES2 | PTDES1 | PTDES0 |
| 0x185F | Reserved | — | — | — | — | — | — | — | — |
| 0x1860 | **PTEPE** | PTEPE7 | PTEPE6 | PTEPE5 | PTEPE4 | PTEPE3 | PTEPE2 | PTEPE1 | PTEPE0 |
| 0x1861 | **PTESE** | PTESE7 | PTESE6 | PTESE5 | PTESE4 | PTESE3 | PTESE2 | PTESE1 | PTESE0 |
| 0x1862 | **PTEDS** | PTEDS7 | PTEDS6 | PTEDS5 | PTEDS4 | PTEDS3 | PTEDS2 | PTEDS1 | PTEDS0 |
| 0x1863–<br>0x1867 | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1868 | **PTFPE** | PTFPE7 | PTFPE6 | PTFPE5 | PTFPE4 | PTFPE3 | PTFPE2 | PTFPE1 | PTFPE0 |
| 0x1869 | **PTFSE** | PTFSE7 | PTFSE6 | PTFSE5 | PTFSE4 | PTFSE3 | PTFSE2 | PTFSE1 | PTFSE0 |
| 0x186A | **PTFDS** | PTFDS7 | PTFDS6 | PTFDS5 | PTFDS4 | PTFDS3 | PTFDS2 | PTFDS1 | PTFDS0 |
| 0x186B–<br>0x186F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1870 | **PTGPE** | 0 | 0 | PTGPE5 | PTGPE4 | PTGPE3 | PTGPE2 | PTGPE1 | PTGPE0 |
| 0x1871 | **PTGSE** | 0 | 0 | PTGSE5 | PTGSE4 | PTGSE3 | PTGSE2 | PTGSE1 | PTGSE0 |
| 0x1872 | **PTGDS** | 0 | 0 | PTGDS5 | PTGDS4 | PTGDS3 | PTGDS2 | PTGDS1 | PTGDS0 |
| 0x1873–<br>0x187F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1880 | **CANCTL0** | RXFRM | RXACT | CSWAI | SYNCH | TIME | WUPE | SLPRQ | INITRQ |
| 0x1881 | **CANCTL1** | CANE | CLKSRC | LOOPB | LISTEN | BORM | WUPM | SLPAK | INITAK |
| 0x1882 | **CANBTR0** | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |

**Table 4-3. High-Page Register Summary (Sheet 3 of 3)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1883 | CANBTR1 | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| 0x1884 | CANRFLG | WUPIF | CSCIF | RSTAT1 | RSTAT0 | TSTAT1 | TSTAT0 | OVRIF | RXF |
| 0x1885 | CANRIER | WUPIE | CSCIE | RSTATE1 | RSTATE0 | TSTATE1 | TSTATE0 | OVRIE | RXFIE |
| 0x1886 | CANTFLG | 0 | 0 | 0 | 0 | 0 | TXE2 | TXE1 | TXE0 |
| 0x1887 | CANTIER | 0 | 0 | 0 | 0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| 0x1888 | CANTARQ | 0 | 0 | 0 | 0 | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 |
| 0x1889 | CANTAAK | 0 | 0 | 0 | 0 | 0 | ABTAK2 | ABTAK1 | ABTAK0 |
| 0x188A | CANTBSEL | 0 | 0 | 0 | 0 | 0 | TX2 | TX1 | TX0 |
| 0x188B | CANIDAC | 0 | 0 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| 0x188C | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x188D | CANMISC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOHOLD |
| 0x188E | CANRXERR | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| 0x188F | CANTXERR | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| 0x1890 – 0x1893 | CANIDAR0 – CANIDAR3 | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| 0x1894 – 0x1897 | CANIDMR0 – CANIDMR3 | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| 0x1898 – 0x189B | CANIDAR4 – CANIDAR7 | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| 0x189C– 0x189F | CANIDMR4 – CANIDMR7 | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| 0x18BE | CANTTSRH | TSR15 | TSR14 | TSR13 | TSR12 | TSR11 | TSR10 | TSR9 | TSR8 |
| 0x18BF | CANTTSRL | TSR7 | TSR6 | TSR5 | TSR4 | TSR3 | TSR2 | TSR1 | TSR0 |
| 0x18C0– 0x18FF | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |

1 This bit is reserved. User must write a 1 to this bit. Failing to do so may result in unexpected behavior.

Figure 4-4 shows the structure of receive and transmit buffers for extended identifier mapping. These registers vary depending on whether standard or extended mapping is selected. See Chapter 12, “Freescale Controller Area Network (S08MSCANV1),” for details on extended and standard identifier mapping.

**Table 4-4. MSCAN Foreground Receive and Transmit Buffer Layouts — Extended Mapping Shown**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x18A0 | CANRIDR0 | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| 0x18A1 | CANRIDR1 | ID20 | ID19 | ID18 | SRR(1) | IDE(1) | ID17 | ID16 | ID15 |
| 0x18A2 | CANRIDR2 | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| 0x18A3 | CANRIDR3 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR2 |
| 0x18A4 – 0x18AB | CANRDSR0 – CANRDSR7 | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| 0x18AC | CANRDLR | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 |
| 0x18AD | Reserved | — | — | — | — | — | — | — | — |
| 0x18AE | CANRTSRH | TSR15 | TSR14 | TSR13 | TSR12 | TSR11 | TSR10 | TSR9 | TSR8 |

**Table 4-4. MSCAN Foreground Receive and Transmit Buffer Layouts — Extended Mapping Shown**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **0x18AF** | **CANRTSRL** | TSR7 | TSR6 | TSR5 | TSR4 | TSR3 | TSR2 | TSR1 | TSR0 |
| **0x18B0** | **CANTIDR0** | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| **0x18B1** | **CANTIDR1** | ID2 | ID1 | ID0 | RTR | IDE | — | — | — |
| **0x18B2** | **CANTIDR2** | — | — | — | — | — | — | — | — |
| **0x18B3** | **CANTIDR3** | — | — | — | — | — | — | — | — |
| **0x18B4 – 0x18BB** | **CANTDSR0 – CANTDSR7** | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| **0x18BC** | **CANTDLR** | — | — | — | — | DLC3 | DLC2 | DLC1 | DLC0 |
| **0x18BD** | **CANTTBPR** | PRIO7 | PRIO6 | PRIO5 | PRIO4 | PRIO3 | PRIO2 | PRIO1 | PRIO0 |

1 SRR and IDE are both 1s.

2 The position of RTR differs between extended and standard identifier mapping.

Nonvolatile Flash registers, shown in Table 4-5, are located in the Flash memory. These registers include an 8-byte backdoor key, NVBACKKEY, which can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the nonvolatile register area of the Flash memory are transferred into corresponding FPROT and FOPT working registers in the high-page registers to control security and block protection options.

**Table 4-5. Nonvolatile Register Summary**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| **0xFFAE** | Reserved for storage of FTRIM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FTRIM |
| **0xFFAF** | Res. for storage of MCGTRM | TRIM | | | | | | | |
| **0xFFB0–0xFFB7** | **NVBACKKEY** | 8-Byte Comparison Key | | | | | | | |
| **0xFFB8–0xFFBC** | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| **0xFFBD** | **NVPROT** | EPS | | FPS | | | | | |
| **0xFFBE** | Reserved | — | — | — | — | — | — | — | — |
| **0xFFBF** | **NVOPT** | KEYEN | FNORED | EPGMOD | 0 | 0 | 0 | SEC | |

Provided the key enable (KEYEN) bit is 1, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory. (A security key cannot be entered directly through background debug commands.) This security key can be disabled completely by programming the KEYEN bit to 0. If the security key is disabled, the only way to disengage security is by mass erasing the Flash if needed (normally through the background debug interface) and verifying that Flash is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC) to the unsecured state (1:0).

## 4.4 RAM

The MC9S08DZ60 Series includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data while the MCU is in low-power wait, stop2, or stop3 mode. At power-on the contents of RAM are uninitialized. RAM data is unaffected by any reset if the supply voltage does not drop below the minimum value for RAM retention ($V_{RAM}$).

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08DZ60 Series, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor equate file).

```
LDHX    #RamLast+1    ;point one past RAM
TXS                   ;SP<-(H:X-1)
```

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or code executing from non-secure memory. See Section 4.5.9, “Security”, for a detailed description of the security feature.

## 4.5 Flash and EEPROM

MC9S08DZ60 Series devices include Flash and EEPROM memory intended primarily for program and data storage. In-circuit programming allows the operating program and data to be loaded into Flash and EEPROM, respectively, after final assembly of the application product. It is possible to program the arrays through the single-wire background debug interface. Because no special voltages are needed for erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to the *HCS08 Family Reference Manual, Volume I*, Freescale Semiconductor document order number HCS08RMv1.

### 4.5.1 Features

Features of the Flash and EEPROM memory include:

- Array size (see Table 1-1 for exact array sizes)
- Flash sector size: 768 bytes
- EEPROM sector size: selectable 4-byte or 8-byte sector mapping operation
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection and vector redirection
- Security feature for Flash, EEPROM, and RAM

- Burst programming capability
- Sector erase abort

### 4.5.2 Program and Erase Times

Before any program or erase command can be accepted, the Flash and EEPROM clock divider register (FCDIV) must be written to set the internal clock for the Flash and EEPROM module to a frequency ($f_{FCLK}$) between 150 kHz and 200 kHz (see Section 4.5.11.1, “Flash and EEPROM Clock Divider Register (FCDIV)”). This register can be written only once, so normally this write is performed during reset initialization. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{FCLK}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses is used by the command processor to complete a program or erase command.

Table 4-6 shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK ($f_{FCLK}$). The time for one cycle of FCLK is $t_{FCLK} = 1/f_{FCLK}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{FCLK} = 5$ µs. Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

**Table 4-6. Program and Erase Times**

| Parameter | Cycles of FCLK | Time if FCLK = 200 kHz |
|---|---|---|
| Byte program | 9 | 45 µs |
| Burst program | 4 | 20 µs[1] |
| Sector erase | 4000 | 20 ms |
| Mass erase | 20,000 | 100 ms |
| Sector erase abort | 4 | 20 µs[1] |

1 Excluding start/end overhead

### 4.5.3 Program and Erase Command Execution

The FCDIV register must be initialized after any reset and any error flag is cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the Flash or EEPROM array. The address and data information from this write is latched into the Flash and EEPROM interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For sector erase commands, the address can be any address in the sector of Flash or EEPROM to be erased. For mass erase and blank check commands, the address can be any address in the Flash or EEPROM memory. Flash and EEPROM erase independently of each other.

**NOTE**

Before programming a particular byte in the Flash or EEPROM, the sector in which that particular byte resides must be erased by a mass or sector erase operation. Reprogramming bits in an already programmed byte without first performing an erase operation may disturb data stored in the Flash or EEPROM memory.

2. Write the command code for the desired command to FCMD. The six valid commands are blank check (0x05), byte program (0x20), burst program (0x25), sector erase (0x40), mass erase[1] (0x41), and sector erase abort (0x47). The command code is latched into the command buffer.
3. Write a 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

   A partial command sequence can be aborted manually by writing a 0 to FCBEF any time after the write to the memory array and before writing the 1 that clears FCBEF and launches the complete command. Aborting a command in this way sets the FACCERR access error flag which must be cleared before starting a new command.

   A strictly monitored procedure must be obeyed or the command will not be accepted. This minimizes the possibility of any unintended changes to the memory contents. The command complete flag (FCCF) indicates when a command is complete. The command sequence must be completed by clearing FCBEF to launch the command. Figure 4-2 is a flowchart for executing all of the commands except for burst programming and sector erase abort.
4. Wait until the FCCF bit in FSTAT is set. As soon as FCCF= 1, the operation has completed successfully.

1. A mass erase is possible only when the Flash block is fully unprotected.

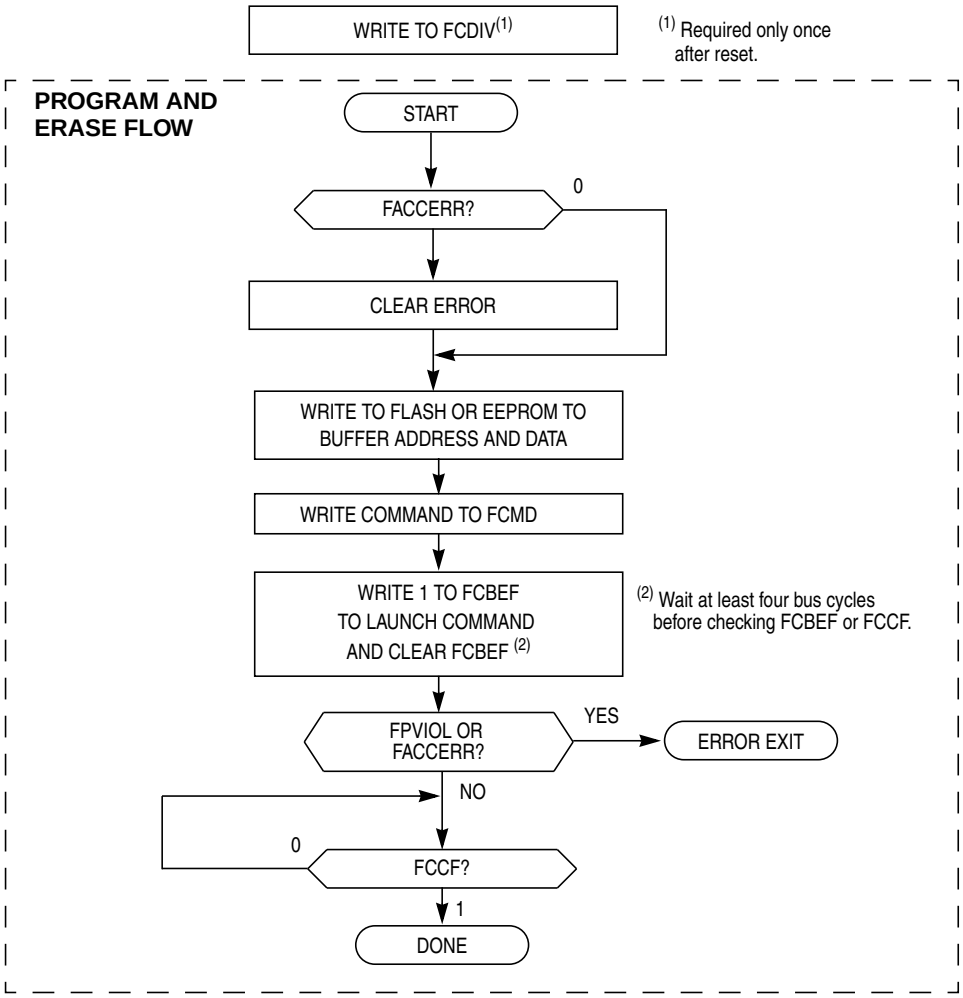



**Figure 4-2. Program and Erase Flowchart**

## 4.5.4 Burst Program Execution

The burst program command is used to program sequential bytes of data in less time than would be required using the standard program command. This is possible because the high voltage to the Flash array does not need to be disabled between program operations. Ordinarily, when a program or erase command is issued, an internal charge pump associated with the Flash memory must be enabled to supply high voltage to the array. Upon completion of the command, the charge pump is turned off. When a burst program command is issued, the charge pump is enabled and remains enabled after completion of the burst program operation if these two conditions are met:

- The next burst program command sequence has begun before the FCCF bit is set.
- The next sequential address selects a byte on the same burst block as the current byte being programmed. A burst block in this Flash memory consists of 32 bytes. A new burst block begins at each 32-byte address boundary.

The first byte of a series of sequential bytes being programmed in burst mode will take the same amount of time to program as a byte programmed in standard mode. Subsequent bytes will program in the burst

program time provided that the conditions above are met. If the next sequential address is the beginning of a new row, the program time for that byte will be the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump will be disabled and high voltage removed from the array.

A flowchart to execute the burst program operation is shown in Figure 4-3.

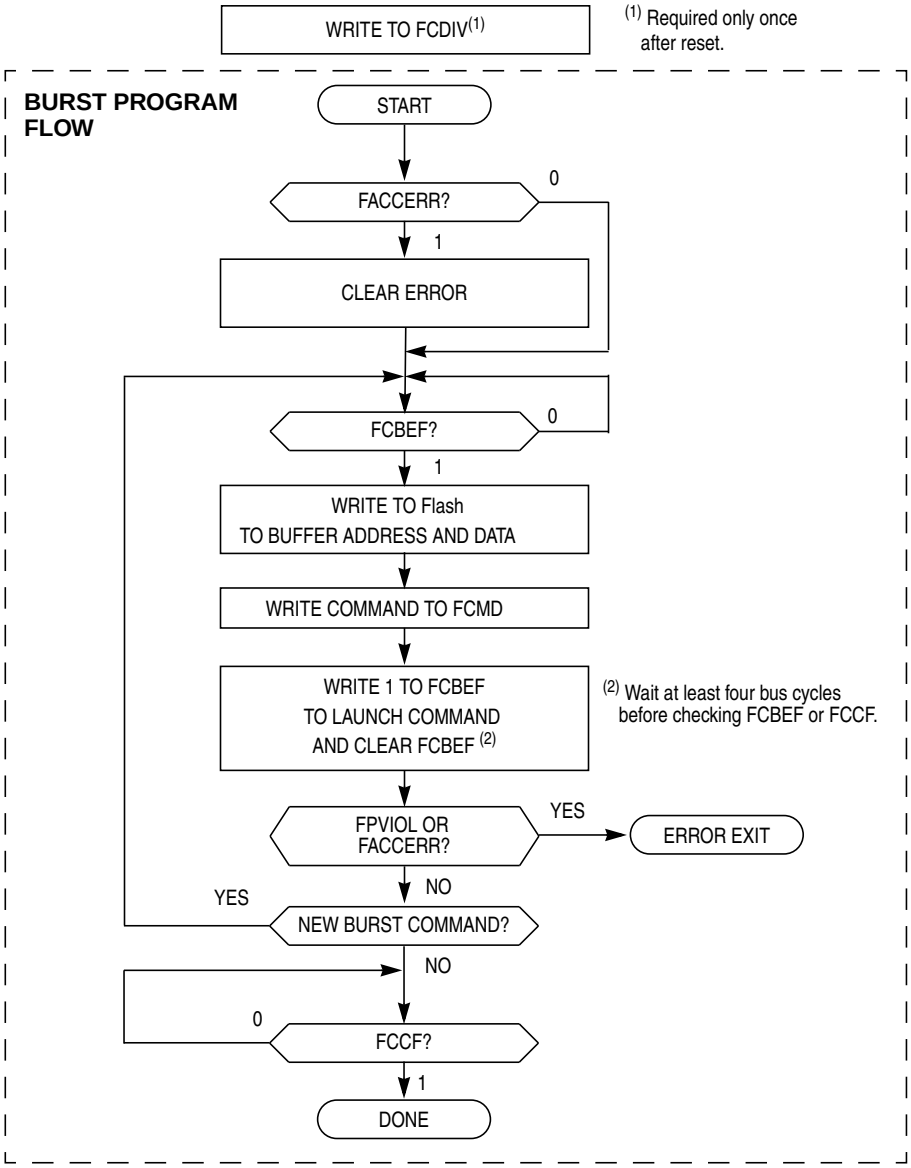



**Figure 4-3. Burst Program Flowchart**

### 4.5.5 Sector Erase Abort

The sector erase abort operation will terminate the active sector erase operation so that other sectors are available for read and program operations without waiting for the sector erase operation to complete.

The sector erase abort command write sequence is as follows:

1. Write to any Flash or EEPROM address to start the command write sequence for the sector erase abort command. The address and data written are ignored.
2. Write the sector erase abort command, 0x47, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the sector erase abort command.

If the sector erase abort command is launched resulting in the early termination of an active sector erase operation, the FACCERR flag will set once the operation completes as indicated by the FCCF flag being set. The FACCERR flag sets to inform the user that the Flash sector may not be fully erased and a new sector erase command must be launched before programming any location in that specific sector.

If the sector erase abort command is launched but the active sector erase operation completes normally, the FACCERR flag will not set upon completion of the operation as indicated by the FCCF flag being set. Therefore, if the FACCERR flag is not set after the sector erase abort command has completed, a sector being erased when the abort command was launched will be fully erased.

A flowchart to execute the sector erase abort operation is shown in Figure 4-4.

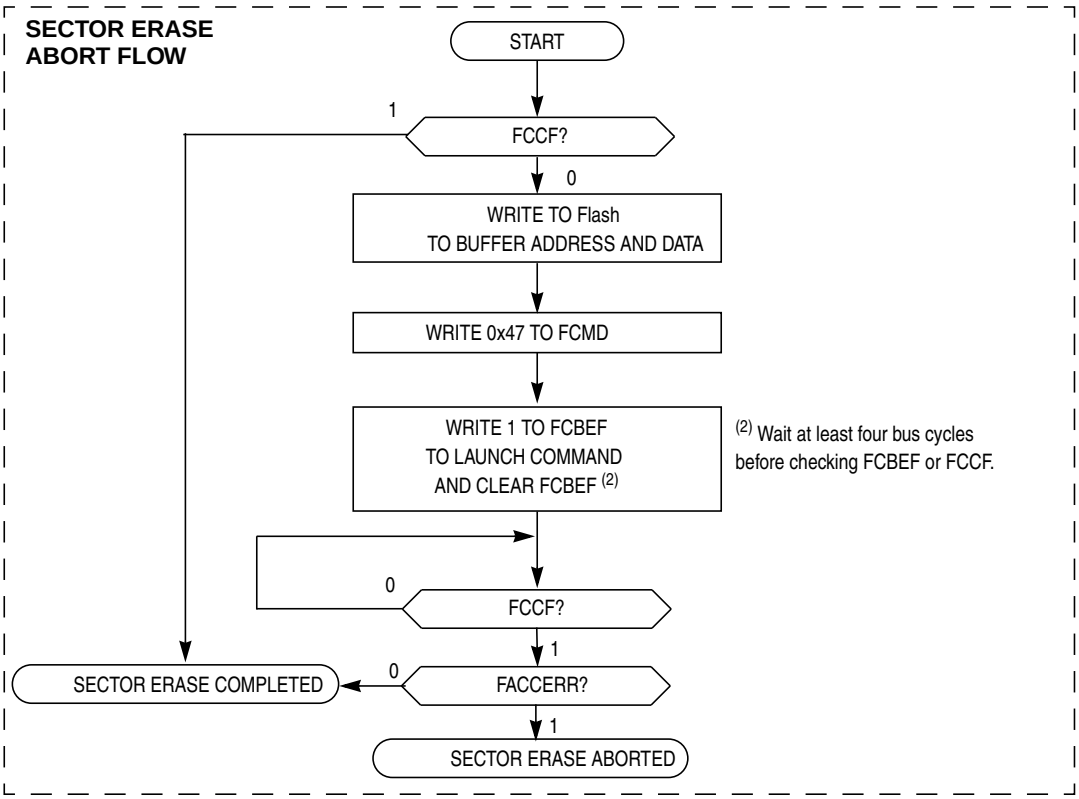



**Figure 4-4. Sector Erase Abort Flowchart**

**NOTE**

The FCBEF flag will not set after launching the sector erase abort command. If an attempt is made to start a new command write sequence with a sector erase abort operation active, the FACCERR flag in the FSTAT register will be set. A new command write sequence may be started after clearing the ACCERR flag, if set.

**NOTE**

The sector erase abort command should be used sparingly since a sector erase operation that is aborted counts as a complete program/erase cycle.

### 4.5.6 Access Errors

An access error occurs whenever the command execution protocol is violated.

Any of the following specific actions will cause the access error flag (FACCERR) in FSTAT to be set. FACCERR must be cleared by writing a 1 to FACCERR in FSTAT before any command can be processed.

- Writing to a Flash address before the internal Flash and EEPROM clock frequency has been set by writing to the FCDIV register.
- Writing to a Flash address while FCBEF is not set. (A new command cannot be started until the command buffer is empty.)
- Writing a second time to a Flash address before launching the previous command. (There is only one write to Flash for every command.)
- Writing a second time to FCMD before launching the previous command. (There is only one write to FCMD for every command.)
- Writing to any Flash control register other than FCMD after writing to a Flash address.
- Writing any command code other than the six allowed codes (0x05, 0x20, 0x25, 0x40, 0x41, or 0x47) to FCMD.
- Writing any Flash control register other than to write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD.
- The MCU enters stop mode while a program or erase command is in progress. (The command is aborted.)
- Writing the byte program, burst program, sector erase or sector erase abort command code (0x20, 0x25, 0x40, or 0x47) with a background debug command while the MCU is secured. (The background debug controller can do blank check and mass erase commands only when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command.

### 4.5.7 Block Protection

The block protection feature prevents the protected region of Flash or EEPROM from program or erase changes. Block protection is controlled through the Flash and EEPROM protection register (FPROT). The EPS bits determine the protected region of EEPROM and the FPS bits determine the protected region of Flash. See Section 4.5.11.4, “Flash and EEPROM Protection Register (FPROT and NVPROT)”.

After exit from reset, FPROT is loaded with the contents of the NVPROT location, which is in the nonvolatile register block of the Flash memory. Any FPROT write that attempts to decrease the size of the protected region will be ignored. Because NVPROT is within the last sector of Flash, if any amount of memory is protected, NVPROT is itself protected and cannot be unprotected (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands, which provides a way to erase and reprogram protected Flash memory.

One use for block protection is to block protect an area of Flash memory for a bootloader program. this bootloader program can call a routine outside of Flash that can be used to sector erase the rest of the Flash memory and reprogram it. The bootloader is protected even if MCU power is lost during an erase and reprogram operation.

### 4.5.8 Vector Redirection

While any Flash is block protected, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to 0. For redirection to occur, at least some portion of the Flash memory must be block protected by programming the NVPROT register located at address 0xFFBD. All interrupt vectors (memory locations 0xFFC0–0xFFFD) are redirected, though the reset vector (0xFFFE:0xFFFF) is not.

For example, if 1536 bytes of Flash are protected, the protected address region is from 0xFA00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFD) are redirected to the locations 0xF9C0–0xF9FD. If vector redirection is enabled and an interrupt occurs, the values in the locations 0xF9E0:0xF9E1 are used for the vector instead of the values in the locations 0xFFE0:0xFFE1. This allows the user to reprogram the unprotected portion of the Flash with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

### 4.5.9 Security

The MC9S08DZ60 Series includes circuitry to prevent unauthorized access to the contents of Flash, EEPROM, and RAM memory. When security is engaged, Flash, EEPROM, and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two register bits (SEC[1:0]) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from Flash into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location,

which can be performed at the same time the Flash memory is programmed. The 1:0 state disengages security; the other three combinations engage security. Notice the erased state (1:1) makes the MCU secure. During development, whenever the Flash is erased, it is good practice to immediately program the SEC0 bit to 0 in NVOPT so SEC = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

The on-chip debug module cannot be enabled while the MCU is secure. The separate background debug controller can be used for background memory access commands, but the MCU cannot enter active background mode except by holding BKGD low at the rising edge of reset.

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. If the nonvolatile KEYEN bit in NVOPT/FOPT is 0, the backdoor key is disabled and there is no way to disengage security without completely erasing all Flash locations. If KEYEN is 1, a secure user program can temporarily disengage security by:

1. Writing 1 to KEYACC in the FCNFG register. This makes the Flash module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a Flash program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be performed in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX must not be used for these writes because these writes cannot be performed on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was written matches the key stored in the Flash locations, SEC bits are automatically changed to 1:0 and security will be disengaged until the next reset.

The security key can be written only from secure memory (either RAM, EEPROM, or Flash), so it cannot be entered through background commands without the cooperation of a secure user program.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in Flash memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other Flash memory location. The nonvolatile registers are in the same 768-byte block of Flash as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

Security can always be disengaged through the background debug interface by taking these steps:

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase Flash if necessary.
3. Blank check Flash. Provided Flash is completely erased, security is disengaged until the next reset.

To avoid returning to secure mode after the next reset, program NVOPT so SEC = 1:0.

### 4.5.10 EEPROM Mapping

Only half of the EEPROM is in the memory map. The EPGSEL bit in FCNFG register selects which half of the array can be accessed in foreground while the other half can not be accessed in background. There are two mapping mode options that can be selected to configure the 8-byte EEPROM sectors: 4-byte mode and 8-byte mode. Each mode is selected by the EPGMOD bit in the FOPT register.

In 4-byte sector mode (EPGMOD = 0), each 8-byte sector splits four bytes on foreground and four bytes on background but on the same addresses. The EPGSEL bit selects which four bytes can be accessed. During a sector erase, the entire 8-byte sector (four bytes in foreground and four bytes in background) is erased.

In 8-byte sector mode (EPGMOD = 1), each entire 8-byte sector is in a single page. The EPGSEL bit selects which sectors are on background. During a sector erase, the entire 8-byte sector in foreground is erased.

### 4.5.11 Flash and EEPROM Registers and Control Bits

The Flash and EEPROM modules have seven 8-bit registers in the high-page register space and three locations in the nonvolatile register space in Flash memory. Two of those locations are copied into two corresponding high-page control registers at reset. There is also an 8-byte comparison key in Flash memory. Refer to Table 4-3 and Table 4-5 for the absolute address assignments for all Flash and EEPROM registers. This section refers to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

#### 4.5.11.1 Flash and EEPROM Clock Divider Register (FCDIV)

Bit 7 of this register is a read-only flag. Bits 6:0 may be read at any time but can be written only one time. Before any erase or programming operations are possible, write to this register to set the frequency of the clock for the nonvolatile memory system within acceptable limits.

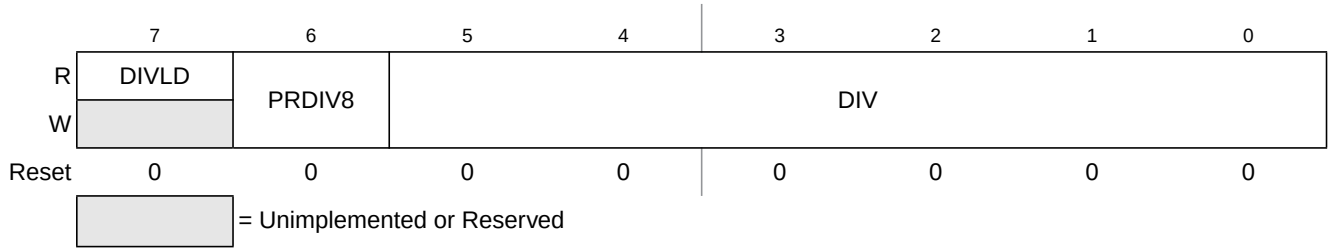



**Figure 4-5. Flash and EEPROM Clock Divider Register (FCDIV)**

**Table 4-7. FCDIV Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DIVLD | **Divisor Loaded Status Flag** — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit and the first write to this register causes this bit to become set regardless of the data written.<br>0 FCDIV has not been written since reset; erase and program operations disabled for Flash and EEPROM.<br>1 FCDIV has been written since reset; erase and program operations enabled for Flash and EEPROM. |
| 6<br>PRDIV8 | **Prescale (Divide) Flash and EEPROM Clock by 8** (This bit is write once.)<br>0 Clock input to the Flash and EEPROM clock divider is the bus rate clock.<br>1 Clock input to the Flash and EEPROM clock divider is the bus rate clock divided by 8. |
| 5:0<br>DIV | **Divisor for Flash and EEPROM Clock Divider** — These bits are write once. The Flash and EEPROM clock divider divides the bus rate clock (or the bus rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV field plus one. The resulting frequency of the internal Flash and EEPROM clock must fall within the range of 200 kHz to 150 kHz for proper Flash and EEPROM operations. Program/Erase timing pulses are one cycle of this internal Flash and EEPROM clock which corresponds to a range of 5 µs to 6.7 µs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See Equation 4-1 and Equation 4-2. |

$$\text{if PRDIV8 = 0 — } f_{FCLK} = f_{Bus} \div (DIV + 1) \qquad \textit{Eqn. 4-1}$$

$$\text{if PRDIV8 = 1 — } f_{FCLK} = f_{Bus} \div (8 \times (DIV + 1)) \qquad \textit{Eqn. 4-2}$$

Table 4-8 shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

**Table 4-8. Flash and EEPROM Clock Divider Settings**

| $f_{Bus}$ | PRDIV8 (Binary) | DIV (Decimal) | $f_{FCLK}$ | Program/Erase Timing Pulse (5 µs Min, 6.7 µs Max) |
|---|---|---|---|---|
| 20 MHz | 1 | 12 | 192.3 kHz | 5.2 µs |
| 10 MHz | 0 | 49 | 200 kHz | 5 µs |
| 8 MHz | 0 | 39 | 200 kHz | 5 µs |
| 4 MHz | 0 | 19 | 200 kHz | 5 µs |
| 2 MHz | 0 | 9 | 200 kHz | 5 µs |
| 1 MHz | 0 | 4 | 200 kHz | 5 µs |
| 200 kHz | 0 | 0 | 200 kHz | 5 µs |
| 150 kHz | 0 | 0 | 150 kHz | 6.7 µs |

### 4.5.11.2 Flash and EEPROM Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from Flash into FOPT. To change the value in this register, erase and reprogram the NVOPT location in Flash memory as usual and then issue a new MCU reset.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KEYEN | FNORED | EPGMOD | 0 | 0 | 0 | SEC | |
| W | | | | | | | | |
| Reset | F | F | F | 0 | 0 | 0 | F | F |

= Unimplemented or Reserved

F = loaded from nonvolatile location NVOPT during reset

**Figure 4-6. Flash and EEPROM Options Register (FOPT)**

**Table 4-9. FOPT Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>KEYEN | **Backdoor Key Mechanism Enable** — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.5.9, “Security.”<br>0 No backdoor key access allowed.<br>1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset. |
| 6<br>FNORED | **Vector Redirection Disable** — When this bit is 1, then vector redirection is disabled.<br>0 Vector redirection enabled.<br>1 Vector redirection disabled. |
| 5<br>EPGMOD | **EEPROM Sector Mode** — When this bit is 0, each sector is split into two pages (4-byte mode). When this bit is 1, each sector is in a single page (8-byte mode).<br>0 Half of each EEPROM sector is in Page 0 and the other half is in Page 1.<br>1 Each sector is in a single page. |
| 1:0<br>SEC | **Security State Code** — This 2-bit field determines the security state of the MCU as shown in Table 4-10. When the MCU is secure, the contents of RAM, EEPROM and Flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. SEC changes to 1:0 after successful backdoor key entry or a successful blank check of Flash. For more detailed information about security, refer to Section 4.5.9, “Security.” |

**Table 4-10. Security States[1]**

| SEC[1:0] | Description |
|---|---|
| 0:0 | secure |
| 0:1 | secure |
| 1:0 | unsecured |
| 1:1 | secure |

1 SEC changes to 1:0 after successful backdoor key entry or a successful blank check of Flash.

### 4.5.11.3 Flash and EEPROM Configuration Register (FCNFG)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | EPGSEL | KEYACC | Reserved[1] | 0 | 0 | 0 | 1 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

= Unimplemented or Reserved

**Figure 4-7. Flash Configuration Register (FCNFG)**

1 User must write a 1 to this bit. Failing to do so may result in unexpected behavior.

**Table 4-11. FCNFG Register Field Descriptions**

| Field | Description |
|---|---|
| 6<br>EPGSEL | **EEPROM Page Select** — This bit selects which EEPROM page is accessed in the memory map.<br>0 Page 0 is in foreground of memory map. Page 1 is in background and can not be accessed.<br>1 Page 1 is in foreground of memory map. Page 0 is in background and can not be accessed. |
| 5<br>KEYACC | **Enable Writing of Access Key** — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to Section 4.5.9, “Security.”<br>0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a Flash programming or erase command.<br>1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes. |

### 4.5.11.4 Flash and EEPROM Protection Register (FPROT and NVPROT)

The FPROT register defines which Flash and EEPROM sectors are protected against program and erase operations.

During the reset sequence, the FPROT register is loaded from the nonvolatile location NVPROT. To change the protection that will be loaded during the reset sequence, the sector containing NVPROT must be unprotected and erased, then NVPROT can be reprogrammed.

FPROT bits are readable at any time and writable as long as the size of the protected region is being increased. Any write to FPROT that attempts to decrease the size of the protected memory will be ignored.

Trying to alter data in any protected area will result in a protection violation error and the FPVIOL flag will be set in the FSTAT register. Mass erase is not possible if any one of the sectors is protected.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EPS[1] | | | | FPS[1] | | | |
| W | | | | | | | | |
| Reset | This register is loaded from nonvolatile location NVPROT during reset. | | | | | | | |

1 Background commands can be used to change the contents of these bits in FPROT.

**Figure 4-8. Flash and EEPROM Protection Register (FPROT)**

**Table 4-12. FPROT Register Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>EPS | **EEPROM Protect Select Bits** — This 2-bit field determines the protected EEPROM locations that cannot be erased or programmed. See Table 4-13. |
| 5:0<br>FPS | **Flash Protect Select Bits** — This 6-bit field determines the protected Flash locations that cannot be erased or programmed. SeeTable 4-14. |

**Table 4-13. EEPROM Block Protection**

| EPS | Address Area Protected | Memory Size Protected (bytes) | Number of Sectors Protected |
|---|---|---|---|
| 0x3 | N/A | 0 | 0 |
| 0x2 | 0x17F0 - 0x17FF | 32 | 4 |
| 0x1 | 0x17E0 - 0x17FF | 64 | 8 |
| 0x0 | 0x17C0–0x17FF | 128 | 16 |

**Table 4-14. Flash Block Protection**

| FPS | Address Area Protected | Memory Size Protected (bytes) | Number of Sectors Protected |
|---|---|---|---|
| 0x3F | N/A | 0 | 0 |
| 0x3E | 0xFA00–0xFFFF | 1.5K | 2 |
| 0x3D | 0xF400–0xFFFF | 3K | 4 |
| 0x3C | 0xEE00–0xFFFF | 4.5K | 6 |
| 0x3B | 0xE800–0xFFFF | 6K | 8 |
| ... | ... | ... | ... |
| 0x37 | 0xD000–0xFFFF | 12K | 16 |
| 0x36 | 0xCA00–0xFFFF | 13.5K | 18 |
| 0x35 | 0xC400–0xFFFF | 15K | 20 |
| 0x34 | 0xBE00–0xFFFF | 16.5K | 22 |
| ... | ... | ... | ... |
| 0x2C | 0x8E00–0xFFFF | 28.5K | 38 |
| 0x2B | 0x8800–0xFFFF | 30K | 40 |
| 0x2A | 0x8200–0xFFFF | 31.5K | 42 |
| 0x29 | 0x7C00–0xFFFF | 33K | 44 |
| ... | ... | ... | ... |
| 0x22 | 0x5200–0xFFFF | 43.5K | 58 |
| 0x21 | 0x4C00–0xFFFF | 45K | 60 |
| 0x20 | 0x4600–0xFFFF | 46.5K | 62 |
| 0x1F | 0x4000–0xFFFF | 48K | 64 |
| ... | ... | ... | ... |

**Table 4-14. Flash Block Protection (continued)**

| FPS | Address Area Protected | Memory Size Protected (bytes) | Number of Sectors Protected |
|---|---|---|---|
| 0x1B | 0x2800–0xFFFF | 54K | 72 |
| 0x1A | 0x2200–0xFFFF | 55.5K | 74 |
| 0x19 | 0x1C00–0xFFFF | 57K | 76 |
| 0x18–0x00 | 0x0000–0xFFFF | 64K | 86 |

## 4.5.11.5 Flash and EEPROM Status Register (FSTAT)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 4-9. Flash and EEPROM Status Register (FSTAT)**

**Table 4-15. FSTAT Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FCBEF | **Command Buffer Empty Flag** — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a 1 to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered.<br>0 Command buffer is full (not ready for additional commands).<br>1 A new burst program command can be written to the command buffer. |
| 6<br>FCCF | **Command Complete Flag** — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect.<br>0 Command in progress<br>1 All commands complete |
| 5<br>FPVIOL | **Protection Violation Flag** — FPVIOL is set automatically when a command that attempts to erase or program a location in a protected block is launched (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL.<br>0 No protection violation.<br>1 An attempt was made to erase or program a protected location. |

**Table 4-15. FSTAT Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>FACCERR | **Access Error Flag** — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.5.6, "Access Errors." FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect.<br>0 No access error.<br>1 An access error has occurred. |
| 2<br>FBLANK | **Verified as All Blank (erased) Flag** — FBLANK is set automatically at the conclusion of a blank check command if the entire Flash or EEPROM array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect.<br>0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the Flash or EEPROM array is not completely erased.<br>1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the Flash or EEPROM array is completely erased (all 0xFFFF). |

### 4.5.11.6 Flash and EEPROM Command Register (FCMD)

Only six command codes are recognized in normal user modes, as shown in Table 4-16. All other command codes are illegal and generate an access error. Refer to Section 4.5.3, "Program and Erase Command Execution," for a detailed discussion of Flash and EEPROM programming and erase operations.

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

**Figure 4-10. Flash and EEPROM Command Register (FCMD)**

**Table 4-16. Flash and EEPROM Commands**

| Command | FCMD | Equate File Label |
|---|---|---|
| Blank check | 0x05 | mBlank |
| Byte program | 0x20 | mByteProg |
| Burst program | 0x25 | mBurstProg |
| Sector erase | 0x40 | mSectorErase |
| Mass erase | 0x41 | mMassErase |
| Sector erase abort | 0x47 | mEraseAbort |

It is not necessary to perform a blank check command after a mass erase operation. Only blank check is required as part of the security unlocking mechanism.

# Chapter 5
# Resets, Interrupts, and General System Control

## 5.1 Introduction

This section discusses basic reset and interrupt mechanisms and their various sources in the MC9S08DZ60 Series. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this data sheet. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog, are not part of on-chip peripheral systems with their own chapters.

## 5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vector for each module (reduces polling overhead); see Table 5-1

## 5.3 MCU Reset

Resetting the MCU provides a way to start processing from a known set of initial conditions. During reset, most control and status registers are forced to initial values and the program counter is loaded from the reset vector (0xFFFE:0xFFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pull-up devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. (See the CPU chapter for information on the Interrupt (I) bit.) SP is forced to 0x00FF at reset.

The MC9S08DZ60 Series has eight sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Low-voltage detect (LVD)
- Loss of clock (LOC)
- Background debug forced reset (BDFR)

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register (SRS).

## 5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the COP watchdog is enabled (see Section 5.8.4, “System Options Register 1 (SOPT1),” for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPT bits in SOPT1.

The COP counter is reset by writing 0x55 and 0xAA (in this order) to the address of SRS during the selected timeout period. Writes do not affect the data in the read-only SRS. As soon as the write sequence is done, the COP timeout period is restarted. If the program fails to do this during the time-out period, the MCU will reset. Also, if any value other than 0x55 or 0xAA is written to SRS, the MCU is immediately reset.

The COPCLKS bit in SOPT2 (see Section 5.8.5, “System Options Register 2 (SOPT2),” for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1-kHz clock source. With each clock source, there are three associated time-outs controlled by the COPT bits in SOPT1. Table 5-6 summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1-kHz clock source and the longest time-out ($2^{10}$ cycles).

When the bus clock source is selected, windowed COP operation is available by setting COPW in the SOPT2 register. In this mode, writes to the SRS register to clear the COP timer must occur in the last 25% of the selected timeout period. A premature write immediately resets the MCU. When the 1-kHz clock source is selected, windowed COP operation is not available.

The COP counter is initialized by the first writes to the SOPT1 and SOPT2 registers and after any system reset. Subsequent writes to SOPT1 and SOPT2 have no effect on COP operation. Even if the application will use the reset default settings of COPT, COPCLKS, and COPW bits, the user should write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. This will prevent accidental changes if the application program gets lost.

The write to SRS that services (clears) the COP counter should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

If the bus clock source is selected, the COP counter does not increment while the MCU is in background debug mode or while the system is in stop mode. The COP counter resumes when the MCU exits background debug mode or stop mode.

If the 1-kHz clock source is selected, the COP counter is re-initialized to zero upon entry to either background debug mode or stop mode and begins from zero upon exit from background debug mode or stop mode.

## 5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing resumes where it left off before the interrupt. Other than the software interrupt (SWI), which is a program instruction, interrupts are caused by hardware events such as an edge on the IRQ pin or a timer-overflow event. The debug module can also generate an SWI under certain circumstances.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond unless the local interrupt enable is a 1 to enable the interrupt and the I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which prevents all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence obeys the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit can be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction which restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information from the stack.

**NOTE**

For compatibility with M68HC08 devices, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it immediately before the RTI that is used to return from the ISR.

If more than one interrupt is pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-1).

### 5.5.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

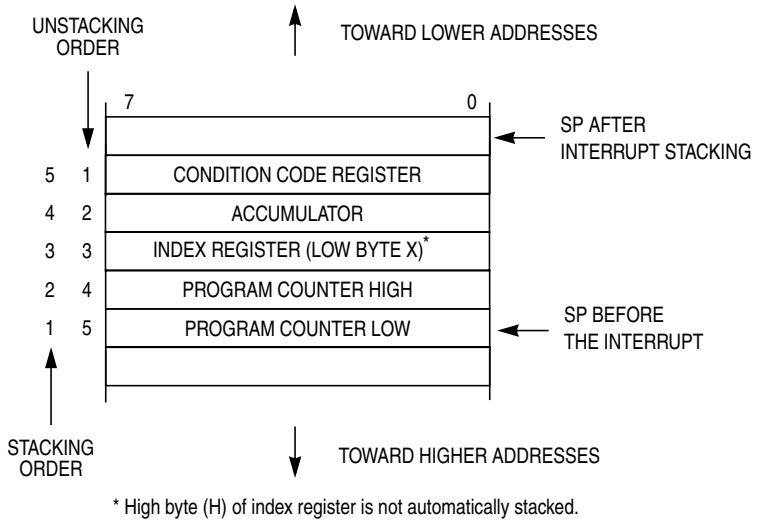



**Figure 5-1. Interrupt Stack Frame**

When an RTI instruction is executed, these values are recovered from the stack in reverse order. As part of the RTI sequence, the CPU fills the instruction pipeline by reading three bytes of program information, starting from the PC address recovered from the stack.

The status flag corresponding to the interrupt source must be acknowledged (cleared) before returning from the ISR. Typically, the flag is cleared at the beginning of the ISR so that if another interrupt is generated by this same source, it will be registered so it can be serviced after completion of the current ISR.

### 5.5.2 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQ status and control register, IRQSC. When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events. When the MCU is in stop mode and system clocks are shut down, a separate asynchronous path is used so the IRQ (if enabled) can wake the MCU.

#### 5.5.2.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in IRQSC must be 1 in order for the IRQ pin to act as the interrupt request (IRQ) input. As an IRQ input, the user can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD), and whether an event causes an interrupt or only sets the IRQF flag which can be polled by software.

The IRQ pin, when enabled, defaults to use an internal pull device (IRQPDD = 0), the device is a pull-up or pull-down depending on the polarity chosen. If the user desires to use an external pull-up or pull-down, the IRQPDD can be written to a 1 to turn off the internal device.

BIH and BIL instructions may be used to detect the level on the IRQ pin when the pin is configured to act as the IRQ input.

#### 5.5.2.2 Edge and Level Sensitivity

The IRQMOD control bit reconfigures the detection logic so it detects edge events and pin levels. In the edge and level detection mode, the IRQF status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

### 5.5.3 Interrupt Vectors, Sources, and Local Masks

Table 5-1 provides a summary of all interrupt sources. Higher-priority sources are located toward the bottom of the table. The high-order byte of the address for the interrupt service routine is located at the first address in the vector address column, and the low-order byte of the address for the interrupt service routine is located at the next higher address.

When an interrupt condition occurs, an associated flag bit becomes set. If the associated local interrupt enable is 1, an interrupt request is sent to the CPU. Within the CPU, if the global interrupt mask (I bit in the CCR) is 0, the CPU will finish the current instruction; stack the PCL, PCH, X, A, and CCR CPU registers; set the I bit; and then fetch the interrupt vector for the highest priority pending interrupt. Processing then continues in the interrupt service routine.

**Table 5-1. Vector Summary[1]**

| Vector No. | Address (High/Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|
| 31 | 0xFFC0/0xFFC1 | Vacmp2 | ACMP2 | ACF | ACIE | Analog comparator 2 |
| 30 | 0xFFC2/0xFFC3 | Vacmp1 | ACMP1 | ACF | ACIE | Analog comparator 1 |
| 29 | 0xFFC4/0xFFC5 | Vcantx | MSCAN | TXE[2:0] | TXEIE[2:0] | CAN transmit |
| 28 | 0xFFC6/0xFFC7 | Vcanrx | MSCAN | RXF | RXFIE | CAN receive |
| 27 | 0xFFC8/0xFFC9 | Vcanerr | MSCAN | CSCIF, OVRIF | CSCIE, OVRIE | CAN errors |
| 26 | 0xFFCA/0xFFCB | Vcanwu | MSCAN | WUPIF | WUPIE | CAN wake-up |
| 25 | 0xFFCC/0xFFCD | Vrtc | RTC | RTIF | RTIE | Real-time interrupt |
| 24 | 0xFFCE/0xFFCF | Viic | IIC | IICIS | IICIE | IIC control |
| 23 | 0xFFD0/0xFFD1 | Vadc | ADC | COCO | AIEN | ADC |
| 22 | 0xFFD2/0xFFD3 | Vport | Port A,B,D | PTAIF, PTBIF, PTDIF | PTAIE, PTBIE, PTDIE | Port Pins |
| 21 | 0xFFD4/0xFFD5 | Vsci2tx | SCI2 | TDRE, TC | TIE, TCIE | SCI2 transmit |
| 20 | 0xFFD6/0xFFD7 | Vsci2rx | SCI2 | IDLE, LBKDIF, RDRF, RXEDGIF | ILIE, LBKDIE, RIE, RXEDGIE | SCI2 receive |
| 19 | 0xFFD8/0xFFD9 | Vsci2err | SCI2 | OR, NF FE, PF | ORIE, NFIE, FEIE, PFIE | SCI2 error |
| 18 | 0xFFDA/0xFFDB | Vsci1tx | SCI1 | TDRE, TC | TIE, TCIE | SCI1 transmit |
| 17 | 0xFFDC/0xFFDD | Vsci1rx | SCI1 | IDLE, LBKDIF, RDRF, RXEDGIF | ILIE, LBKDIE, RIE, RXEDGIE | SCI1 receive |
| 16 | 0xFFDE/0xFFDF | Vsci1err | SCI1 | OR, NF, FE, PF | ORIE, NFIE, FEIE, PFIE | SCI1 error |
| 15 | 0xFFE0/0xFFE1 | Vspi | SPI | SPIF, MODF, SPTEF | SPIE, SPIE, SPTIE | SPI |
| 14 | 0xFFE2/0xFFE3 | Vtpm2ovf | TPM2 | TOF | TOIE | TPM2 overflow |
| 13 | 0xFFE4/0xFFE5 | Vtpm2ch1 | TPM2 | CH1F | CH1IE | TPM2 channel 1 |
| 12 | 0xFFE6/0xFFE7 | Vtpm2ch0 | TPM2 | CH0F | CH0IE | TPM2 channel 0 |
| 11 | 0xFFE8/0xFFE9 | Vtpm1ovf | TPM1 | TOF | TOIE | TPM1 overflow |
| 10 | 0xFFEA/0xFFEB | Vtpm1ch5 | TPM1 | CH5F | CH5IE | TPM1 channel 5 |
| 9 | 0xFFEC/0xFFED | Vtpm1ch4 | TPM1 | CH4F | CH4IE | TPM1 channel 4 |
| 8 | 0xFFEE/0xFFEF | Vtpm1ch3 | TPM1 | CH3F | CH3IE | TPM1 channel 3 |
| 7 | 0xFFF0/0xFFF1 | Vtpm1ch2 | TPM1 | CH2F | CH2IE | TPM1 channel 2 |
| 6 | 0xFFF2/0xFFF3 | Vtpm1ch1 | TPM1 | CH1F | CH1IE | TPM1 channel 1 |
| 5 | 0xFFF4/0xFFF5 | Vtpm1ch0 | TPM1 | CH0F | CH0IE | TPM1 channel 0 |
| 4 | 0xFFF6/0xFFF7 | Vlol | MCG | LOLS | LOLIE | MCG loss of lock |
| 3 | 0xFFF8/0xFFF9 | Vlvd | System control | LVWF | LVWIE | Low-voltage warning |
| 2 | 0xFFFA/0xFFFB | Virq | IRQ | IRQF | IRQIE | IRQ pin |
| 1 | 0xFFFC/0xFFFD | Vswi | Core | SWI Instruction | — | Software interrupt |
| 0 | 0xFFFE/0xFFFF | Vreset | System control | COP,<br>LOC,<br>LVD,<br>$\overline{\text{RESET}}$,<br>ILOP,<br>ILAD,<br>POR,<br>BDFR | COPE<br>CME<br>LVDRE<br>—<br>—<br>—<br>—<br>— | Watchdog timer<br>Loss-of-clock<br>Low-voltage detect<br>External pin<br>Illegal opcode<br>Illegal address<br>Power-on-reset<br>BDM-forced reset |

1 Vector priority is shown from lowest (first row) to highest (last row). For example, Vreset is the highest priority vector.

## 5.6 Low-Voltage Detect (LVD) System

The MC9S08DZ60 Series includes a system to protect against low-voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with trip voltages for warning and detection. The LVD circuit is enabled when LVDE in SPMSC1 is set to 1. The LVD is disabled upon entering any of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU cannot enter stop2 (it will enter stop3 instead), and the current consumption in stop3 with the LVD enabled will be higher.

### 5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the power-on reset rearm voltage level, $V_{POR}$, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above the low-voltage detection low threshold, $V_{LVDL}$. Both the POR bit and the LVD bit in SRS are set following a POR.

### 5.6.2 Low-Voltage Detection (LVD) Reset Operation

The LVD can be configured to generate a reset upon detection of a low-voltage condition by setting LVDRE to 1. The low-voltage detection threshold is determined by the LVDV bit. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage has risen above the low-voltage detection threshold. The LVD bit in the SRS register is set following either an LVD reset or POR.

### 5.6.3 Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low-voltage warning flag to indicate to the user that the supply voltage is approaching the low-voltage condition. When a low-voltage warning condition is detected and is configured for interrupt operation (LVWIE set to 1), LVWF in SPMSC1 will be set and an LVW interrupt request will occur.

## 5.7 MCLK Output

The PTA0 pin is shared with the MCLK clock output. If the MCSEL bits are all zeroes, the MCLK clock is disabled. Setting any of the MCSEL bits causes the PTA0 pin to output a divided version of the internal MCU bus clock regardless of the state of the port data direction control bit for the pin. The divide ratio is determined by the MCSEL bits. The slew rate and drive strength for the pin are controlled by PTASE0 and PTADS0, respectively. The maximum clock output frequency is limited if slew rate control is enabled, see the electrical specifications for the maximum frequency under different conditions.

## 5.8 Reset, Interrupt, and System Control Registers and Control Bits

One 8-bit register in the direct page register space and eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

Refer to Table 4-2 and Table 4-3 in Chapter 4, “Memory,” of this data sheet for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in Chapter 3, “Modes of Operation.”

## 5.8.1 Interrupt Pin Request Status and Control Register (IRQSC)

This direct page register includes status and control bits which are used to configure the IRQ function, report status, and acknowledge IRQ events.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | IRQPDD | IRQEDG | IRQPE | IRQF | 0 | IRQIE | IRQMOD |
| W | | | | | | IRQACK | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 5-2. Interrupt Request Status and Control Register (IRQSC)**

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

| Field | Description |
|---|---|
| 6<br>IRQPDD | **Interrupt Request (IRQ) Pull Device Disable**— This read/write control bit is used to disable the internal pull-up/pull-down device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used.<br>0 IRQ pull device enabled if IRQPE = 1.<br>1 IRQ pull device disabled if IRQPE = 1. |
| 5<br>IRQEDG | **Interrupt Request (IRQ) Edge Select** — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When the IRQ pin is enabled as the IRQ input and is configured to detect rising edges, it has a pull-down. When the IRQ pin is enabled as the IRQ input and is configured to detect falling edges, it has a pull-up.<br>0 IRQ is falling edge or falling edge/low-level sensitive.<br>1 IRQ is rising edge or rising edge/high-level sensitive. |
| 4<br>IRQPE | **IRQ Pin Enable** — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request.<br>0 IRQ pin function is disabled.<br>1 IRQ pin function is enabled. |
| 3<br>IRQF | **IRQ Flag** — This read-only status bit indicates when an interrupt request event has occurred.<br>0 No IRQ request.<br>1 IRQ event detected. |
| 2<br>IRQACK | **IRQ Acknowledge** — This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level. |
| 1<br>IRQIE | **IRQ Interrupt Enable** — This read/write control bit determines whether IRQ events generate an interrupt request.<br>0 Interrupt request when IRQF set is disabled (use polling).<br>1 Interrupt requested whenever IRQF = 1. |
| 0<br>IRQMOD | **IRQ Detection Mode** — This read/write control bit selects either edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels that are detected as interrupt request events. See Section 5.5.2.2, “Edge and Level Sensitivity” for more details.<br>0 IRQ event on falling edges or rising edges only.<br>1 IRQ event on falling edges and low levels or on rising edges and high levels. |

## 5.8.2 System Reset Status Register (SRS)

This high page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDFR register, none of the status bits in SRS will be set. Writing any value to this register address causes a COP reset when the COP is enabled except the values 0x55 and 0xAA. Writing a 0x55-0xAA sequence to this address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | POR | PIN | COP | ILOP | ILAD | LOC | LVD | 0 |
| W | Writing 0x55, 0xAA to SRS address clears COP watchdog timer. | | | | | | | |
| POR: | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| LVD: | u | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Any other reset: | 0 | Note(1) | Note(1) | Note(1) | Note(1) | 0 | 0 | 0 |

1 Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

**Figure 5-3. System Reset Status (SRS)**

**Table 5-3. SRS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>POR | **Power-On Reset** — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold.<br>0 Reset not caused by POR.<br>1 POR caused reset. |
| 6<br>PIN | **External Reset Pin** — Reset was caused by an active-low level on the external reset pin.<br>0 Reset not caused by external reset pin.<br>1 Reset came from external reset pin. |
| 5<br>COP | **Computer Operating Properly (COP) Watchdog** — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by COPE = 0.<br>0 Reset not caused by COP timeout.<br>1 Reset caused by COP timeout. |
| 4<br>ILOP | **Illegal Opcode** — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register.<br>0 Reset not caused by an illegal opcode.<br>1 Reset caused by an illegal opcode. |
| 3<br>ILAD | **Illegal Address** — Reset was caused by an attempt to access either data or an instruction at an unimplemented memory address.<br>0 Reset not caused by an illegal address.<br>1 Reset caused by an illegal address. |

**Table 5-3. SRS Register Field Descriptions**

| Field | Description |
|---|---|
| 2<br>LOC | **Loss of Clock** — Reset was caused by a loss of external clock.<br>0 Reset not caused by loss of external clock<br>1 Reset caused by loss of external clock |
| 1<br>LVD | **Low-Voltage Detect** — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR.<br>0 Reset not caused by LVD trip or POR.<br>1 Reset caused by LVD trip or POR. |

## 5.8.3 System Background Debug Force Reset Register (SBDFR)

This high page register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

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

= Unimplemented or Reserved

1 BDFR is writable only through serial background debug commands, not from user programs.

**Figure 5-4. System Background Debug Force Reset Register (SBDFR)**

**Table 5-4. SBDFR Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>BDFR | **Background Debug Force Reset** — A serial background command such as WRITE_BYTE can be used to allow an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program. |

## 5.8.4 System Options Register 1 (SOPT1)

This high page register is a write-once register so only the first write after reset is honored. It can be read at any time. Any subsequent attempt to write to SOPT1 (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. This register should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | COPT | | STOPE | SCI2PS | IICPS | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 5-5. System Options Register 1 (SOPT1)**

**Table 5-5. SOPT1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>COPT[1:0] | **COP Watchdog Timeout** — These write-once bits select the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. See Table 5-6. |
| 5<br>STOPE | **Stop Mode Enable** — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced.<br>0 Stop mode disabled.<br>1 Stop mode enabled. |
| 4<br>SCI2PS | **SCI2 Pin Select**— This write-once bit selects the location of the RxD2 and TxD2 pins of the SCI2 module.<br>0 TxD2 on PTF0, RxD2 on PTF1.<br>1 TxD2 on PTE6, RxD2 on PTE7. |
| 3<br>IICPS | **IIC Pin Select**— This write-once bit selects the location of the SCL and SDA pins of the IIC module.<br>0 SCL on PTF2, SDA on PTF3.<br>1 SCL on PTE4, SDA on PTE5. |

**Table 5-6. COP Configuration Options**

| Control Bits | | Clock Source | COP Window[1] Opens (COPW = 1) | COP Overflow Count |
|---|---|---|---|---|
| COPCLKS | COPT[1:0] | | | |
| N/A | 0:0 | N/A | N/A | COP is disabled |
| 0 | 0:1 | 1 kHz | N/A | $2^5$ cycles (32 ms[2]) |
| 0 | 1:0 | 1 kHz | N/A | $2^8$ cycles (256 ms[1]) |
| 0 | 1:1 | 1 kHz | N/A | $2^{10}$ cycles (1.024 s[1]) |
| 1 | 0:1 | Bus | 6144 cycles | $2^{13}$ cycles |
| 1 | 1:0 | Bus | 49,152 cycles | $2^{16}$ cycles |
| 1 | 1:1 | Bus | 196,608 cycles | $2^{18}$ cycles |

1 Windowed COP operation requires the user to clear the COP timer in the last 25% of the selected timeout period. This column displays the minimum number of clock counts required before the COP timer can be reset when in windowed COP mode (COPW = 1).

2 Values shown in milliseconds based on $t_{LPO}$ = 1 ms. See $t_{LPO}$ in the appendix Section A.12.1, "Control Timing," for the tolerance of this value.

## 5.8.5 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the MC9S08DZ60 Series devices.

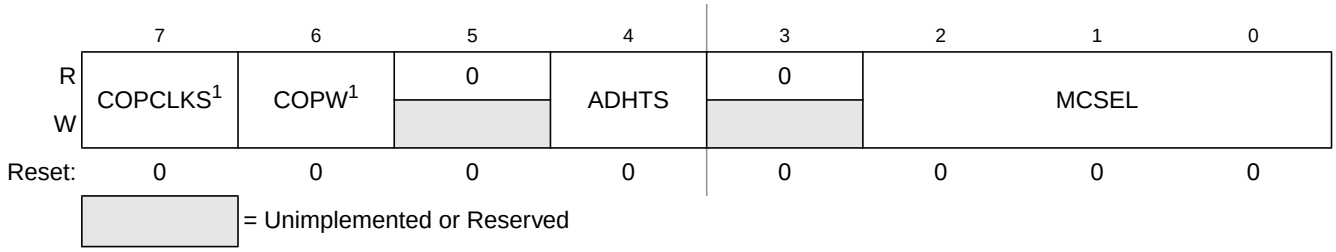



**Figure 5-6. System Options Register 2 (SOPT2)**

1 This bit can be written only one time after reset. Additional writes are ignored.

**Table 5-7. SOPT2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COPCLKS | **COP Watchdog Clock Select** — This write-once bit selects the clock source of the COP watchdog. See Table 5-6 for details.<br>0 Internal 1-kHz clock is source to COP.<br>1 Bus clock is source to COP. |
| 6<br>COPW | **COP Window** — This write-once bit selects the COP operation mode. When set, the 0x55-0xAA write sequence to the SRS register must occur in the last 25% of the selected period. Any write to the SRS register during the first 75% of the selected period will reset the MCU.<br>0 Normal COP operation.<br>1 Window COP operation. |
| 4<br>ADHTS | **ADC Hardware Trigger Select** — This bit selects which hardware trigger initiates conversion for the analog to digital converter when the ADC hardware trigger is enabled (ADCTRG is set in ADCSC2 register).<br>0 Real Time Counter (RTC) overflow.<br>1 External Interrupt Request (IRQ) pin. |
| 2:0<br>MCSEL | **MCLK Divide Select**— These bits enable the MCLK output on PTA0 pin and select the divide ratio for the MCLK output according to the formula below when the MCSEL bits are not equal to all zeroes. In case that the MCSEL bits are all zeroes, the MCLK output is disabled.<br>MCLK frequency = Bus Clock frequency ÷ (2 * MCSEL) |

## 5.8.6 System Device Identification Register (SDIDH, SDIDL)

These high page read-only registers are included so host development systems can identify the HCS08 derivative and revision number. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | ID11 | ID10 | ID9 | ID8 |
| W | | | | | | | | |
| Reset: | 0[1] | 0[1] | 0[1] | 0[1] | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

1 The revision number that is hard coded into these bits reflects the current silicon revision level.

**Figure 5-7. System Device Identification Register — High (SDIDH)**

**Table 5-8. SDIDH Register Field Descriptions**

| Field | Description |
|---|---|
| 3:0<br>ID[11:8] | **Part Identification Number** — MC9S08DZ60 Series MCUs are hard-coded to the value 0x00E. See also ID bits in Table 5-9. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

= Unimplemented or Reserved

**Figure 5-8. System Device Identification Register — Low (SDIDL)**

**Table 5-9. SDIDL Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>ID[7:0] | **Part Identification Number** — MC9S08DZ60 Series MCUs are hard-coded to the value 0x00E. See also ID bits in Table 5-8. |

## 5.8.7 System Power Management Status and Control 1 Register (SPMSC1)

This high page register contains status and control bits to support the low-voltage detect function, and to enable the bandgap voltage reference for use by the ADC and ACMP modules. This register should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVWF[1] | 0 | LVWIE | LVDRE[2] | LVDSE | LVDE[2] | 0 | BGBE |
| W | | LVWACK | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

= Unimplemented or Reserved

1 LVWF will be set in the case when $V_{Supply}$ transitions below the trip point or after reset and $V_{Supply}$ is already below $V_{LVW}$.

2 This bit can be written only one time after reset. Additional writes are ignored.

**Figure 5-9. System Power Management Status and Control 1 Register (SPMSC1)**

**Table 5-10. SPMSC1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LVWF | **Low-Voltage Warning Flag** — The LVWF bit indicates the low-voltage warning status.<br>0 low-voltage warning is not present.<br>1 low-voltage warning is present or was present. |
| 6<br>LVWACK | **Low-Voltage Warning Acknowledge** — If LVWF = 1, a low-voltage condition has occurred. To acknowledge this low-voltage warning, write 1 to LVWACK, which will automatically clear LVWF to 0 if the low-voltage warning is no longer present. |
| 5<br>LVWIE | **Low-Voltage Warning Interrupt Enable** — This bit enables hardware interrupt requests for LVWF.<br>0 Hardware interrupt disabled (use polling).<br>1 Request a hardware interrupt when LVWF = 1. |
| 4<br>LVDRE | **Low-Voltage Detect Reset Enable** — This write-once bit enables LVD events to generate a hardware reset (provided LVDE = 1).<br>0 LVD events do not generate hardware resets.<br>1 Force an MCU reset when an enabled low-voltage detect event occurs. |
| 3<br>LVDSE | **Low-Voltage Detect Stop Enable** — Provided LVDE = 1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode.<br>0 Low-voltage detect disabled during stop mode.<br>1 Low-voltage detect enabled during stop mode. |
| 2<br>LVDE | **Low-Voltage Detect Enable** — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register.<br>0 LVD logic disabled.<br>1 LVD logic enabled. |
| 0<br>BGBE | **Bandgap Buffer Enable** — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC and ACMP modules on one of its internal channels.<br>0 Bandgap buffer disabled.<br>1 Bandgap buffer enabled. |

## 5.8.8 System Power Management Status and Control 2 Register (SPMSC2)

This register is used to report the status of the low-voltage warning function, and to configure the stop mode behavior of the MCU. This register should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | LVDV[1] | LVWV | PPDF | 0 | 0 | PPDC[2] |
| W | | | | | | PPDACK | | |
| Power-on Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LVD Reset: | 0 | 0 | u | u | 0 | 0 | 0 | 0 |
| Any other Reset: | 0 | 0 | u | u | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved  u = Unaffected by reset

1 This bit can be written only one time after power-on reset. Additional writes are ignored.

2 This bit can be written only one time after reset. Additional writes are ignored.

**Figure 5-10. System Power Management Status and Control 2 Register (SPMSC2)**

**Table 5-11. SPMSC2 Register Field Descriptions**

| Field | Description |
|---|---|
| 5<br>LVDV | **Low-Voltage Detect Voltage Select** — This write-once bit selects the low-voltage detect (LVD) trip point setting. It also selects the warning voltage range. See Table 5-12. |
| 4<br>LVWV | **Low-Voltage Warning Voltage Select** — This bit selects the low-voltage warning (LVW) trip point voltage. See Table 5-12. |
| 3<br>PPDF | **Partial Power Down Flag** — This read-only status bit indicates that the MCU has recovered from stop2 mode.<br>0 MCU has not recovered from stop2 mode.<br>1 MCU recovered from stop2 mode. |
| 2<br>PPDACK | **Partial Power Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF bit. |
| 0<br>PPDC | **Partial Power Down Control** — This write-once bit controls whether stop2 or stop3 mode is selected.<br>0 Stop3 mode enabled.<br>1 Stop2, partial power down, mode enabled. |

**Table 5-12. LVD and LVW Trip Point Typical Values[1]**

| LVDV:LVWV | LVW Trip Point | LVD Trip Point |
|---|---|---|
| 0:0 | $V_{LVW0}$ = 2.74 V | $V_{LVD0}$ = 2.56 V |
| 0:1 | $V_{LVW1}$ = 2.92 V | |
| 1:0 | $V_{LVW2}$ = 4.3 V | $V_{LVD1}$ = 4.0 V |
| 1:1 | $V_{LVW3}$ = 4.6 V | |

1 See Electrical Characteristics appendix for minimum and maximum values.

# Chapter 6
# Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9S08DZ60 Series has seven parallel I/O ports which include a total of up to 53 I/O pins and one input-only pin. See Chapter 2, “Pins and Connections,” for more information about pin assignments and external hardware considerations of these pins.

Many of these pins are shared with on-chip peripherals such as timer systems, communication systems, or pin interrupts as shown in Table 2-1. The peripheral modules have priority over the general-purpose I/O functions so that when a peripheral is enabled, the I/O functions associated with the shared pins are disabled.

After reset, the shared peripheral functions are disabled and the pins are configured as inputs (PTxDDn = 0). The pin control functions for each pin are configured as follows: slew rate control enabled (PTxSEn = 1), low drive strength selected (PTxDSn = 0), and internal pull-ups disabled (PTxPEn = 0).

**NOTE**

- Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user’s reset initialization routine in the application program must either enable on-chip pull-up devices or change the direction of unconnected pins to outputs so the pins do not float.
- The PTE1 pin does not contain a clamp diode to $V_{DD}$ and should not be driven above $V_{DD}$. The voltage measured on the internally pulled up PTE1 pin may be as low as $V_{DD} – 0.7$ V. The internal gates connected to this pin are pulled all the way to $V_{DD}$.

## 6.1 Port Data and Data Direction

Reading and writing of parallel I/Os are performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram shown in Figure 6-1.

The data direction control bit (PTxDDn) determines whether the output buffer for the associated pin is enabled, and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit will continue to control the source for reads of the port data register.

When a shared analog function is enabled for a pin, both the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PTxDDn = 0) and the input buffer is disabled.

In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority such that if both the digital and analog functions are enabled, the analog function controls the pin.

It is a good programming practice to write to the port data register before changing the direction of a port pin to become an output. This ensures that the pin will not be driven momentarily with an old data value that happened to be in the port data register.

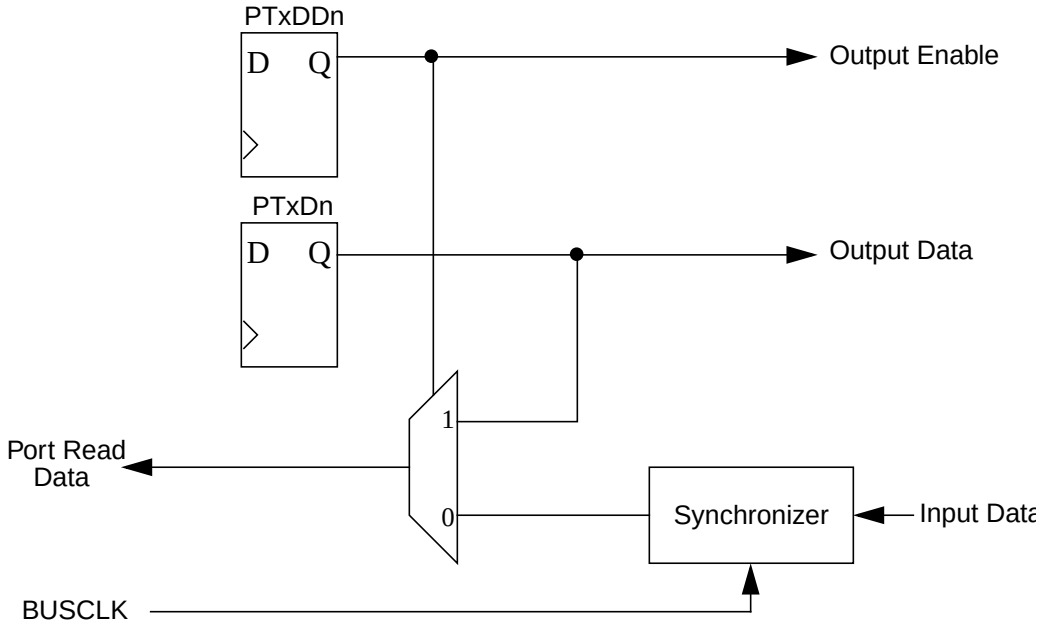



**Figure 6-1. Parallel I/O Block Diagram**

## 6.2 Pull-up, Slew Rate, and Drive Strength

Associated with the parallel I/O ports is a set of registers located in the high page register space that operate independently of the parallel I/O registers. These registers are used to control pull-ups, slew rate, and drive strength for the pins.

An internal pull-up device can be enabled for each port pin by setting the corresponding bit in the pull-up enable register (PTxPEn). The pull-up device is disabled if the pin is configured as an output by the parallel I/O control logic or any shared peripheral function regardless of the state of the corresponding pull-up enable register bit. The pull-up device is also disabled if the pin is controlled by an analog function.

Slew rate control can be enabled for each port pin by setting the corresponding bit in the slew rate control register (PTxSEn). When enabled, slew control limits the rate at which an output can transition in order to reduce EMC emissions. Slew rate control has no effect on pins that are configured as inputs.

**NOTE**

Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PTxDSn). When high drive is selected, a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the MCU are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this, the EMC emissions may be affected by enabling pins as high drive.

## 6.3 Pin Interrupts

Port A, port B, and port D pins can be configured as external interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes.

The block diagram for each port interrupt logic is shown Figure 6-2.

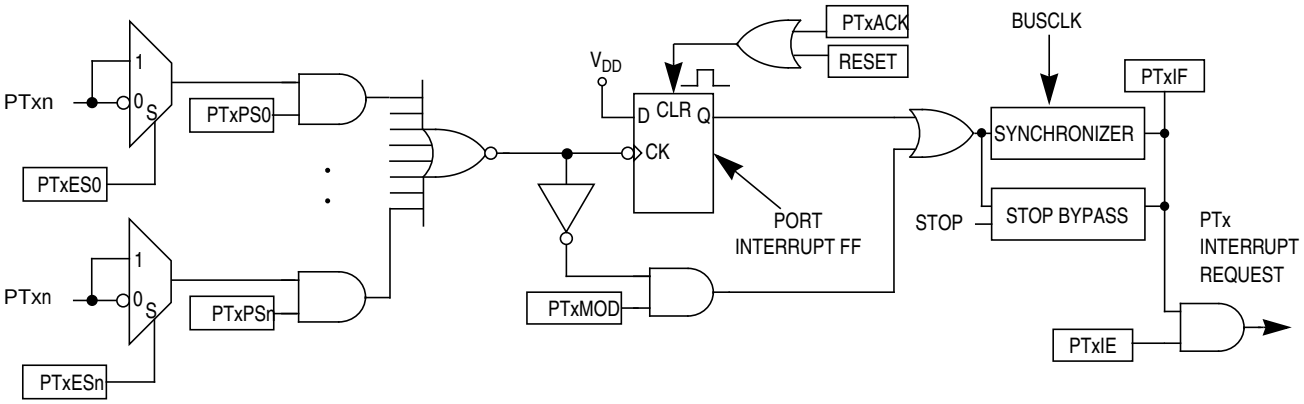



**Figure 6-2. Port Interrupt Block Diagram**

Writing to the PTxPSn bits in the port interrupt pin select register (PTxPS) independently enables or disables each port pin. Each port can be configured as edge sensitive or edge and level sensitive based on the PTxMOD bit in the port interrupt status and control register (PTxSC). Edge sensitivity can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the PTxESn bits in the port interrupt edge select register (PTxES).

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled port inputs must be at the deasserted logic level. A falling edge is detected when an enabled port input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

### 6.3.1 Edge Only Sensitivity

A valid edge on an enabled port pin will set PTxIF in PTxSC. If PTxIE in PTxSC is set, an interrupt request will be presented to the CPU. Clearing of PTxIF is accomplished by writing a 1 to PTxACK in PTxSC.

### 6.3.2 Edge and Level Sensitivity

A valid edge or level on an enabled port pin will set PTxIF in PTxSC. If PTxIE in PTxSC is set, an interrupt request will be presented to the CPU. Clearing of PTxIF is accomplished by writing a 1 to PTxACK in PTxSC provided all enabled port inputs are at their deasserted levels. PTxIF will remain set if any enabled port pin is asserted while attempting to clear by writing a 1 to PTxACK.

### 6.3.3 Pull-up/Pull-down Resistors

The port interrupt pins can be configured to use an internal pull-up/pull-down resistor using the associated I/O port pull-up enable register. If an internal resistor is enabled, the PTxES register is used to select whether the resistor is a pull-up (PTxESn = 0) or a pull-down (PTxESn = 1).

### 6.3.4 Pin Interrupt Initialization

When an interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization, the user should do the following:

1. Mask interrupts by clearing PTxIE in PTxSC.
2. Select the pin polarity by setting the appropriate PTxESn bits in PTxES.
3. If using internal pull-up/pull-down device, configure the associated pull enable bits in PTxPE.
4. Enable the interrupt pins by setting the appropriate PTxPSn bits in PTxPS.
5. Write to PTxACK in PTxSC to clear any false interrupts.
6. Set PTxIE in PTxSC to enable interrupts.

## 6.4 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the stop mode that is entered. An explanation of pin behavior for the various stop modes follows:

- Stop2 mode is a partial power-down mode, whereby I/O latches are maintained in their state as before the STOP instruction was executed. CPU register status and the state of I/O registers should be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode. Upon recovery from stop2 mode, before accessing any I/O, the user should examine the state of the PPDF bit in the SPMSC2 register. If the PPDF bit is 0, I/O must be initialized as if a power on reset had occurred. If the PPDF bit is 1, peripherals may require initialization to be restored to their pre-stop condition. This can be done using data previously stored in RAM if it was saved before the STOP instruction was executed. The user must then write a 1 to the PPDACK bit in the SPMSC2 register. Access to I/O is now permitted again in the user application program.
- In stop3 mode, all I/O is maintained because internal logic circuity stays powered up. Upon recovery, normal I/O function is available to the user.

## 6.5 Parallel I/O and Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports. The data and data direction registers are located in page zero of the memory map. The pull up, slew rate, drive strength, and interrupt control registers are located in the high page section of the memory map.

Refer to tables in Chapter 4, “Memory,” for the absolute address assignments for all parallel I/O and their pin control registers. This section refers to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

## 6.5.1 Port A Registers

Port A is controlled by the registers listed below.

### 6.5.1.1 Port A Data Register (PTAD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAD7 | PTAD6 | PTAD5 | PTAD4 | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-3. Port A Data Register (PTAD)**

**Table 6-1. PTAD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTAD[7:0] | **Port A Data Register Bits** — For port A pins that are inputs, reads return the logic level on the pin. For port A pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups/pull-downs disabled. |

### 6.5.1.2 Port A Data Direction Register (PTADD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTADD7 | PTADD6 | PTADD5 | PTADD4 | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-4. Port A Data Direction Register (PTADD)**

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

| Field | Description |
|---|---|
| 7:0<br>PTADD[7:0] | **Data Direction for Port A Bits** — These read/write bits control the direction of port A pins and what is read for PTAD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn. |

### 6.5.1.3 Port A Pull Enable Register (PTAPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAPE7 | PTAPE6 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-5. Internal Pull Enable for Port A Register (PTAPE)**

**Table 6-3. PTAPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTAPE[7:0] | **Internal Pull Enable for Port A Bits** — Each of these control bits determines if the internal pull-up or pull-down device is enabled for the associated PTA pin. For port A pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up/pull-down device disabled for port A bit n.<br>1 Internal pull-up/pull-down device enabled for port A bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.1.4 Port A Slew Rate Enable Register (PTASE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTASE7 | PTASE6 | PTASE5 | PTASE4 | PTASE3 | PTASE2 | PTASE1 | PTASE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-6. Slew Rate Enable for Port A Register (PTASE)**

**Table 6-4. PTASE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTASE[7:0] | **Output Slew Rate Enable for Port A Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port A bit n.<br>1 Output slew rate control enabled for port A bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.1.5 Port A Drive Strength Selection Register (PTADS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTADS7 | PTADS6 | PTADS5 | PTADS4 | PTADS3 | PTADS2 | PTADS1 | PTADS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-7. Drive Strength Selection for Port A Register (PTADS)**

**Table 6-5. PTADS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTADS[7:0] | **Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port A bit n.<br>1 High output drive strength selected for port A bit n. |

### 6.5.1.6 Port A Interrupt Status and Control Register (PTASC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PTAIF | 0 | PTAIE | PTAMOD |
| W | | | | | | PTAACK | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-8. Port A Interrupt Status and Control Register (PTASC)**

**Table 6-6. PTASC Register Field Descriptions**

| Field | Description |
|---|---|
| 3<br>PTAIF | **Port A Interrupt Flag** — PTAIF indicates when a port A interrupt is detected. Writes have no effect on PTAIF.<br>0 No port A interrupt detected.<br>1 Port A interrupt detected. |
| 2<br>PTAACK | **Port A Interrupt Acknowledge** — Writing a 1 to PTAACK is part of the flag clearing mechanism. PTAACK always reads as 0. |
| 1<br>PTAIE | **Port A Interrupt Enable** — PTAIE determines whether a port A interrupt is requested.<br>0 Port A interrupt request not enabled.<br>1 Port A interrupt request enabled. |
| 0<br>PTAMOD | **Port A Detection Mode** — PTAMOD (along with the PTAES bits) controls the detection mode of the port A interrupt pins.<br>0 Port A pins detect edges only.<br>1 Port A pins detect both edges and levels. |

### 6.5.1.7 Port A Interrupt Pin Select Register (PTAPS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAPS7 | PTAPS6 | PTAPS5 | PTAPS4 | PTAPS3 | PTAPS2 | PTAPS1 | PTAPS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-9. Port A Interrupt Pin Select Register (PTAPS)**

**Table 6-7. PTAPS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTAPS[7:0] | **Port A Interrupt Pin Selects** — Each of the PTAPSn bits enable the corresponding port A interrupt pin.<br>0 Pin not enabled as interrupt.<br>1 Pin enabled as interrupt. |

### 6.5.1.8 Port A Interrupt Edge Select Register (PTAES)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAES7 | PTAES6 | PTAES5 | PTAES4 | PTAES3 | PTAES2 | PTAES1 | PTAES0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-10. Port A Edge Select Register (PTAES)**

**Table 6-8. PTAES Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTAES[7:0] | **Port A Edge Selects** — Each of the PTAESn bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.<br>0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation.<br>1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation. |

## 6.5.2 Port B Registers

Port B is controlled by the registers listed below.

### 6.5.2.1 Port B Data Register (PTBD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBD7 | PTBD6 | PTBD5 | PTBD4 | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-11. Port B Data Register (PTBD)**

**Table 6-9. PTBD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBD[7:0] | **Port B Data Register Bits** — For port B pins that are inputs, reads return the logic level on the pin. For port B pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port B pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTBD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups/pull-downs disabled. |

### 6.5.2.2 Port B Data Direction Register (PTBDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBDD7 | PTBDD6 | PTBDD5 | PTBDD4 | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-12. Port B Data Direction Register (PTBDD)**

**Table 6-10. PTBDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBDD[7:0] | **Data Direction for Port B Bits** — These read/write bits control the direction of port B pins and what is read for PTBD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn. |

### 6.5.2.3 Port B Pull Enable Register (PTBPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBPE7 | PTBPE6 | PTBPE5 | PTBPE4 | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-13. Internal Pull Enable for Port B Register (PTBPE)**

**Table 6-11. PTBPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBPE[7:0] | **Internal Pull Enable for Port B Bits** — Each of these control bits determines if the internal pull-up or pull-down device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up/pull-down device disabled for port B bit n.<br>1 Internal pull-up/pull-down device enabled for port B bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.2.4 Port B Slew Rate Enable Register (PTBSE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBSE7 | PTBSE6 | PTBSE5 | PTBSE4 | PTBSE3 | PTBSE2 | PTBSE1 | PTBSE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-14. Slew Rate Enable for Port B Register (PTBSE)**

**Table 6-12. PTBSE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBSE[7:0] | **Output Slew Rate Enable for Port B Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port B bit n.<br>1 Output slew rate control enabled for port B bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.2.5 Port B Drive Strength Selection Register (PTBDS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBDS7 | PTBDS6 | PTBDS5 | PTBDS4 | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-15. Drive Strength Selection for Port B Register (PTBDS)**

**Table 6-13. PTBDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBDS[7:0] | **Output Drive Strength Selection for Port B Bits** — Each of these control bits selects between low and high output drive for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port B bit n.<br>1 High output drive strength selected for port B bit n. |

### 6.5.2.6 Port B Interrupt Status and Control Register (PTBSC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PTBIF | 0 | PTBIE | PTBMOD |
| W | | | | | | PTBACK | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-16. Port B Interrupt Status and Control Register (PTBSC)**

**Table 6-14. PTBSC Register Field Descriptions**

| Field | Description |
|---|---|
| 3<br>PTBIF | **Port B Interrupt Flag** — PTBIF indicates when a Port B interrupt is detected. Writes have no effect on PTBIF.<br>0 No Port B interrupt detected.<br>1 Port B interrupt detected. |
| 2<br>PTBACK | **Port B Interrupt Acknowledge** — Writing a 1 to PTBACK is part of the flag clearing mechanism. PTBACK always reads as 0. |
| 1<br>PTBIE | **Port B Interrupt Enable** — PTBIE determines whether a port B interrupt is requested.<br>0 Port B interrupt request not enabled.<br>1 Port B interrupt request enabled. |
| 0<br>PTBMOD | **Port B Detection Mode** — PTBMOD (along with the PTBES bits) controls the detection mode of the port B interrupt pins.<br>0 Port B pins detect edges only.<br>1 Port B pins detect both edges and levels. |

### 6.5.2.7 Port B Interrupt Pin Select Register (PTBPS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBPS7 | PTBPS6 | PTBPS5 | PTBPS4 | PTBPS3 | PTBPS2 | PTBPS1 | PTBPS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-17. Port B Interrupt Pin Select Register (PTBPS)**

**Table 6-15. PTBPS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBPS[7:0] | **Port B Interrupt Pin Selects** — Each of the PTBPSn bits enable the corresponding port B interrupt pin.<br>0 Pin not enabled as interrupt.<br>1 Pin enabled as interrupt. |

### 6.5.2.8 Port B Interrupt Edge Select Register (PTBES)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBES7 | PTBES6 | PTBES5 | PTBES4 | PTBES3 | PTBES2 | PTBES1 | PTBES0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-18. Port B Edge Select Register (PTBES)**

**Table 6-16. PTBES Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBES[7:0] | **Port B Edge Selects** — Each of the PTBESn bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.<br>0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation.<br>1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation. |

## 6.5.3 Port C Registers

Port C is controlled by the registers listed below.

### 6.5.3.1 Port C Data Register (PTCD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCD7 | PTCD6 | PTCD5 | PTCD4 | PTCD3 | PTCD2 | PTCD1 | PTCD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-19. Port C Data Register (PTCD)**

**Table 6-17. PTCD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCD[7:0] | **Port C Data Register Bits** — For port C pins that are inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port C pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTCD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled. |

### 6.5.3.2 Port C Data Direction Register (PTCDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCDD7 | PTCDD6 | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-20. Port C Data Direction Register (PTCDD)**

**Table 6-18. PTCDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCDD[7:0] | **Data Direction for Port C Bits** — These read/write bits control the direction of port C pins and what is read for PTCD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port C bit n and PTCD reads return the contents of PTCDn. |

### 6.5.3.3 Port C Pull Enable Register (PTCPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCPE7 | PTCPE6 | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-21. Internal Pull Enable for Port C Register (PTCPE)**

**Table 6-19. PTCPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCPE[7:0] | **Internal Pull Enable for Port C Bits** — Each of these control bits determines if the internal pull-up device is enabled for the associated PTC pin. For port C pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up device disabled for port C bit n.<br>1 Internal pull-up device enabled for port C bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.3.4 Port C Slew Rate Enable Register (PTCSE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCSE7 | PTCSE6 | PTCSE5 | PTCSE4 | PTCSE3 | PTCSE2 | PTCSE1 | PTCSE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

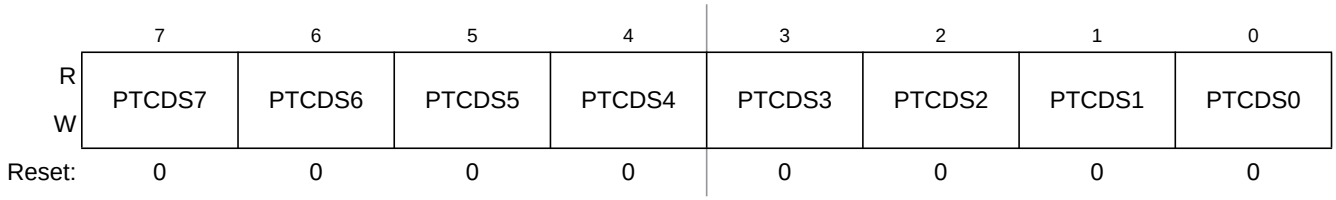
**Figure 6-22. Slew Rate Enable for Port C Register (PTCSE)**

**Table 6-20. PTCSE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCSE[7:0] | **Output Slew Rate Enable for Port C Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port C bit n.<br>1 Output slew rate control enabled for port C bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.3.5 Port C Drive Strength Selection Register (PTCDS)



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCDS7 | PTCDS6 | PTCDS5 | PTCDS4 | PTCDS3 | PTCDS2 | PTCDS1 | PTCDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-23. Drive Strength Selection for Port C Register (PTCDS)**

**Table 6-21. PTCDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCDS[7:0] | **Output Drive Strength Selection for Port C Bits** — Each of these control bits selects between low and high output drive for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port C bit n.<br>1 High output drive strength selected for port C bit n. |

## 6.5.4 Port D Registers

Port D is controlled by the registers listed below.

### 6.5.4.1 Port D Data Register (PTDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDD7 | PTDD6 | PTDD5 | PTDD4 | PTDD3 | PTDD2 | PTDD1 | PTDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-24. Port D Data Register (PTDD)**

**Table 6-22. PTDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDD[7:0] | **Port D Data Register Bits** — For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port D pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups/pull-downs disabled. |

### 6.5.4.2 Port D Data Direction Register (PTDDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDDD7 | PTDDD6 | PTDDD5 | PTDDD4 | PTDDD3 | PTDDD2 | PTDDD1 | PTDDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-25. Port D Data Direction Register (PTDDD)**

**Table 6-23. PTDDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDDD[7:0] | **Data Direction for Port D Bits** — These read/write bits control the direction of port D pins and what is read for PTDD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port D bit n and PTDD reads return the contents of PTDDn. |

### 6.5.4.3 Port D Pull Enable Register (PTDPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDPE7 | PTDPE6 | PTDPE5 | PTDPE4 | PTDPE3 | PTDPE2 | PTDPE1 | PTDPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-26. Internal Pull Enable for Port D Register (PTDPE)**

**Table 6-24. PTDPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDPE[7:0] | **Internal Pull Enable for Port D Bits** — Each of these control bits determines if the internal pull-up or pull-down device is enabled for the associated PTD pin. For port D pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up/pull-down device disabled for port D bit n.<br>1 Internal pull-up/pull-down device enabled for port D bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.4.4 Port D Slew Rate Enable Register (PTDSE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDSE7 | PTDSE6 | PTDSE5 | PTDSE4 | PTDSE3 | PTDSE2 | PTDSE1 | PTDSE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-27. Slew Rate Enable for Port D Register (PTDSE)**

**Table 6-25. PTDSE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDSE[7:0] | **Output Slew Rate Enable for Port D Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTD pin. For port D pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port D bit n.<br>1 Output slew rate control enabled for port D bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.4.5 Port D Drive Strength Selection Register (PTDDS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDDS7 | PTDDS6 | PTDDS5 | PTDDS4 | PTDDS3 | PTDDS2 | PTDDS1 | PTDDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-28. Drive Strength Selection for Port D Register (PTDDS)**

**Table 6-26. PTDDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDDS[7:0] | **Output Drive Strength Selection for Port D Bits** — Each of these control bits selects between low and high output drive for the associated PTD pin. For port D pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port D bit n.<br>1 High output drive strength selected for port D bit n. |

### 6.5.4.6 Port D Interrupt Status and Control Register (PTDSC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PTDIF | 0 | PTDIE | PTDMOD |
| W | | | | | | PTDACK | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-29. Port D Interrupt Status and Control Register (PTDSC)**

**Table 6-27. PTDSC Register Field Descriptions**

| Field | Description |
|---|---|
| 3<br>PTDIF | **Port D Interrupt Flag** — PTDIF indicates when a port D interrupt is detected. Writes have no effect on PTDIF.<br>0 No port D interrupt detected.<br>1 Port D interrupt detected. |
| 2<br>PTDACK | **Port D Interrupt Acknowledge** — Writing a 1 to PTDACK is part of the flag clearing mechanism. PTDACK always reads as 0. |
| 1<br>PTDIE | **Port D Interrupt Enable** — PTDIE determines whether a port D interrupt is requested.<br>0 Port D interrupt request not enabled.<br>1 Port D interrupt request enabled. |
| 0<br>PTDMOD | **Port A Detection Mode** — PTDMOD (along with the PTDES bits) controls the detection mode of the port D interrupt pins.<br>0 Port D pins detect edges only.<br>1 Port D pins detect both edges and levels. |

### 6.5.4.7 Port D Interrupt Pin Select Register (PTDPS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDPS7 | PTDPS6 | PTDPS5 | PTDPS4 | PTDPS3 | PTDPS2 | PTDPS1 | PTDPS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-30. Port D Interrupt Pin Select Register (PTDPS)**

**Table 6-28. PTDPS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDPS[7:0] | **Port D Interrupt Pin Selects** — Each of the PTDPSn bits enable the corresponding port D interrupt pin.<br>0 Pin not enabled as interrupt.<br>1 Pin enabled as interrupt. |

### 6.5.4.8 Port D Interrupt Edge Select Register (PTDES)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTDES7 | PTDES6 | PTDES5 | PTDES4 | PTDES3 | PTDES2 | PTDES1 | PTDES0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-31. Port D Edge Select Register (PTDES)**

**Table 6-29. PTDES Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTDES[7:0] | **Port D Edge Selects** — Each of the PTDESn bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled.<br>0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation.<br>1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation. |

## 6.5.5 Port E Registers

Port E is controlled by the registers listed below.

### 6.5.5.1 Port E Data Register (PTED)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTED7 | PTED6 | PTED5 | PTED4 | PTED3 | PTED2 | PTED1[1] | PTED0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-32. Port E Data Register (PTED)**

1 Reads of this bit always return the pin value of the associated pin, regardless of the value stored in the port data direction bit.

**Table 6-30. PTED Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTED[7:0] | **Port E Data Register Bits** — For port E pins that are inputs, reads return the logic level on the pin. For port E pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port E pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTED to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled. |

### 6.5.5.2 Port E Data Direction Register (PTEDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTEDD7 | PTEDD6 | PTEDD5 | PTEDD4 | PTEDD3 | PTEDD2 | PTEDD1[1] | PTEDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-33. Port E Data Direction Register (PTEDD)**

1 PTEDD1 has no effect on the input-only PTE1 pin.

**Table 6-31. PTEDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTEDD[7:0] | **Data Direction for Port E Bits** — These read/write bits control the direction of port E pins and what is read for PTED reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port E bit n and PTED reads return the contents of PTEDn. |

### 6.5.5.3 Port E Pull Enable Register (PTEPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTEPE7 | PTEPE6 | PTEPE5 | PTEPE4 | PTEPE3 | PTEPE2 | PTEPE1 | PTEPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-34. Internal Pull Enable for Port E Register (PTEPE)**

**Table 6-32. PTEPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTEPE[7:0] | **Internal Pull Enable for Port E Bits** — Each of these control bits determines if the internal pull-up device is enabled for the associated PTE pin. For port E pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up device disabled for port E bit n.<br>1 Internal pull-up device enabled for port E bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.5.4 Port E Slew Rate Enable Register (PTESE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTESE7 | PTESE6 | PTESE5 | PTESE4 | PTESE3 | PTESE2 | PTESE1[1] | PTESE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-35. Slew Rate Enable for Port E Register (PTESE)**

1 PTESE1 has no effect on the input-only PTE1 pin.

**Table 6-33. PTESE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTESE[7:0] | **Output Slew Rate Enable for Port E Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTE pin. For port E pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port E bit n.<br>1 Output slew rate control enabled for port E bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.5.5 Port E Drive Strength Selection Register (PTEDS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTEDS7 | PTEDS6 | PTEDS5 | PTEDS4 | PTEDS3 | PTEDS2 | PTEDS1[1] | PTEDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-36. Drive Strength Selection for Port E Register (PTEDS)**

1 PTEDS1 has no effect on the input-only PTE1 pin.

**Table 6-34. PTEDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTEDS[7:0] | **Output Drive Strength Selection for Port E Bits** — Each of these control bits selects between low and high output drive for the associated PTE pin. For port E pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port E bit n.<br>1 High output drive strength selected for port E bit n. |

## 6.5.6 Port F Registers

Port F is controlled by the registers listed below.

### 6.5.6.1 Port F Data Register (PTFD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTFD7 | PTFD6 | PTFD5 | PTFD4 | PTFD3 | PTFD2 | PTFD1 | PTFD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-37. Port F Data Register (PTFD)**

**Table 6-35. PTFD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTFD[7:0] | **Port F Data Register Bits** — For port F pins that are inputs, reads return the logic level on the pin. For port F pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port F pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTFD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled. |

### 6.5.6.2 Port F Data Direction Register (PTFDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTFDD7 | PTFDD6 | PTFDD5 | PTFDD4 | PTFDD3 | PTFDD2 | PTFDD1 | PTFDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-38. Port F Data Direction Register (PTFDD)**

**Table 6-36. PTFDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTFDD[7:0] | **Data Direction for Port F Bits** — These read/write bits control the direction of port F pins and what is read for PTFD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port F bit n and PTFD reads return the contents of PTFDn. |

### 6.5.6.3 Port F Pull Enable Register (PTFPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTFPE7 | PTFPE6 | PTFPE5 | PTFPE4 | PTFPE3 | PTFPE2 | PTFPE1 | PTFPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-39. Internal Pull Enable for Port F Register (PTFPE)**

**Table 6-37. PTFPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTFPE[7:0] | **Internal Pull Enable for Port F Bits** — Each of these control bits determines if the internal pull-up device is enabled for the associated PTF pin. For port F pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up device disabled for port F bit n.<br>1 Internal pull-up device enabled for port F bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.6.4 Port F Slew Rate Enable Register (PTFSE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTFSE7 | PTFSE6 | PTFSE5 | PTFSE4 | PTFSE3 | PTFSE2 | PTFSE1 | PTFSE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-40. Slew Rate Enable for Port F Register (PTFSE)**

**Table 6-38. PTFSE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTFSE[7:0] | **Output Slew Rate Enable for Port F Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port F bit n.<br>1 Output slew rate control enabled for port F bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.6.5 Port F Drive Strength Selection Register (PTFDS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTFDS7 | PTFDS6 | PTFDS5 | PTFDS4 | PTFDS3 | PTFDS2 | PTFDS1 | PTFDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-41. Drive Strength Selection for Port F Register (PTFDS)**

**Table 6-39. PTFDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTFDS[7:0] | **Output Drive Strength Selection for Port F Bits** — Each of these control bits selects between low and high output drive for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port F bit n.<br>1 High output drive strength selected for port F bit n. |

## 6.5.7 Port G Registers

Port G is controlled by the registers listed below.

### 6.5.7.1 Port G Data Register (PTGD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTGD5 | PTGD4 | PTGD3 | PTGD2 | PTGD1 | PTGD0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-42. Port G Data Register (PTGD)**

**Table 6-40. PTGD Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTGD[5:0] | **Port G Data Register Bits** — For port G pins that are inputs, reads return the logic level on the pin. For port G pins that are configured as outputs, reads return the last value written to this register.<br>Writes are latched into all bits of this register. For port G pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>Reset forces PTGD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled. |

### 6.5.7.2 Port G Data Direction Register (PTGDD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTGDD5 | PTGDD4 | PTGDD3 | PTGDD2 | PTGDD1 | PTGDD0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-43. Port G Data Direction Register (PTGDD)**

**Table 6-41. PTGDD Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTGDD[5:0] | **Data Direction for Port G Bits** — These read/write bits control the direction of port G pins and what is read for PTGD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port G bit n and PTGD reads return the contents of PTGDn. |

### 6.5.7.3 Port G Pull Enable Register (PTGPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTGPE5 | PTGPE4 | PTGPE3 | PTGPE2 | PTGPE1 | PTGPE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-44. Internal Pull Enable for Port G Register (PTGPE)**

**Table 6-42. PTGPE Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTGPE[5:0] | **Internal Pull Enable for Port G Bits** — Each of these control bits determines if the internal pull-up device is enabled for the associated PTG pin. For port G pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pull-up device disabled for port G bit n.<br>1 Internal pull-up device enabled for port G bit n. |

**NOTE**

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

### 6.5.7.4 Port G Slew Rate Enable Register (PTGSE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTGSE5 | PTGSE4 | PTGSE3 | PTGSE2 | PTGSE1 | PTGSE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-45. Slew Rate Enable for Port G Register (PTGSE)**

**Table 6-43. PTGSE Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTGSE[5:0] | **Output Slew Rate Enable for Port G Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTG pin. For port G pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for port G bit n.<br>1 Output slew rate control enabled for port G bit n. |

**Note:** Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

### 6.5.7.5 Port G Drive Strength Selection Register (PTGDS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTGDS5 | PTGDS4 | PTGDS3 | PTGDS2 | PTGDS1 | PTGDS0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-46. Drive Strength Selection for Port G Register (PTGDS)**

**Table 6-44. PTGDS Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTGDS[5:0 | **Output Drive Strength Selection for Port G Bits** — Each of these control bits selects between low and high output drive for the associated PTG pin. For port G pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port G bit n.<br>1 High output drive strength selected for port G bit n. |

# Chapter 7
# Central Processor Unit (S08CPUV3)

## 7.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family. For a more detailed discussion, refer to the *HCS08 Family Reference Manual, volume 1*, Freescale Semiconductor document order number HCS08RMV1/D.

The HCS08 CPU is fully source- and object-code-compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes were added to improve C compiler efficiency and to support a new background debug system which replaces the monitor mode of earlier M68HC08 microcontrollers (MCU).

### 7.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- All registers and memory are mapped to a single 64-Kbyte address space
- 16-bit stack pointer (any size stack anywhere in 64-Kbyte address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
  - Inherent — Operands in internal registers
  - Relative — 8-bit signed offset to branch destination
  - Immediate — Operand in next object code byte(s)
  - Direct — Operand in memory at 0x0000–0x00FF
  - Extended — Operand anywhere in 64-Kbyte address space
  - Indexed relative to H:X — Five submodes including auto increment
  - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- STOP and WAIT instructions to invoke low-power operating modes

## 7.2 Programmer's Model and CPU Registers

Figure 7-1 shows the five CPU registers. CPU registers are not part of the memory map.

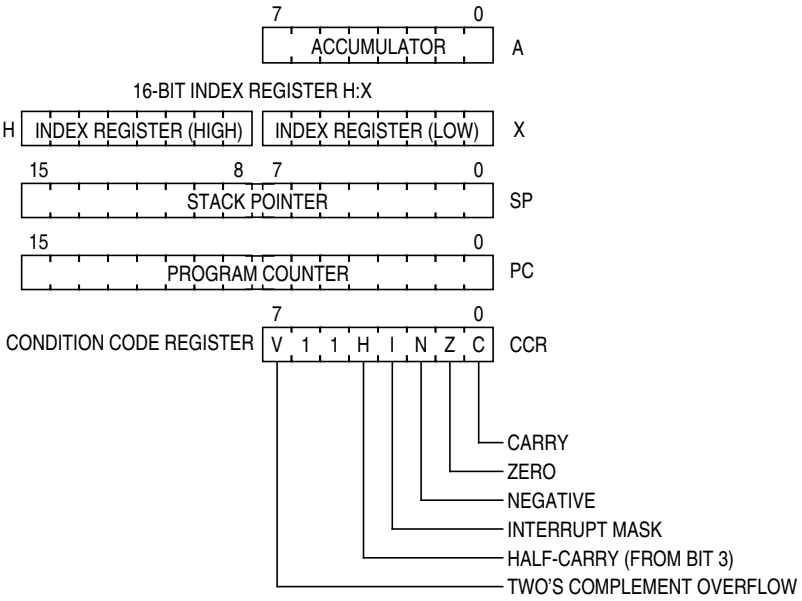



**Figure 7-1. CPU Registers**

### 7.2.1 Accumulator (A)

The A accumulator is a general-purpose 8-bit register. One operand input to the arithmetic logic unit (ALU) is connected to the accumulator and the ALU results are often stored into the A accumulator after arithmetic and logical operations. The accumulator can be loaded from memory using various addressing modes to specify the address where the loaded data comes from, or the contents of A can be stored to memory using various addressing modes to specify the address where data from A will be stored.

Reset has no effect on the contents of the A accumulator.

### 7.2.2 Index Register (H:X)

This 16-bit register is actually two separate 8-bit registers (H and X), which often work together as a 16-bit address pointer where H holds the upper byte of an address and X holds the lower byte of the address. All indexed addressing mode instructions use the full 16-bit value in H:X as an index reference pointer; however, for compatibility with the earlier M68HC05 Family, some instructions operate only on the low-order 8-bit half (X).

Many instructions treat X as a second general-purpose 8-bit register that can be used to hold 8-bit data values. X can be cleared, incremented, decremented, complemented, negated, shifted, or rotated. Transfer instructions allow data to be transferred from A or transferred to A where arithmetic and logical operations can then be performed.

For compatibility with the earlier M68HC05 Family, H is forced to 0x00 during reset. Reset has no effect on the contents of X.

### 7.2.3 Stack Pointer (SP)

This 16-bit address pointer register points at the next available location on the automatic last-in-first-out (LIFO) stack. The stack may be located anywhere in the 64-Kbyte address space that has RAM and can be any size up to the amount of available RAM. The stack is used to automatically save the return address for subroutine calls, the return address and CPU registers during interrupts, and for local variables. The AIS (add immediate to stack pointer) instruction adds an 8-bit signed immediate value to SP. This is most often used to allocate or deallocate space for local variables on the stack.

SP is forced to 0x00FF at reset for compatibility with the earlier M68HC05 Family. HCS08 programs normally change the value in SP to the address of the last location (highest address) in on-chip RAM during reset initialization to free up direct page RAM (from the end of the on-chip registers to 0x00FF).

The RSP (reset stack pointer) instruction was included for compatibility with the M68HC05 Family and is seldom used in new HCS08 programs because it only affects the low-order half of the stack pointer.

### 7.2.4 Program Counter (PC)

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

During normal program execution, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, interrupt, and return operations load the program counter with an address other than that of the next sequential location. This is called a change-of-flow.

During reset, the program counter is loaded with the reset vector that is located at 0xFFFE and 0xFFFF. The vector stored there is the address of the first instruction that will be executed after exiting the reset state.

### 7.2.5 Condition Code Register (CCR)

The 8-bit condition code register contains the interrupt mask (I) and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to 1. The following paragraphs describe the functions of the condition code bits in general terms. For a more detailed explanation of how each instruction sets the CCR bits, refer to the *HCS08 Family Reference Manual, volume 1,* Freescale Semiconductor document order number HCS08RMv1.

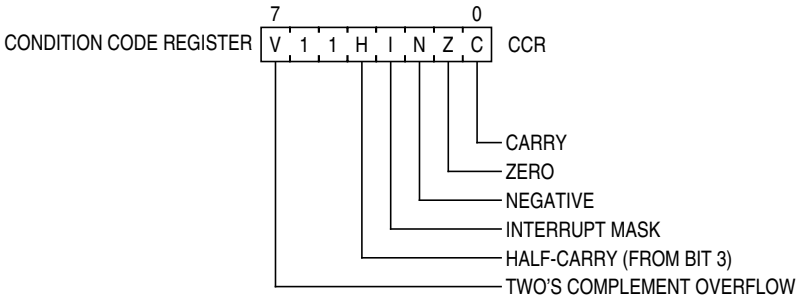



**Figure 7-2. Condition Code Register**

**Table 7-1. CCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>V | **Two's Complement Overflow Flag** — The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag.<br>0 No overflow<br>1 Overflow |
| 4<br>H | **Half-Carry Flag** — The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C condition code bits to automatically add a correction value to the result from a previous ADD or ADC on BCD operands to correct the result to a valid BCD value.<br>0 No carry between bits 3 and 4<br>1 Carry between bits 3 and 4 |
| 3<br>I | **Interrupt Mask Bit** — When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the first instruction of the interrupt service routine is executed.<br>Interrupts are not recognized at the instruction boundary after any instruction that clears I (CLI or TAP). This ensures that the next instruction after a CLI or TAP will always be executed without the possibility of an intervening interrupt, provided I was set.<br>0 Interrupts enabled<br>1 Interrupts disabled |
| 2<br>N | **Negative Flag** — The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result. Simply loading or storing an 8-bit or 16-bit value causes N to be set if the most significant bit of the loaded or stored value was 1.<br>0 Non-negative result<br>1 Negative result |
| 1<br>Z | **Zero Flag** — The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of 0x00 or 0x0000. Simply loading or storing an 8-bit or 16-bit value causes Z to be set if the loaded or stored value was all 0s.<br>0 Non-zero result<br>1 Zero result |
| 0<br>C | **Carry/Borrow Flag** — The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag.<br>0 No carry out of bit 7<br>1 Carry out of bit 7 |

## 7.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, all memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte linear address space so a 16-bit binary address can uniquely identify any memory location. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

### 7.3.1 Inherent Addressing Mode (INH)

In this addressing mode, operands needed to complete the instruction (if any) are located within CPU registers so the CPU does not need to access memory to get any operands.

### 7.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the destination location for branch instructions. A signed 8-bit offset value is located in the memory location immediately following the opcode. During execution, if the branch condition is true, the signed offset is sign-extended to a 16-bit value and is added to the current contents of the program counter, which causes program execution to continue at the branch destination address.

### 7.3.3 Immediate Addressing Mode (IMM)

In immediate addressing mode, the operand needed to complete the instruction is included in the object code immediately following the instruction opcode in memory. In the case of a 16-bit immediate operand, the high-order byte is located in the next memory location after the opcode, and the low-order byte is located in the next memory location after that.

### 7.3.4 Direct Addressing Mode (DIR)

In direct addressing mode, the instruction includes the low-order eight bits of an address in the direct page (0x0000–0x00FF). During execution a 16-bit address is formed by concatenating an implied 0x00 for the high-order half of the address and the direct address from the instruction to get the 16-bit address where the desired operand is located. This is faster and more memory efficient than specifying a complete 16-bit address for the operand.

### 7.3.5 Extended Addressing Mode (EXT)

In extended addressing mode, the full 16-bit address of the operand is located in the next two bytes of program memory after the opcode (high byte first).

### 7.3.6 Indexed Addressing Mode

Indexed addressing mode has seven variations including five that use the 16-bit H:X index register pair and two that use the stack pointer as the base reference.

#### 7.3.6.1 Indexed, No Offset (IX)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction.

#### 7.3.6.2 Indexed, No Offset with Post Increment (IX+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction. The index register pair is then incremented (H:X = H:X + 0x0001) after the operand has been fetched. This addressing mode is only used for MOV and CBEQ instructions.

#### 7.3.6.3 Indexed, 8-Bit Offset (IX1)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

#### 7.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction. The index register pair is then incremented (H:X = H:X + 0x0001) after the operand has been fetched. This addressing mode is used only for the CBEQ instruction.

#### 7.3.6.5 Indexed, 16-Bit Offset (IX2)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

#### 7.3.6.6 SP-Relative, 8-Bit Offset (SP1)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

#### 7.3.6.7 SP-Relative, 16-Bit Offset (SP2)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

## 7.4 Special Operations

The CPU performs a few special operations that are similar to instructions but do not have opcodes like other CPU instructions. In addition, a few instructions such as STOP and WAIT directly affect other MCU circuitry. This section provides additional information about these operations.

### 7.4.1 Reset Sequence

Reset can be caused by a power-on-reset (POR) event, internal conditions such as the COP (computer operating properly) watchdog, or by assertion of an external active-low reset pin. When a reset event occurs, the CPU immediately stops whatever it is doing (the MCU does not wait for an instruction boundary before responding to a reset event). For a more detailed discussion about how the MCU recognizes resets and determines the source, refer to the Resets, Interrupts, and System Configuration chapter.

The reset event is considered concluded when the sequence to determine whether the reset came from an internal source is done and when the reset pin is no longer asserted. At the conclusion of a reset event, the CPU performs a 6-cycle sequence to fetch the reset vector from 0xFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

### 7.4.2 Interrupt Sequence

When an interrupt is requested, the CPU completes the current instruction before responding to the interrupt. At this point, the program counter is pointing at the start of the next instruction, which is where the CPU should return after servicing the interrupt. The CPU responds to an interrupt by performing the same sequence of operations as for a software interrupt (SWI) instruction, except the address used for the vector fetch is determined by the highest priority interrupt that is pending when the interrupt sequence started.

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.
3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.
6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the

interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

For compatibility with the earlier M68HC05 MCUs, the high-order half of the H:X index register pair (H) is not saved on the stack as part of the interrupt sequence. The user must use a PSHH instruction at the beginning of the service routine to save H and then use a PULH instruction just before the RTI that ends the interrupt service routine. It is not necessary to save H if you are certain that the interrupt service routine does not use any instructions or auto-increment addressing modes that might change the value of H.

The software interrupt (SWI) instruction is like a hardware interrupt except that it is not masked by the global I bit in the CCR and it is associated with an instruction opcode within the program so it is not asynchronous to program execution.

### 7.4.3 Wait Mode Operation

The WAIT instruction enables interrupts by clearing the I bit in the CCR. It then halts the clocks to the CPU to reduce overall power consumption while the CPU is waiting for the interrupt or reset event that will wake the CPU from wait mode. When an interrupt or reset event occurs, the CPU clocks will resume and the interrupt or reset event will be processed normally.

If a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in wait mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in wait mode.

### 7.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in stop mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in stop mode.

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode. Refer to the Modes of Operation chapter for more details.

### 7.4.5 BGND Instruction

The BGND instruction is new to the HCS08 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

Software-based breakpoints can be set by replacing an opcode at the desired breakpoint address with the BGND opcode. When the program reaches this breakpoint address, the CPU is forced to active background mode rather than continuing the user program.

## 7.5 HCS08 Instruction Set Summary

Table 7-2 provides a summary of the HCS08 instruction set in all possible addressing modes. The table shows operand construction, execution time in internal bus clock cycles, and cycle-by-cycle details for each addressing mode variation of each instruction.

**Table 7-2. Instruction Set Summary (Sheet 1 of 9)**

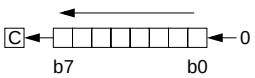
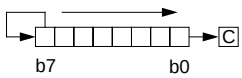
| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| ADC *#opr8i*<br>ADC *opr8a*<br>ADC *opr16a*<br>ADC *oprx16*,X<br>ADC *oprx8*,X<br>ADC ,X<br>ADC *oprx16*,SP<br>ADC *oprx8*,SP | Add with Carry<br>A ← (A) + (M) + (C) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A9 ii<br>B9 dd<br>C9 hh ll<br>D9 ee ff<br>E9 ff<br>F9<br>9E D9 ee ff<br>9E E9 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ 1 1 ↕ | – ↕ ↕ ↕ |
| ADD *#opr8i*<br>ADD *opr8a*<br>ADD *opr16a*<br>ADD *oprx16*,X<br>ADD *oprx8*,X<br>ADD ,X<br>ADD *oprx16*,SP<br>ADD *oprx8*,SP | Add without Carry<br>A ← (A) + (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AB ii<br>BB dd<br>CB hh ll<br>DB ee ff<br>EB ff<br>FB<br>9E DB ee ff<br>9E EB ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ 1 1 ↕ | – ↕ ↕ ↕ |
| AIS *#opr8i* | Add Immediate Value (Signed) to Stack Pointer<br>SP ← (SP) + (M) | IMM | A7 ii | 2 | pp | – 1 1 – | – – – – |
| AIX *#opr8i* | Add Immediate Value (Signed) to Index Register (H:X)<br>H:X ← (H:X) + (M) | IMM | AF ii | 2 | pp | – 1 1 – | – – – – |
| AND *#opr8i*<br>AND *opr8a*<br>AND *opr16a*<br>AND *oprx16*,X<br>AND *oprx8*,X<br>AND ,X<br>AND *oprx16*,SP<br>AND *oprx8*,SP | Logical AND<br>A ← (A) & (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A4 ii<br>B4 dd<br>C4 hh ll<br>D4 ee ff<br>E4 ff<br>F4<br>9E D4 ee ff<br>9E E4 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 1 1 – | – ↕ ↕ – |
| ASL *opr8a*<br>ASLA<br>ASLX<br>ASL *oprx8*,X<br>ASL ,X<br>ASL *oprx8*,SP | Arithmetic Shift Left<br>  <br><br>(Same as LSL) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38 dd<br>48<br>58<br>68 ff<br>78<br>9E 68 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| ASR *opr8a*<br>ASRA<br>ASRX<br>ASR *oprx8*,X<br>ASR ,X<br>ASR *oprx8*,SP | Arithmetic Shift Right<br>  <br> | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 37 dd<br>47<br>57<br>67 ff<br>77<br>9E 67 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ ↕ |

**Table 7-2. Instruction Set Summary (Sheet 2 of 9)**

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| BCC *rel* | Branch if Carry Bit Clear<br>(if C = 0) | REL | 24 rr | 3 | ppp | – 1 1 – | – – – – |
| BCLR *n*,*opr8a* | Clear Bit n in Memory<br>(Mn ← 0) | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 11 dd<br>13 dd<br>15 dd<br>17 dd<br>19 dd<br>1B dd<br>1D dd<br>1F dd | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 | rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp | – 1 1 – | – – – – |
| BCS *rel* | Branch if Carry Bit Set (if C = 1)<br>(Same as BLO) | REL | 25 rr | 3 | ppp | – 1 1 – | – – – – |
| BEQ *rel* | Branch if Equal (if Z = 1) | REL | 27 rr | 3 | ppp | – 1 1 – | – – – – |
| BGE *rel* | Branch if Greater Than or Equal To<br>(if N ⊕ V = 0) (Signed) | REL | 90 rr | 3 | ppp | – 1 1 – | – – – – |
| BGND | Enter active background if ENBDM=1<br>Waits for and processes BDM commands<br>until GO, TRACE1, or TAGGO | INH | 82 | 5+ | fp...ppp | – 1 1 – | – – – – |
| BGT *rel* | Branch if Greater Than (if Z \| (N ⊕ V) = 0)<br>(Signed) | REL | 92 rr | 3 | ppp | – 1 1 – | – – – – |
| BHCC *rel* | Branch if Half Carry Bit Clear (if H = 0) | REL | 28 rr | 3 | ppp | – 1 1 – | – – – – |
| BHCS *rel* | Branch if Half Carry Bit Set (if H = 1) | REL | 29 rr | 3 | ppp | – 1 1 – | – – – – |
| BHI *rel* | Branch if Higher (if C \| Z = 0) | REL | 22 rr | 3 | ppp | – 1 1 – | – – – – |
| BHS *rel* | Branch if Higher or Same (if C = 0)<br>(Same as BCC) | REL | 24 rr | 3 | ppp | – 1 1 – | – – – – |
| BIH *rel* | Branch if IRQ Pin High (if IRQ pin = 1) | REL | 2F rr | 3 | ppp | – 1 1 – | – – – – |
| BIL *rel* | Branch if IRQ Pin Low (if IRQ pin = 0) | REL | 2E rr | 3 | ppp | – 1 1 – | – – – – |
| BIT *#opr8i*<br>BIT *opr8a*<br>BIT *opr16a*<br>BIT *oprx16*,X<br>BIT *oprx8*,X<br>BIT ,X<br>BIT *oprx16*,SP<br>BIT *oprx8*,SP | Bit Test<br>(A) & (M)<br>(CCR Updated but Operands Not Changed) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A5 ii<br>B5 dd<br>C5 hh ll<br>D5 ee ff<br>E5 ff<br>F5<br>9E D5 ee ff<br>9E E5 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 1 1 – | – ↕ ↕ – |
| BLE *rel* | Branch if Less Than or Equal To<br>(if Z \| (N ⊕ V) = 1) (Signed) | REL | 93 rr | 3 | ppp | – 1 1 – | – – – – |
| BLO *rel* | Branch if Lower (if C = 1) (Same as BCS) | REL | 25 rr | 3 | ppp | – 1 1 – | – – – – |
| BLS *rel* | Branch if Lower or Same (if C \| Z = 1) | REL | 23 rr | 3 | ppp | – 1 1 – | – – – – |
| BLT *rel* | Branch if Less Than (if N ⊕ V = 1) (Signed) | REL | 91 rr | 3 | ppp | – 1 1 – | – – – – |
| BMC *rel* | Branch if Interrupt Mask Clear (if I = 0) | REL | 2C rr | 3 | ppp | – 1 1 – | – – – – |
| BMI *rel* | Branch if Minus (if N = 1) | REL | 2B rr | 3 | ppp | – 1 1 – | – – – – |
| BMS *rel* | Branch if Interrupt Mask Set (if I = 1) | REL | 2D rr | 3 | ppp | – 1 1 – | – – – – |
| BNE *rel* | Branch if Not Equal (if Z = 0) | REL | 26 rr | 3 | ppp | – 1 1 – | – – – – |

## Table 7-2. Instruction Set Summary (Sheet 3 of 9)

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| BPL *rel* | Branch if Plus (if N = 0) | REL | 2A rr | 3 | ppp | – 1 1 – | – – – – |
| BRA *rel* | Branch Always (if I = 1) | REL | 20 rr | 3 | ppp | – 1 1 – | – – – – |
| BRCLR *n*,*opr8a*,*rel* | Branch if Bit *n* in Memory Clear (if (Mn) = 0) | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 01 dd rr<br>03 dd rr<br>05 dd rr<br>07 dd rr<br>09 dd rr<br>0B dd rr<br>0D dd rr<br>0F dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 | rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp | – 1 1 – | – – – ↕ |
| BRN *rel* | Branch Never (if I = 0) | REL | 21 rr | 3 | ppp | – 1 1 – | – – – – |
| BRSET *n*,*opr8a*,*rel* | Branch if Bit *n* in Memory Set (if (Mn) = 1) | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 00 dd rr<br>02 dd rr<br>04 dd rr<br>06 dd rr<br>08 dd rr<br>0A dd rr<br>0C dd rr<br>0E dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 | rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp<br>rpppp | – 1 1 – | – – – ↕ |
| BSET *n*,*opr8a* | Set Bit *n* in Memory (Mn ← 1) | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 10 dd<br>12 dd<br>14 dd<br>16 dd<br>18 dd<br>1A dd<br>1C dd<br>1E dd | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 | rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp<br>rfwpp | – 1 1 – | – – – – |
| BSR *rel* | Branch to Subroutine<br>PC ← (PC) + $0002<br>push (PCL); SP ← (SP) – $0001<br>push (PCH); SP ← (SP) – $0001<br>PC ← (PC) + *rel* | REL | AD rr | 5 | ssppp | – 1 1 – | – – – – |
| CBEQ *opr8a*,*rel*<br>CBEQA #*opr8i*,*rel*<br>CBEQX #*opr8i*,*rel*<br>CBEQ *oprx8*,X+,*rel*<br>CBEQ ,X+,*rel*<br>CBEQ *oprx8*,SP,*rel* | Compare and... Branch if (A) = (M)<br>Branch if (A) = (M)<br>Branch if (X) = (M)<br>Branch if (A) = (M)<br>Branch if (A) = (M)<br>Branch if (A) = (M) | DIR<br>IMM<br>IMM<br>IX1+<br>IX+<br>SP1 | 31 dd rr<br>41 ii rr<br>51 ii rr<br>61 ff rr<br>71 rr<br>9E 61 ff rr | 5<br>4<br>4<br>5<br>5<br>6 | rpppp<br>pppp<br>pppp<br>rpppp<br>rfppp<br>prpppp | – 1 1 – | – – – – |
| CLC | Clear Carry Bit (C ← 0) | INH | 98 | 1 | p | – 1 1 – | – – – 0 |
| CLI | Clear Interrupt Mask Bit (I ← 0) | INH | 9A | 1 | p | – 1 1 – | 0 – – – |
| CLR *opr8a*<br>CLRA<br>CLRX<br>CLRH<br>CLR *oprx8*,X<br>CLR ,X<br>CLR *oprx8*,SP | Clear M ← $00<br>A ← $00<br>X ← $00<br>H ← $00<br>M ← $00<br>M ← $00<br>M ← $00 | DIR<br>INH<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3F dd<br>4F<br>5F<br>8C<br>6F ff<br>7F<br>9E 6F ff | 5<br>1<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | 0 1 1 – | – 0 1 – |

**Table 7-2. Instruction Set Summary (Sheet 4 of 9)**

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| CMP #*opr8i*<br>CMP *opr8a*<br>CMP *opr16a*<br>CMP *oprx16*,X<br>CMP *oprx8*,X<br>CMP ,X<br>CMP *oprx16*,SP<br>CMP *oprx8*,SP | Compare Accumulator with Memory<br>A – M<br>(CCR Updated But Operands Not Changed) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A1 ii<br>B1 dd<br>C1 hh ll<br>D1 ee ff<br>E1 ff<br>F1<br>9E D1 ee ff<br>9E E1 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| COM *opr8a*<br>COMA<br>COMX<br>COM *oprx8*,X<br>COM ,X<br>COM *oprx8*,SP | Complement (One's Complement)<br>M ← (M̄)= $FF – (M)<br>A ← (Ā) = $FF – (A)<br>X ← (X̄) = $FF – (X)<br>M ← (M̄) = $FF – (M)<br>M ← (M̄) = $FF – (M)<br>M ← (M̄) = $FF – (M) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 33 dd<br>43<br>53<br>63 ff<br>73<br>9E 63 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | 0 1 1 – | – ↕ ↕ 1 |
| CPHX *opr16a*<br>CPHX #*opr16i*<br>CPHX *opr8a*<br>CPHX *oprx8*,SP | Compare Index Register (H:X) with Memory<br>(H:X) – (M:M + $0001)<br>(CCR Updated But Operands Not Changed) | EXT<br>IMM<br>DIR<br>SP1 | 3E hh ll<br>65 jj kk<br>75 dd<br>9E F3 ff | 6<br>3<br>5<br>6 | prrfpp<br>ppp<br>rrfpp<br>prrfpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| CPX #*opr8i*<br>CPX *opr8a*<br>CPX *opr16a*<br>CPX *oprx16*,X<br>CPX *oprx8*,X<br>CPX ,X<br>CPX *oprx16*,SP<br>CPX *oprx8*,SP | Compare X (Index Register Low) with Memory<br>X – M<br>(CCR Updated But Operands Not Changed) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A3 ii<br>B3 dd<br>C3 hh ll<br>D3 ee ff<br>E3 ff<br>F3<br>9E D3 ee ff<br>9E E3 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| DAA | Decimal Adjust Accumulator<br>After ADD or ADC of BCD Values | INH | 72 | 1 | p | U 1 1 – | – ↕ ↕ ↕ |
| DBNZ *opr8a*,*rel*<br>DBNZA *rel*<br>DBNZX *rel*<br>DBNZ *oprx8*,X,*rel*<br>DBNZ ,X,*rel*<br>DBNZ *oprx8*,SP,*rel* | Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0)<br>DBNZX Affects X Not H | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3B dd rr<br>4B rr<br>5B rr<br>6B ff rr<br>7B rr<br>9E 6B ff rr | 7<br>4<br>4<br>7<br>6<br>8 | rfwpppp<br>fppp<br>fppp<br>rfwpppp<br>rfwppp<br>prfwpppp | – 1 1 – | – – – – |
| DEC *opr8a*<br>DECA<br>DECX<br>DEC *oprx8*,X<br>DEC ,X<br>DEC *oprx8*,SP | Decrement<br>M ← (M) – $01<br>A ← (A) – $01<br>X ← (X) – $01<br>M ← (M) – $01<br>M ← (M) – $01<br>M ← (M) – $01 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3A dd<br>4A<br>5A<br>6A ff<br>7A<br>9E 6A ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ – |
| DIV | Divide<br>A ← (H:A)÷(X); H ← Remainder | INH | 52 | 6 | fffffp | – 1 1 – | – – ↕ ↕ |
| EOR #*opr8i*<br>EOR *opr8a*<br>EOR *opr16a*<br>EOR *oprx16*,X<br>EOR *oprx8*,X<br>EOR ,X<br>EOR *oprx16*,SP<br>EOR *oprx8*,SP | Exclusive OR Memory with Accumulator<br>A ← (A ⊕ M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A8 ii<br>B8 dd<br>C8 hh ll<br>D8 ee ff<br>E8 ff<br>F8<br>9E D8 ee ff<br>9E E8 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 1 1 – | – ↕ ↕ – |

**Table 7-2. Instruction Set Summary (Sheet 5 of 9)**

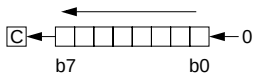
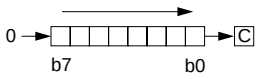
| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR V 1 1 H | Affect on CCR I N Z C |
|---|---|---|---|---|---|---|---|
| INC *oprx8*,X<br>INC ,X<br>INC *oprx8*,SP | Increment M ← (M) + $01<br>A ← (A) + $01<br>X ← (X) + $01<br>M ← (M) + $01<br>M ← (M) + $01<br>M ← (M) + $01 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3C dd<br>4C<br>5C<br>6C ff<br>7C<br>9E 6C ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ – |
| JMP *opr8a*<br>JMP *opr16a*<br>JMP *oprx16*,X<br>JMP *oprx8*,X<br>JMP ,X | Jump<br>PC ← Jump Address | DIR<br>EXT<br>IX2<br>IX1<br>IX | BC dd<br>CC hh ll<br>DC ee ff<br>EC ff<br>FC | 3<br>4<br>4<br>3<br>3 | ppp<br>pppp<br>pppp<br>ppp<br>ppp | – 1 1 – | – – – – |
| JSR *opr8a*<br>JSR *opr16a*<br>JSR *oprx16*,X<br>JSR *oprx8*,X<br>JSR ,X | Jump to Subroutine<br>PC ← (PC) + *n* (*n* = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – $0001<br>Push (PCH); SP ← (SP) – $0001<br>PC ← Unconditional Address | DIR<br>EXT<br>IX2<br>IX1<br>IX | BD dd<br>CD hh ll<br>DD ee ff<br>ED ff<br>FD | 5<br>6<br>6<br>5<br>5 | ssppp<br>pssppp<br>pssppp<br>ssppp<br>ssppp | – 1 1 – | – – – – |
| LDA *#oprx8*<br>LDA *oprx8*<br>LDA *oprx16*<br>LDA *oprx16*,X<br>LDA *oprx8*,X<br>LDA ,X<br>LDA *oprx16*,SP<br>LDA *oprx8*,SP | Load Accumulator from Memory<br>A ← (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A6 ii<br>B6 dd<br>C6 hh ll<br>D6 ee ff<br>E6 ff<br>F6<br>9E D6 ee ff<br>9E E6 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 1 1 – | – ↕ ↕ – |
| LDHX *#oprx16*<br>LDHX *oprx8*<br>LDHX *oprx16*<br>LDHX ,X<br>LDHX *oprx16*,X<br>LDHX *oprx8*,X<br>LDHX *oprx8*,SP | Load Index Register (H:X)<br>H:X ← (M:M + $0001) | IMM<br>DIR<br>EXT<br>IX<br>IX2<br>IX1<br>SP1 | 45 jj kk<br>55 dd<br>32 hh ll<br>9E AE<br>9E BE ee ff<br>9E CE ff<br>9E FE ff | 3<br>4<br>5<br>5<br>6<br>5<br>5 | ppp<br>rrpp<br>prrpp<br>prrfp<br>pprrpp<br>prrpp<br>prrpp | 0 1 1 – | – ↕ ↕ – |
| LDX *#oprx8*<br>LDX *oprx8*<br>LDX *oprx16*<br>LDX *oprx16*,X<br>LDX *oprx8*,X<br>LDX ,X<br>LDX *oprx16*,SP<br>LDX *oprx8*,SP | Load X (Index Register Low) from Memory<br>X ← (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AE ii<br>BE dd<br>CE hh ll<br>DE ee ff<br>EE ff<br>FE<br>9E DE ee ff<br>9E EE ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 1 1 – | – ↕ ↕ – |
| LSL *oprx8*,X<br>LSL ,X<br>LSL *oprx8*,SP | Logical Shift Left<br>  <br><br>(Same as ASL) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38 dd<br>48<br>58<br>68 ff<br>78<br>9E 68 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| LSR *oprx8*,X<br>LSR ,X<br>LSR *oprx8*,SP | Logical Shift Right<br>  <br> | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 34 dd<br>44<br>54<br>64 ff<br>74<br>9E 64 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – 0 ↕ ↕ |

**Table 7-2. Instruction Set Summary (Sheet 6 of 9)**

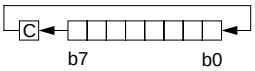
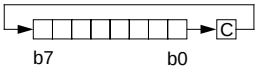
| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR V 1 1 H | Affect on CCR I N Z C |
|---|---|---|---|---|---|---|---|
| MOV *oprx8*,*oprx8*<br>MOV *oprx8*,X+<br>MOV #*oprx8*,*oprx8*<br>MOV ,X+,*oprx8* | Move<br>$(M)_{destination} \leftarrow (M)_{source}$<br>In IX+/DIR and DIR/IX+ Modes,<br>H:X ← (H:X) + $0001 | DIR/DIR<br>DIR/IX+<br>IMM/DIR<br>IX+/DIR | 4E dd dd<br>5E dd<br>6E ii dd<br>7E dd | 5<br>5<br>4<br>5 | rpwpp<br>rfwpp<br>pwpp<br>rfwpp | 0 1 1 – | – ↕ ↕ – |
| MUL | Unsigned multiply<br>X:A ← (X) × (A) | INH | 42 | 5 | fffffp | – 1 1 0 | – – – 0 |
| NEG *oprx8*<br>NEGA<br>NEGX<br>NEG *oprx8*,X<br>NEG ,X<br>NEG *oprx8*,SP | Negate (Two's Complement)<br>M ← – (M) = $00 – (M)<br>A ← – (A) = $00 – (A)<br>X ← – (X) = $00 – (X)<br>M ← – (M) = $00 – (M)<br>M ← – (M) = $00 – (M)<br>M ← – (M) = $00 – (M) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 30 dd<br>40<br>50<br>60 ff<br>70<br>9E 60 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| NOP | No Operation — Uses 1 Bus Cycle | INH | 9D | 1 | p | – 1 1 – | – – – – |
| NSA | Nibble Swap Accumulator<br>A ← (A[3:0]:A[7:4]) | INH | 62 | 1 | p | – 1 1 – | – – – – |
| ORA #*oprx8*<br>ORA *oprx8*<br>ORA *oprx16*<br>ORA *oprx16*,X<br>ORA *oprx8*,X<br>ORA ,X<br>ORA *oprx16*,SP<br>ORA *oprx8*,SP | Inclusive OR Accumulator and Memory<br>A ← (A) \| (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AA ii<br>BA dd<br>CA hh ll<br>DA ee ff<br>EA ff<br>FA<br>9E DA ee ff<br>9E EA ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 1 1 – | – ↕ ↕ – |
| PSHA | Push Accumulator onto Stack<br>Push (A); SP ← (SP) – $0001 | INH | 87 | 2 | sp | – 1 1 – | – – – – |
| PSHH | Push H (Index Register High) onto Stack<br>Push (H); SP ← (SP) – $0001 | INH | 8B | 2 | sp | – 1 1 – | – – – – |
| PSHX | Push X (Index Register Low) onto Stack<br>Push (X); SP ← (SP) – $0001 | INH | 89 | 2 | sp | – 1 1 – | – – – – |
| PULA | Pull Accumulator from Stack<br>SP ← (SP + $0001); Pull (A) | INH | 86 | 3 | ufp | – 1 1 – | – – – – |
| PULH | Pull H (Index Register High) from Stack<br>SP ← (SP + $0001); Pull (H) | INH | 8A | 3 | ufp | – 1 1 – | – – – – |
| PULX | Pull X (Index Register Low) from Stack<br>SP ← (SP + $0001); Pull (X) | INH | 88 | 3 | ufp | – 1 1 – | – – – – |
| ROL *oprx8*<br>ROLA<br>ROLX<br>ROL *oprx8*,X<br>ROL ,X<br>ROL *oprx8*,SP | Rotate Left through Carry<br>  | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 39 dd<br>49<br>59<br>69 ff<br>79<br>9E 69 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| ROR *oprx8*<br>RORA<br>RORX<br>ROR *oprx8*,X<br>ROR ,X<br>ROR *oprx8*,SP | Rotate Right through Carry<br>  | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 36 dd<br>46<br>56<br>66 ff<br>76<br>9E 66 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ 1 1 – | – ↕ ↕ ↕ |

**Table 7-2. Instruction Set Summary (Sheet 7 of 9)**

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| RSP | Reset Stack Pointer (Low Byte)<br>SPL ← $FF<br>(High Byte Not Affected) | INH | 9C | 1 | p | – 1 1 – | – – – – |
| RTI | Return from Interrupt<br>SP ← (SP) + $0001; Pull (CCR)<br>SP ← (SP) + $0001; Pull (A)<br>SP ← (SP) + $0001; Pull (X)<br>SP ← (SP) + $0001; Pull (PCH)<br>SP ← (SP) + $0001; Pull (PCL) | INH | 80 | 9 | uuuuufppp | ↕ 1 1 ↕ | ↕ ↕ ↕ ↕ |
| RTS | Return from Subroutine<br>SP ← SP + $0001; Pull (PCH)<br>SP ← SP + $0001; Pull (PCL) | INH | 81 | 5 | ufppp | – 1 1 – | – – – – |
| SBC *#opr8i*<br>SBC *opr8a*<br>SBC *opr16a*<br>SBC *oprx16*,X<br>SBC *oprx8*,X<br>SBC ,X<br>SBC *oprx16*,SP<br>SBC *oprx8*,SP | Subtract with Carry<br>A ← (A) – (M) – (C) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A2 ii<br>B2 dd<br>C2 hh ll<br>D2 ee ff<br>E2 ff<br>F2<br>9E D2 ee ff<br>9E E2 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| SEC | Set Carry Bit<br>(C ← 1) | INH | 99 | 1 | p | – 1 1 – | – – – 1 |
| SEI | Set Interrupt Mask Bit<br>(I ← 1) | INH | 9B | 1 | p | – 1 1 – | 1 – – – |
| STA *opr8a*<br>STA *opr16a*<br>STA *oprx16*,X<br>STA *oprx8*,X<br>STA ,X<br>STA *oprx16*,SP<br>STA *oprx8*,SP | Store Accumulator in Memory<br>M ← (A) | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | B7 dd<br>C7 hh ll<br>D7 ee ff<br>E7 ff<br>F7<br>9E D7 ee ff<br>9E E7 ff | 3<br>4<br>4<br>3<br>2<br>5<br>4 | wpp<br>pwpp<br>pwpp<br>wpp<br>wp<br>ppwpp<br>pwpp | 0 1 1 – | – ↕ ↕ – |
| STHX *opr8a*<br>STHX *opr16a*<br>STHX *oprx8*,SP | Store H:X (Index Reg.)<br>(M:M + $0001) ← (H:X) | DIR<br>EXT<br>SP1 | 35 dd<br>96 hh ll<br>9E FF ff | 4<br>5<br>5 | wwpp<br>pwwpp<br>pwwpp | 0 1 1 – | – ↕ ↕ – |
| STOP | Enable Interrupts: Stop Processing<br>Refer to MCU Documentation<br>I bit ← 0; Stop Processing | INH | 8E | 2 | fp... | – 1 1 – | 0 – – – |
| STX *opr8a*<br>STX *opr16a*<br>STX *oprx16*,X<br>STX *oprx8*,X<br>STX ,X<br>STX *oprx16*,SP<br>STX *oprx8*,SP | Store X (Low 8 Bits of Index Register) in Memory<br>M ← (X) | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | BF dd<br>CF hh ll<br>DF ee ff<br>EF ff<br>FF<br>9E DF ee ff<br>9E EF ff | 3<br>4<br>4<br>3<br>2<br>5<br>4 | wpp<br>pwpp<br>pwpp<br>wpp<br>wp<br>ppwpp<br>pwpp | 0 1 1 – | – ↕ ↕ – |

**Table 7-2. Instruction Set Summary (Sheet 8 of 9)**

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| SUB *#opr8i*<br>SUB *opr8a*<br>SUB *opr16a*<br>SUB *oprx16*,X<br>SUB *oprx8*,X<br>SUB ,X<br>SUB *oprx16*,SP<br>SUB *oprx8*,SP | Subtract<br>A ← (A) – (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A0 ii<br>B0 dd<br>C0 hh ll<br>D0 ee ff<br>E0 ff<br>F0<br>9E D0 ee ff<br>9E E0 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ 1 1 – | – ↕ ↕ ↕ |
| SWI | Software Interrupt<br>PC ← (PC) + $0001<br>Push (PCL); SP ← (SP) – $0001<br>Push (PCH); SP ← (SP) – $0001<br>Push (X); SP ← (SP) – $0001<br>Push (A); SP ← (SP) – $0001<br>Push (CCR); SP ← (SP) – $0001<br>I ← 1;<br>PCH ← Interrupt Vector High Byte<br>PCL ← Interrupt Vector Low Byte | INH | 83 | 11 | ssssvvfppp | – 1 1 – | 1 – – – |
| TAP | Transfer Accumulator to CCR<br>CCR ← (A) | INH | 84 | 1 | p | ↕ 1 1 ↕ | ↕ ↕ ↕ ↕ |
| TAX | Transfer Accumulator to X (Index Register Low)<br>X ← (A) | INH | 97 | 1 | p | – 1 1 – | – – – – |
| TPA | Transfer CCR to Accumulator<br>A ← (CCR) | INH | 85 | 1 | p | – 1 1 – | – – – – |
| TST *opr8a*<br>TSTA<br>TSTX<br>TST *oprx8*,X<br>TST ,X<br>TST *oprx8*,SP | Test for Negative or Zero (M) – $00<br>(A) – $00<br>(X) – $00<br>(M) – $00<br>(M) – $00<br>(M) – $00 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D dd<br>4D<br>5D<br>6D ff<br>7D<br>9E 6D ff | 4<br>1<br>1<br>4<br>3<br>5 | rfpp<br>p<br>p<br>rfpp<br>rfp<br>prfpp | 0 1 1 – | – ↕ ↕ – |
| TSX | Transfer SP to Index Reg.<br>H:X ← (SP) + $0001 | INH | 95 | 2 | fp | – 1 1 – | – – – – |
| TXA | Transfer X (Index Reg. Low) to Accumulator<br>A ← (X) | INH | 9F | 1 | p | – 1 1 – | – – – – |

## Table 7-2. Instruction Set Summary (Sheet 9 of 9)

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | V 1 1 H | I N Z C |
| TXS | Transfer Index Reg. to SP<br>SP ← (H:X) – $0001 | INH | 94 | 2 | fp | – 1 1 – | – – – – |
| WAIT | Enable Interrupts; Wait for Interrupt<br>I bit ← 0; Halt CPU | INH | 8F | 2+ | fp... | – 1 1 – | 0 – – – |

**Source Form:** Everything in the source forms columns, *except expressions in italic characters*, is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic and the characters (#, ( ) and +) are always a literal characters.

| | |
|---|---|
| *n* | Any label or expression that evaluates to a single integer in the range 0-7. |
| *opr8i* | Any label or expression that evaluates to an 8-bit immediate value. |
| *opr16i* | Any label or expression that evaluates to a 16-bit immediate value. |
| *opr8a* | Any label or expression that evaluates to an 8-bit direct-page address ($00xx). |
| *opr16a* | Any label or expression that evaluates to a 16-bit address. |
| *oprx8* | Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing. |
| *oprx16* | Any label or expression that evaluates to a 16-bit value, used for indexed addressing. |
| *rel* | Any label or expression that refers to an address that is within –128 to +127 locations from the start of the next instruction. |

**Operation Symbols:**

| | |
|---|---|
| A | Accumulator |
| CCR | Condition code register |
| H | Index register high byte |
| M | Memory location |
| *n* | Any bit |
| *opr* | Operand (one or two bytes) |
| PC | Program counter |
| PCH | Program counter high byte |
| PCL | Program counter low byte |
| *rel* | Relative program counter offset byte |
| SP | Stack pointer |
| SPL | Stack pointer low byte |
| X | Index register low byte |
| & | Logical AND |
| \| | Logical OR |
| ⊕ | Logical EXCLUSIVE OR |
| ( ) | Contents of |
| + | Add |
| – | Subtract, Negation (two's complement) |
| × | Multiply |
| ÷ | Divide |
| # | Immediate value |
| ← | Loaded with |
| : | Concatenated with |

**Addressing Modes:**

| | |
|---|---|
| DIR | Direct addressing mode |
| EXT | Extended addressing mode |
| IMM | Immediate addressing mode |
| INH | Inherent addressing mode |
| IX | Indexed, no offset addressing mode |
| IX1 | Indexed, 8-bit offset addressing mode |
| IX2 | Indexed, 16-bit offset addressing mode |
| IX+ | Indexed, no offset, post increment addressing mode |
| IX1+ | Indexed, 8-bit offset, post increment addressing mode |
| REL | Relative addressing mode |
| SP1 | Stack pointer, 8-bit offset addressing mode |
| SP2 | Stack pointer 16-bit offset addressing mode |

**Cycle-by-Cycle Codes:**

| | |
|---|---|
| f | Free cycle. This indicates a cycle where the CPU does not require use of the system buses. An f cycle is always one cycle of the system bus clock and is always a read cycle. |
| p | Program fetch; read from next consecutive location in program memory |
| r | Read 8-bit operand |
| s | Push (write) one byte onto stack |
| u | Pop (read) one byte from stack |
| v | Read vector from $FFxx (high byte first) |
| w | Write 8-bit operand |

**CCR Bits:**

| | |
|---|---|
| V | Overflow bit |
| H | Half-carry bit |
| I | Interrupt mask |
| N | Negative bit |
| Z | Zero bit |
| C | Carry/borrow bit |

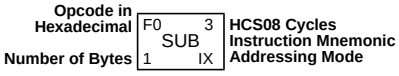
**CCR Effects:**

| | |
|---|---|
| ↕ | Set or cleared |
| – | Not affected |
| U | Undefined |

## Table 7-3. Opcode Map (Sheet 1 of 2)

| Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 5<br>BRSET0<br>3 DIR | 10 5<br>BSET0<br>2 DIR | 20 3<br>BRA<br>2 REL | 30 5<br>NEG<br>2 DIR | 40 1<br>NEGA<br>1 INH | 50 1<br>NEGX<br>1 INH | 60 5<br>NEG<br>2 IX1 | 70 4<br>NEG<br>1 IX | 80 9<br>RTI<br>1 INH | 90 3<br>BGE<br>2 REL | A0 2<br>SUB<br>2 IMM | B0 3<br>SUB<br>2 DIR | C0 4<br>SUB<br>3 EXT | D0 4<br>SUB<br>3 IX2 | E0 3<br>SUB<br>2 IX1 | F0 3<br>SUB<br>1 IX |
| 01 5<br>BRCLR0<br>3 DIR | 11 5<br>BCLR0<br>2 DIR | 21 3<br>BRN<br>2 REL | 31 5<br>CBEQ<br>3 DIR | 41 4<br>CBEQA<br>3 IMM | 51 4<br>CBEQX<br>3 IMM | 61 5<br>CBEQ<br>3 IX1+ | 71 5<br>CBEQ<br>2 IX+ | 81 6<br>RTS<br>1 INH | 91 3<br>BLT<br>2 REL | A1 2<br>CMP<br>2 IMM | B1 3<br>CMP<br>2 DIR | C1 4<br>CMP<br>3 EXT | D1 4<br>CMP<br>3 IX2 | E1 3<br>CMP<br>2 IX1 | F1 3<br>CMP<br>1 IX |
| 02 5<br>BRSET1<br>3 DIR | 12 5<br>BSET1<br>2 DIR | 22 3<br>BHI<br>2 REL | 32 5<br>LDHX<br>3 EXT | 42 5<br>MUL<br>1 INH | 52 6<br>DIV<br>1 INH | 62 1<br>NSA<br>1 INH | 72 1<br>DAA<br>1 INH | 82 5+<br>BGND<br>1 INH | 92 3<br>BGT<br>2 REL | A2 2<br>SBC<br>2 IMM | B2 3<br>SBC<br>2 DIR | C2 4<br>SBC<br>3 EXT | D2 4<br>SBC<br>3 IX2 | E2 3<br>SBC<br>2 IX1 | F2 3<br>SBC<br>1 IX |
| 03 5<br>BRCLR1<br>3 DIR | 13 5<br>BCLR1<br>2 DIR | 23 3<br>BLS<br>2 REL | 33 5<br>COM<br>2 DIR | 43 1<br>COMA<br>1 INH | 53 1<br>COMX<br>1 INH | 63 5<br>COM<br>2 IX1 | 73 4<br>COM<br>1 IX | 83 11<br>SWI<br>1 INH | 93 3<br>BLE<br>2 REL | A3 2<br>CPX<br>2 IMM | B3 3<br>CPX<br>2 DIR | C3 4<br>CPX<br>3 EXT | D3 4<br>CPX<br>3 IX2 | E3 3<br>CPX<br>2 IX1 | F3 3<br>CPX<br>1 IX |
| 04 5<br>BRSET2<br>3 DIR | 14 5<br>BSET2<br>2 DIR | 24 3<br>BCC<br>2 REL | 34 5<br>LSR<br>2 DIR | 44 1<br>LSRA<br>1 INH | 54 1<br>LSRX<br>1 INH | 64 5<br>LSR<br>2 IX1 | 74 4<br>LSR<br>1 IX | 84 1<br>TAP<br>1 INH | 94 2<br>TXS<br>1 INH | A4 2<br>AND<br>2 IMM | B4 3<br>AND<br>2 DIR | C4 4<br>AND<br>3 EXT | D4 4<br>AND<br>3 IX2 | E4 3<br>AND<br>2 IX1 | F4 3<br>AND<br>1 IX |
| 05 5<br>BRCLR2<br>3 DIR | 15 5<br>BCLR2<br>2 DIR | 25 3<br>BCS<br>2 REL | 35 4<br>STHX<br>2 DIR | 45 3<br>LDHX<br>3 IMM | 55 4<br>LDHX<br>2 DIR | 65 3<br>CPHX<br>3 IMM | 75 5<br>CPHX<br>2 DIR | 85 1<br>TPA<br>1 INH | 95 2<br>TSX<br>1 INH | A5 2<br>BIT<br>2 IMM | B5 3<br>BIT<br>2 DIR | C5 4<br>BIT<br>3 EXT | D5 4<br>BIT<br>3 IX2 | E5 3<br>BIT<br>2 IX1 | F5 3<br>BIT<br>1 IX |
| 06 5<br>BRSET3<br>3 DIR | 16 5<br>BSET3<br>2 DIR | 26 3<br>BNE<br>2 REL | 36 5<br>ROR<br>2 DIR | 46 1<br>RORA<br>1 INH | 56 1<br>RORX<br>1 INH | 66 5<br>ROR<br>2 IX1 | 76 4<br>ROR<br>1 IX | 86 3<br>PULA<br>1 INH | 96 5<br>STHX<br>3 EXT | A6 2<br>LDA<br>2 IMM | B6 3<br>LDA<br>2 DIR | C6 4<br>LDA<br>3 EXT | D6 4<br>LDA<br>3 IX2 | E6 3<br>LDA<br>2 IX1 | F6 3<br>LDA<br>1 IX |
| 07 5<br>BRCLR3<br>3 DIR | 17 5<br>BCLR3<br>2 DIR | 27 3<br>BEQ<br>2 REL | 37 5<br>ASR<br>2 DIR | 47 1<br>ASRA<br>1 INH | 57 1<br>ASRX<br>1 INH | 67 5<br>ASR<br>2 IX1 | 77 4<br>ASR<br>1 IX | 87 2<br>PSHA<br>1 INH | 97 1<br>TAX<br>1 INH | A7 2<br>AIS<br>2 IMM | B7 3<br>STA<br>2 DIR | C7 4<br>STA<br>3 EXT | D7 4<br>STA<br>3 IX2 | E7 3<br>STA<br>2 IX1 | F7 2<br>STA<br>1 IX |
| 08 5<br>BRSET4<br>3 DIR | 18 5<br>BSET4<br>2 DIR | 28 3<br>BHCC<br>2 REL | 38 5<br>LSL<br>2 DIR | 48 1<br>LSLA<br>1 INH | 58 1<br>LSLX<br>1 INH | 68 5<br>LSL<br>2 IX1 | 78 4<br>LSL<br>1 IX | 88 3<br>PULX<br>1 INH | 98 1<br>CLC<br>1 INH | A8 2<br>EOR<br>2 IMM | B8 3<br>EOR<br>2 DIR | C8 4<br>EOR<br>3 EXT | D8 4<br>EOR<br>3 IX2 | E8 3<br>EOR<br>2 IX1 | F8 3<br>EOR<br>1 IX |
| 09 5<br>BRCLR4<br>3 DIR | 19 5<br>BCLR4<br>2 DIR | 29 3<br>BHCS<br>2 REL | 39 5<br>ROL<br>2 DIR | 49 1<br>ROLA<br>1 INH | 59 1<br>ROLX<br>1 INH | 69 5<br>ROL<br>2 IX1 | 79 4<br>ROL<br>1 IX | 89 2<br>PSHX<br>1 INH | 99 1<br>SEC<br>1 INH | A9 2<br>ADC<br>2 IMM | B9 3<br>ADC<br>2 DIR | C9 4<br>ADC<br>3 EXT | D9 4<br>ADC<br>3 IX2 | E9 3<br>ADC<br>2 IX1 | F9 3<br>ADC<br>1 IX |
| 0A 5<br>BRSET5<br>3 DIR | 1A 5<br>BSET5<br>2 DIR | 2A 3<br>BPL<br>2 REL | 3A 5<br>DEC<br>2 DIR | 4A 1<br>DECA<br>1 INH | 5A 1<br>DECX<br>1 INH | 6A 5<br>DEC<br>2 IX1 | 7A 4<br>DEC<br>1 IX | 8A 3<br>PULH<br>1 INH | 9A 1<br>CLI<br>1 INH | AA 2<br>ORA<br>2 IMM | BA 3<br>ORA<br>2 DIR | CA 4<br>ORA<br>3 EXT | DA 4<br>ORA<br>3 IX2 | EA 3<br>ORA<br>2 IX1 | FA 3<br>ORA<br>1 IX |
| 0B 5<br>BRCLR5<br>3 DIR | 1B 5<br>BCLR5<br>2 DIR | 2B 3<br>BMI<br>2 REL | 3B 7<br>DBNZ<br>3 DIR | 4B 4<br>DBNZA<br>2 INH | 5B 4<br>DBNZX<br>2 INH | 6B 7<br>DBNZ<br>3 IX1 | 7B 6<br>DBNZ<br>2 IX | 8B 2<br>PSHH<br>1 INH | 9B 1<br>SEI<br>1 INH | AB 2<br>ADD<br>2 IMM | BB 3<br>ADD<br>2 DIR | CB 4<br>ADD<br>3 EXT | DB 4<br>ADD<br>3 IX2 | EB 3<br>ADD<br>2 IX1 | FB 3<br>ADD<br>1 IX |
| 0C 5<br>BRSET6<br>3 DIR | 1C 5<br>BSET6<br>2 DIR | 2C 3<br>BMC<br>2 REL | 3C 5<br>INC<br>2 DIR | 4C 1<br>INCA<br>1 INH | 5C 1<br>INCX<br>1 INH | 6C 5<br>INC<br>2 IX1 | 7C 4<br>INC<br>1 IX | 8C 1<br>CLRH<br>1 INH | 9C 1<br>RSP<br>1 INH | | BC 3<br>JMP<br>2 DIR | CC 4<br>JMP<br>3 EXT | DC 4<br>JMP<br>3 IX2 | EC 3<br>JMP<br>2 IX1 | FC 3<br>JMP<br>1 IX |
| 0D 5<br>BRCLR6<br>3 DIR | 1D 5<br>BCLR6<br>2 DIR | 2D 3<br>BMS<br>2 REL | 3D 4<br>TST<br>2 DIR | 4D 1<br>TSTA<br>1 INH | 5D 1<br>TSTX<br>1 INH | 6D 4<br>TST<br>2 IX1 | 7D 3<br>TST<br>1 IX | | 9D 1<br>NOP<br>1 INH | AD 5<br>BSR<br>2 REL | BD 5<br>JSR<br>2 DIR | CD 6<br>JSR<br>3 EXT | DD 6<br>JSR<br>3 IX2 | ED 5<br>JSR<br>2 IX1 | FD 5<br>JSR<br>1 IX |
| 0E 5<br>BRSET7<br>3 DIR | 1E 5<br>BSET7<br>2 DIR | 2E 3<br>BIL<br>2 REL | 3E 6<br>CPHX<br>3 EXT | 4E 5<br>MOV<br>3 DD | 5E 5<br>MOV<br>2 DIX+ | 6E 4<br>MOV<br>3 IMD | 7E 5<br>MOV<br>2 IX+D | 8E 2+<br>STOP<br>1 INH | 9E<br>Page 2 | AE 2<br>LDX<br>2 IMM | BE 3<br>LDX<br>2 DIR | CE 4<br>LDX<br>3 EXT | DE 4<br>LDX<br>3 IX2 | EE 3<br>LDX<br>2 IX1 | FE 3<br>LDX<br>1 IX |
| 0F 5<br>BRCLR7<br>3 DIR | 1F 5<br>BCLR7<br>2 DIR | 2F 3<br>BIH<br>2 REL | 3F 5<br>CLR<br>2 DIR | 4F 1<br>CLRA<br>1 INH | 5F 1<br>CLRX<br>1 INH | 6F 5<br>CLR<br>2 IX1 | 7F 4<br>CLR<br>1 IX | 8F 2+<br>WAIT<br>1 INH | 9F 1<br>TXA<br>1 INH | AF 2<br>AIX<br>2 IMM | BF 3<br>STX<br>2 DIR | CF 4<br>STX<br>3 EXT | DF 4<br>STX<br>3 IX2 | EF 3<br>STX<br>2 IX1 | FF 2<br>STX<br>1 IX |

| | | | | | |
|---|---|---|---|---|---|
| **INH** | **Inherent** | **REL** | **Relative** | **SP1** | **Stack Pointer, 8-Bit Offset** |
| **IMM** | **Immediate** | **IX** | **Indexed, No Offset** | **SP2** | **Stack Pointer, 16-Bit Offset** |
| **DIR** | **Direct** | **IX1** | **Indexed, 8-Bit Offset** | **IX+** | **Indexed, No Offset with Post Increment** |
| **EXT** | **Extended** | **IX2** | **Indexed, 16-Bit Offset** | **IX1+** | **Indexed, 1-Byte Offset with Post Increment** |
| **DD** | **DIR to DIR** | **IMD** | **IMM to DIR** | | |
| **IX+D** | **IX+ to DIR** | **DIX+** | **DIR to IX+** | | |



## Table 7-3. Opcode Map (Sheet 2 of 2)

| Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 9E60 6 NEG 3 SP1 | | | | | | | 9ED0 5 SUB 4 SP2 | 9EE0 4 SUB 3 SP1 | |
| | | | | | | 9E61 6 CBEQ 4 SP1 | | | | | | | 9ED1 5 CMP 4 SP2 | 9EE1 4 CMP 3 SP1 | |
| | | | | | | | | | | | | | 9ED2 5 SBC 4 SP2 | 9EE2 4 SBC 3 SP1 | |
| | | | | | | 9E63 6 COM 3 SP1 | | | | | | | 9ED3 5 CPX 4 SP2 | 9EE3 4 CPX 3 SP1 | 9EF3 6 CPHX 3 SP1 |
| | | | | | | 9E64 6 LSR 3 SP1 | | | | | | | 9ED4 5 AND 4 SP2 | 9EE4 4 AND 3 SP1 | |
| | | | | | | | | | | | | | 9ED5 5 BIT 4 SP2 | 9EE5 4 BIT 3 SP1 | |
| | | | | | | 9E66 6 ROR 3 SP1 | | | | | | | 9ED6 5 LDA 4 SP2 | 9EE6 4 LDA 3 SP1 | |
| | | | | | | 9E67 6 ASR 3 SP1 | | | | | | | 9ED7 5 STA 4 SP2 | 9EE7 4 STA 3 SP1 | |
| | | | | | | 9E68 6 LSL 3 SP1 | | | | | | | 9ED8 5 EOR 4 SP2 | 9EE8 4 EOR 3 SP1 | |
| | | | | | | 9E69 6 ROL 3 SP1 | | | | | | | 9ED9 5 ADC 4 SP2 | 9EE9 4 ADC 3 SP1 | |
| | | | | | | 9E6A 6 DEC 3 SP1 | | | | | | | 9EDA 5 ORA 4 SP2 | 9EEA 4 ORA 3 SP1 | |
| | | | | | | 9E6B 8 DBNZ 4 SP1 | | | | | | | 9EDB 5 ADD 4 SP2 | 9EEB 4 ADD 3 SP1 | |
| | | | | | | 9E6C 6 INC 3 SP1 | | | | | | | | | |
| | | | | | | 9E6D 5 TST 3 SP1 | | | | | | | | | |
| | | | | | | | | | | 9EAE 5 LDHX 2 IX | 9EBE 6 LDHX 4 IX2 | 9ECE 5 LDHX 3 IX1 | 9EDE 5 LDX 4 SP2 | 9EEE 4 LDX 3 SP1 | 9EFE 5 LDHX 3 SP1 |
| | | | | | | 9E6F 6 CLR 3 SP1 | | | | | | | 9EDF 5 STX 4 SP2 | 9EEF 4 STX 3 SP1 | 9EFF 5 STHX 3 SP1 |

| | | | | | |
|---|---|---|---|---|---|
| **INH** | **Inherent** | **REL** | **Relative** | **SP1** | **Stack Pointer, 8-Bit Offset** |
| **IMM** | **Immediate** | **IX** | **Indexed, No Offset** | **SP2** | **Stack Pointer, 16-Bit Offset** |
| **DIR** | **Direct** | **IX1** | **Indexed, 8-Bit Offset** | **IX+** | **Indexed, No Offset with** |
| **EXT** | **Extended** | **IX2** | **Indexed, 16-Bit Offset** | | **Post Increment** |
| **DD** | **DIR to DIR** | **IMD** | **IMM to DIR** | **IX1+** | **Indexed, 1-Byte Offset with** |
| **IX+D** | **IX+ to DIR** | **DIX+** | **DIR to IX+** | | **Post Increment** |

**Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)**

**Prebyte (9E) and Opcode in Hexadecimal**
**Number of Bytes**

9E60 6
NEG
3 SP1

**HCS08 Cycles**
**Instruction Mnemonic**
**Addressing Mode**

# Chapter 8
# Multi-Purpose Clock Generator (S08MCGV1)

## 8.1 Introduction

The multi-purpose clock generator (MCG) module provides several clock source choices for the MCU. The module contains a frequency-locked loop (FLL) and a phase-locked loop (PLL) that are controllable by either an internal or an external reference clock. The module can select either of the FLL or PLL clocks, or either of the internal or external reference clocks as a source for the MCU system clock. Whichever clock source is chosen, it is passed through a reduced bus divider which allows a lower output clock frequency to be derived. The MCG also controls an external oscillator (XOSC) for the use of a crystal or resonator as the external reference clock.

All devices in the MC9S08DZ60 Series feature the MCG module.

**NOTE**

Refer to Section 1.3, “System Clock Distribution,” for detailed view of the distribution clock sources throughout the chip.

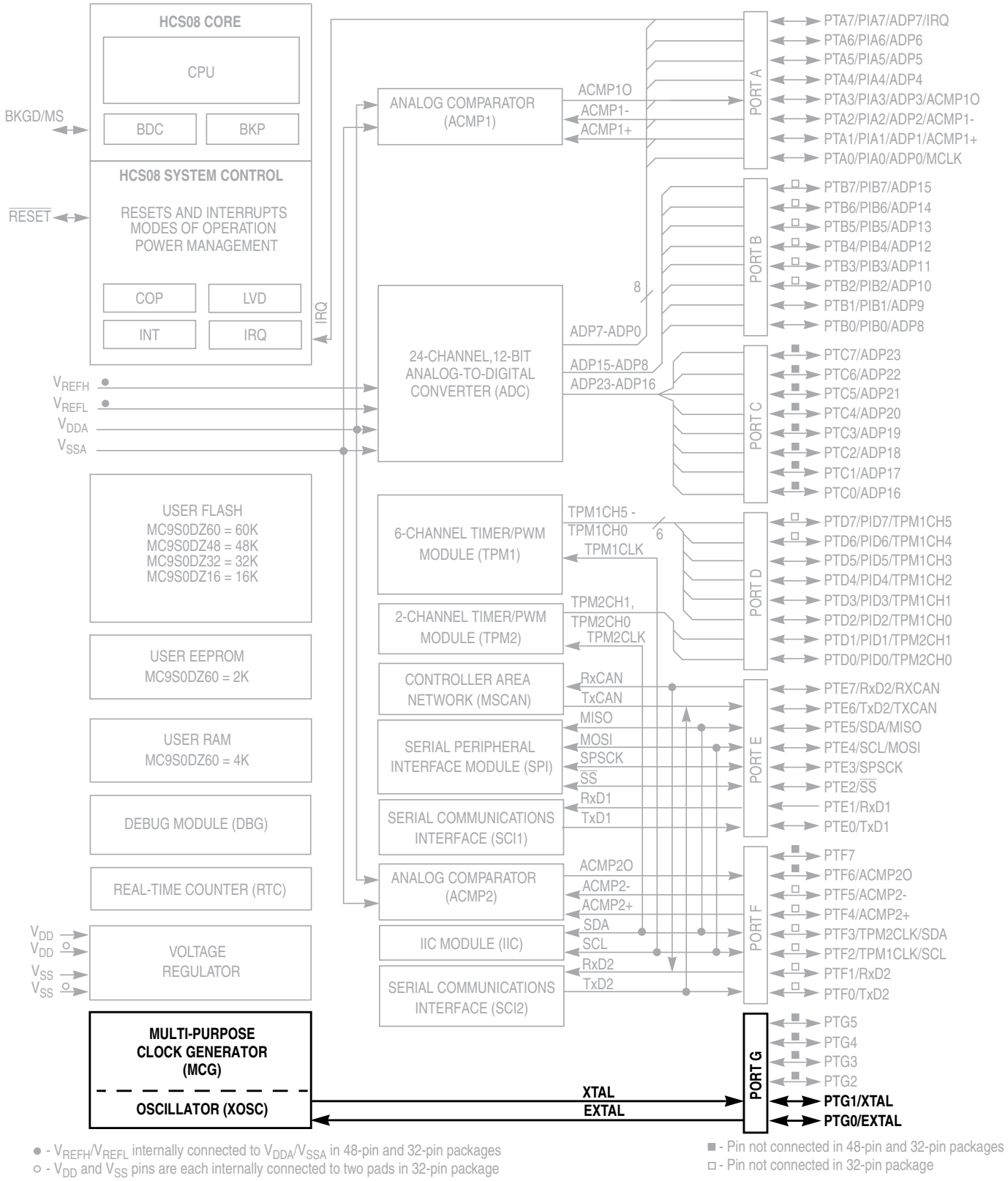



**Figure 8-1. MC9S08DZ60 Block Diagram**

### 8.1.1 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL)
  - 0.2% resolution using internal 32-kHz reference
  - 2% deviation over voltage and temperature using internal 32-kHz reference
  - Internal or external reference can be used to control the FLL
- Phase-locked loop (PLL)
  - Voltage-controlled oscillator (VCO)
  - Modulo VCO frequency divider
  - Phase/Frequency detector
  - Integrated loop filter
  - Lock detector with interrupt capability
- Internal reference clock
  - Nine trim bits for accuracy
  - Can be selected as the clock source for the MCU
- External reference clock
  - Control for external oscillator
  - Clock monitor with reset capability
  - Can be selected as the clock source for the MCU
- Reference divider is provided
- Clock source selected can be divided down by 1, 2, 4, or 8
- BDC clock (MCGLCLK) is provided as a constant divide by 2 of the DCO output whether in an FLL or PLL mode.

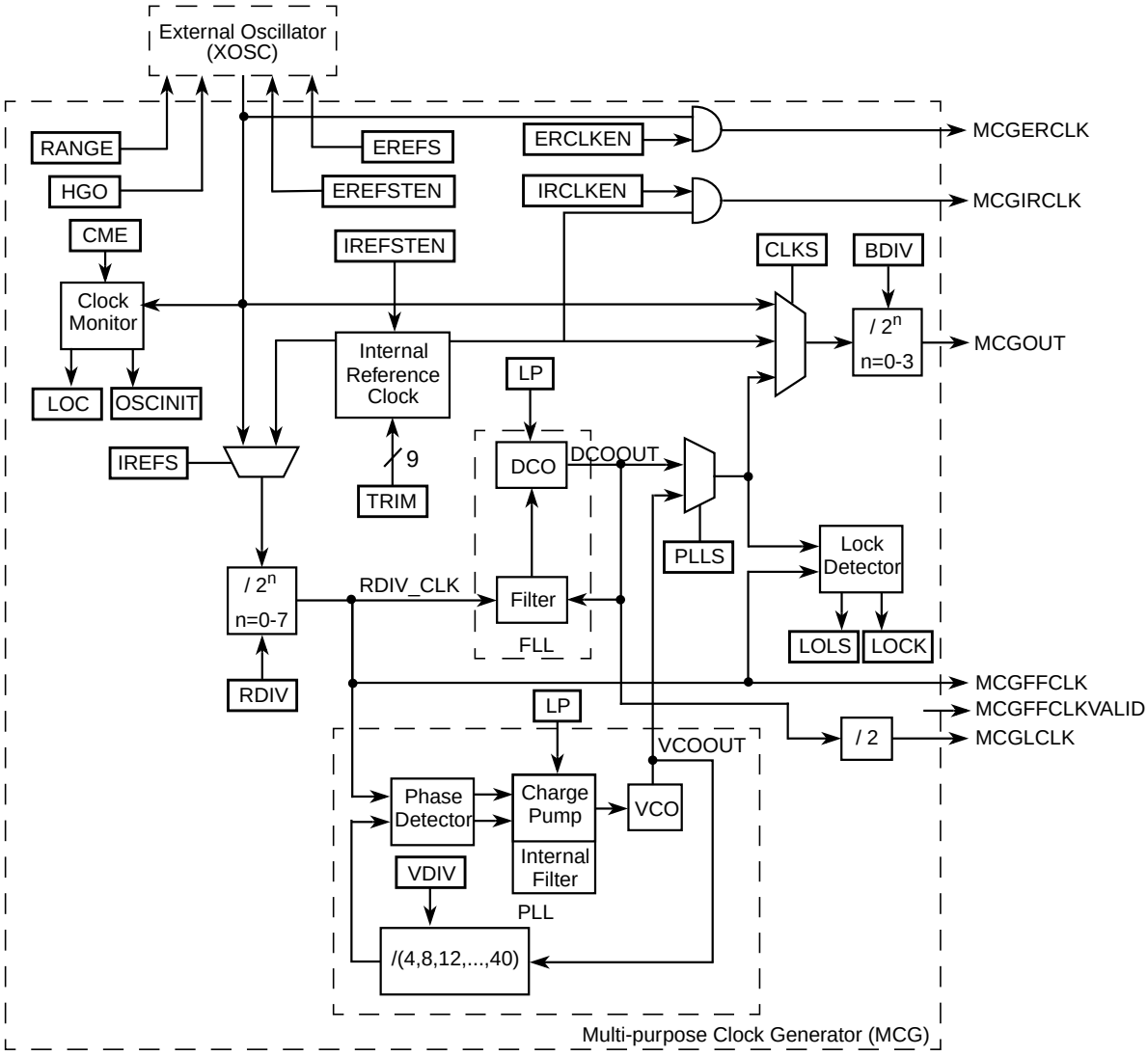



**Figure 8-2. Multi-Purpose Clock Generator (MCG) Block Diagram**

### 8.1.2 Modes of Operation

There are nine modes of operation for the MCG:

- FLL Engaged Internal (FEI)
- FLL Engaged External (FEE)
- FLL Bypassed Internal (FBI)
- FLL Bypassed External (FBE)
- PLL Engaged External (PEE)
- PLL Bypassed External (PBE)
- Bypassed Low Power Internal (BLPI)
- Bypassed Low Power External (BLPE)
- Stop

For details see Section 8.4.1, “Operational Modes.

## 8.2 External Signal Description

There are no MCG signals that connect off chip.

# 8.3 Register Definition

## 8.3.1 MCG Control Register 1 (MCGC1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CLKS | | RDIV | | | IREFS | IRCLKEN | IREFSTEN |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 8-3. MCG Control Register 1 (MCGC1)**

**Table 8-1. MCG Control Register 1 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>CLKS | **Clock Source Select** — Selects the system clock source.<br>00 Encoding 0 — Output of FLL or PLL is selected.<br>01 Encoding 1 — Internal reference clock is selected.<br>10 Encoding 2 — External reference clock is selected.<br>11 Encoding 3 — Reserved, defaults to 00. |
| 5:3<br>RDIV | **Reference Divider** — Selects the amount to divide down the reference clock selected by the IREFS bit. If the FLL is selected, the resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. If the PLL is selected, the resulting frequency must be in the range 1 MHz to 2 MHz.<br>000 Encoding 0 — Divides reference clock by 1 (reset default)<br>001 Encoding 1 — Divides reference clock by 2<br>010 Encoding 2 — Divides reference clock by 4<br>011 Encoding 3 — Divides reference clock by 8<br>100 Encoding 4 — Divides reference clock by 16<br>101 Encoding 5 — Divides reference clock by 32<br>110 Encoding 6 — Divides reference clock by 64<br>111 Encoding 7 — Divides reference clock by 128 |
| 2<br>IREFS | **Internal Reference Select** — Selects the reference clock source.<br>1 Internal reference clock selected<br>0 External reference clock selected |
| 1<br>IRCLKEN | **Internal Reference Clock Enable** — Enables the internal reference clock for use as MCGIRCLK.<br>1 MCGIRCLK active<br>0 MCGIRCLK inactive |
| 0<br>IREFSTEN | **Internal Reference Stop Enable** — Controls whether or not the internal reference clock remains enabled when the MCG enters stop mode.<br>1 Internal reference clock stays enabled in stop if IRCLKEN is set or if MCG is in FEI, FBI, or BLPI mode before entering stop<br>0 Internal reference clock is disabled in stop |

## 8.3.2 MCG Control Register 2 (MCGC2)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | BDIV | | RANGE | HGO | LP | EREFS | ERCLKEN | EREFSTEN |
| Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-4. MCG Control Register 2 (MCGC2)**

**Table 8-2. MCG Control Register 2 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>BDIV | **Bus Frequency Divider** — Selects the amount to divide down the clock source selected by the CLKS bits in the MCGC1 register. This controls the bus frequency.<br>00 Encoding 0 — Divides selected clock by 1<br>01 Encoding 1 — Divides selected clock by 2 (reset default)<br>10 Encoding 2 — Divides selected clock by 4<br>11 Encoding 3 — Divides selected clock by 8 |
| 5<br>RANGE | **Frequency Range Select** — Selects the frequency range for the external oscillator or external clock source.<br>1 High frequency range selected for the external oscillator of 1 MHz to 16 MHz (1 MHz to 40 MHz for external clock source)<br>0 Low frequency range selected for the external oscillator of 32 kHz to 100 kHz (32 kHz to 1 MHz for external clock source) |
| 4<br>HGO | **High Gain Oscillator Select** — Controls the external oscillator mode of operation.<br>1 Configure external oscillator for high gain operation<br>0 Configure external oscillator for low power operation |
| 3<br>LP | **Low Power Select** — Controls whether the FLL (or PLL) is disabled in bypassed modes.<br>1 FLL (or PLL) is disabled in bypass modes (lower power)**.**<br>0 FLL (or PLL) is not disabled in bypass modes. |
| 2<br>EREFS | **External Reference Select** — Selects the source for the external reference clock.<br>1 Oscillator requested<br>0 External Clock Source requested |
| 1<br>ERCLKEN | **External Reference Enable** — Enables the external reference clock for use as MCGERCLK.<br>1 MCGERCLK active<br>0 MCGERCLK inactive |
| 0<br>EREFSTEN | **External Reference Stop Enable** — Controls whether or not the external reference clock remains enabled when the MCG enters stop mode.<br>1 External reference clock stays enabled in stop if ERCLKEN is set or if MCG is in FEE, FBE, PEE, PBE, or BLPE mode before entering stop<br>0 External reference clock is disabled in stop |

## 8.3.3 MCG Trim Register (MCGTRM)

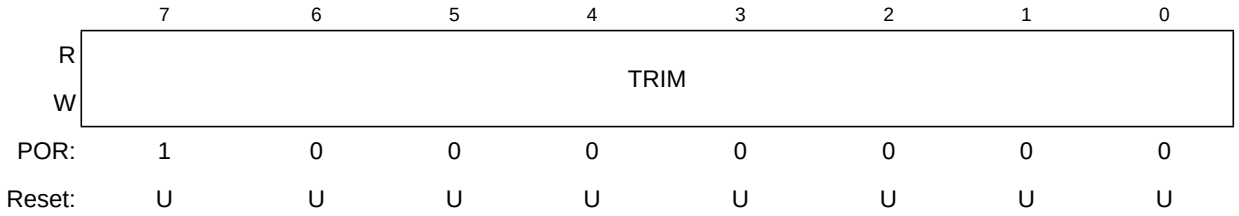



**Figure 8-5. MCG Trim Register (MCGTRM)**

**Table 8-3. MCG Trim Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>TRIM | MCG **Trim Setting** — Controls the internal reference clock frequency by controlling the internal reference clock period. The TRIM bits are binary weighted (i.e., bit 1 will adjust twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period.<br><br>An additional fine trim bit is available in MCGSC as the FTRIM bit.<br><br>If a TRIM[7:0] value stored in nonvolatile memory is to be used, it's the user's responsibility to copy that value from the nonvolatile memory location to this register. |

## 8.3.4 MCG Status and Control Register (MCGSC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LOLS | LOCK | PLLST | IREFST | CLKST | | OSCINIT | FTRIM |
| W | | | | | | | | |
| POR: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Reset: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | U |

**Figure 8-6. MCG Status and Control Register (MCGSC)**

**Table 8-4. MCG Status and Control Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LOLS | **Loss of Lock Status** — This bit is a sticky indication of lock status for the FLL or PLL. LOLS is set when lock detection is enabled and after acquiring lock, the FLL or PLL output frequency has fallen outside the lock exit frequency tolerance, $D_{unl}$. LOLIE determines whether an interrupt request is made when set. LOLS is cleared by reset or by writing a logic 1 to LOLS when LOLS is set. Writing a logic 0 to LOLS has no effect.<br>0 FLL or PLL has not lost lock since LOLS was last cleared.<br>1 FLL or PLL has lost lock since LOLS was last cleared. |
| 6<br>LOCK | **Lock Status** — Indicates whether the FLL or PLL has acquired lock. Lock detection is disabled when both the FLL and PLL are disabled. If the lock status bit is set then changing the value of any of the following bits IREFS, PLLS, RDIV[2:0], TRIM[7:0] (if in FEI or FBI modes), or VDIV[3:0] (if in PBE or PEE modes), will cause the lock status bit to clear and stay cleared until the FLL or PLL has reacquired lock. Stop mode entry will also cause the lock status bit to clear and stay cleared until the FLL or PLL has reacquired lock. Entry into BLPI or BLPE mode will also cause the lock status bit to clear and stay cleared until the MCG has exited these modes and the FLL or PLL has reacquired lock.<br>0 FLL or PLL is currently unlocked.<br>1 FLL or PLL is currently locked. |
| 5<br>PLLST | **PLL Select Status** — The PLLST bit indicates the current source for the PLLS clock. The PLLST bit does not update immediately after a write to the PLLS bit due to internal synchronization between clock domains.<br>0 Source of PLLS clock is FLL clock.<br>1 Source of PLLS clock is PLL clock. |
| 4<br>IREFST | **Internal Reference Status** — The IREFST bit indicates the current source for the reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains.<br>0 Source of reference clock is external reference clock (oscillator or external clock source as determined by the EREFS bit in the MCGC2 register).<br>1 Source of reference clock is internal reference clock. |
| 3:2<br>CLKST | **Clock Mode Status** — The CLKST bits indicate the current clock mode. The CLKST bits do not update immediately after a write to the CLKS bits due to internal synchronization between clock domains.<br>00 Encoding 0 — Output of FLL is selected.<br>01 Encoding 1 — Internal reference clock is selected.<br>10 Encoding 2 — External reference clock is selected.<br>11 Encoding 3 — Output of PLL is selected. |

**Table 8-4. MCG Status and Control Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>OSCINIT | **OSC Initialization** — If the external reference clock is selected by ERCLKEN or by the MCG being in FEE, FBE, PEE, PBE, or BLPE mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either EREFS is cleared or when the MCG is in either FEI, FBI, or BLPI mode and ERCLKEN is cleared. |
| 0<br>FTRIM | MCG **Fine Trim** — Controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible.<br><br>If an FTRIM value stored in nonvolatile memory is to be used, it's the user's responsibility to copy that value from the nonvolatile memory location to this register's FTRIM bit. |

## 8.3.5 MCG Control Register 3 (MCGC3)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LOLIE | PLLS | CME | 0 | VDIV | | | |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 8-7. MCG PLL Register (MCGPLL)**

**Table 8-5. MCG PLL Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LOLIE | **Loss of Lock Interrupt Enable** — Determines if an interrupt request is made following a loss of lock indication. The LOLIE bit only has an effect when LOLS is set.<br>0 No request on loss of lock.<br>1 Generate an interrupt request on loss of lock. |
| 6<br>PLLS | **PLL Select** — Controls whether the PLL or FLL is selected. If the PLLS bit is clear, the PLL is disabled in all modes. If the PLLS is set, the FLL is disabled in all modes.<br>1 PLL is selected<br>0 FLL is selected |

**Table 8-5. MCG PLL Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>CME | **Clock Monitor Enable** — Determines if a reset request is made following a loss of external clock indication. The CME bit should only be set to a logic 1 when either the MCG is in an operational mode that uses the external clock (FEE, FBE, PEE, PBE, or BLPE) or the external reference is enabled (ERCLKEN=1 in the MCGC2 register). Whenever the CME bit is set to a logic 1, the value of the RANGE bit in the MCGC2 register should not be changed.<br>0 Clock monitor is disabled.<br>1 Generate a reset request on loss of external clock. |
| 3:0<br>VDIV | **VCO Divider** — Selects the amount to divide down the VCO output of PLL. The VDIV bits establish the multiplication factor (M) applied to the reference clock frequency.<br>0000 Encoding 0 — Reserved.<br>0001 Encoding 1 — Multiply by 4.<br>0010 Encoding 2 — Multiply by 8.<br>0011 Encoding 3 — Multiply by 12.<br>0100 Encoding 4 — Multiply by 16.<br>0101 Encoding 5 — Multiply by 20.<br>0110 Encoding 6 — Multiply by 24.<br>0111 Encoding 7 — Multiply by 28.<br>1000 Encoding 8 — Multiply by 32.<br>1001 Encoding 9 — Multiply by 36.<br>1010 Encoding 10 — Multiply by 40.<br>1011 Encoding 11 — Reserved (default to M=40).<br>11xx Encoding 12-15 — Reserved (default to M=40). |

# 8.4 Functional Description

## 8.4.1 Operational Modes

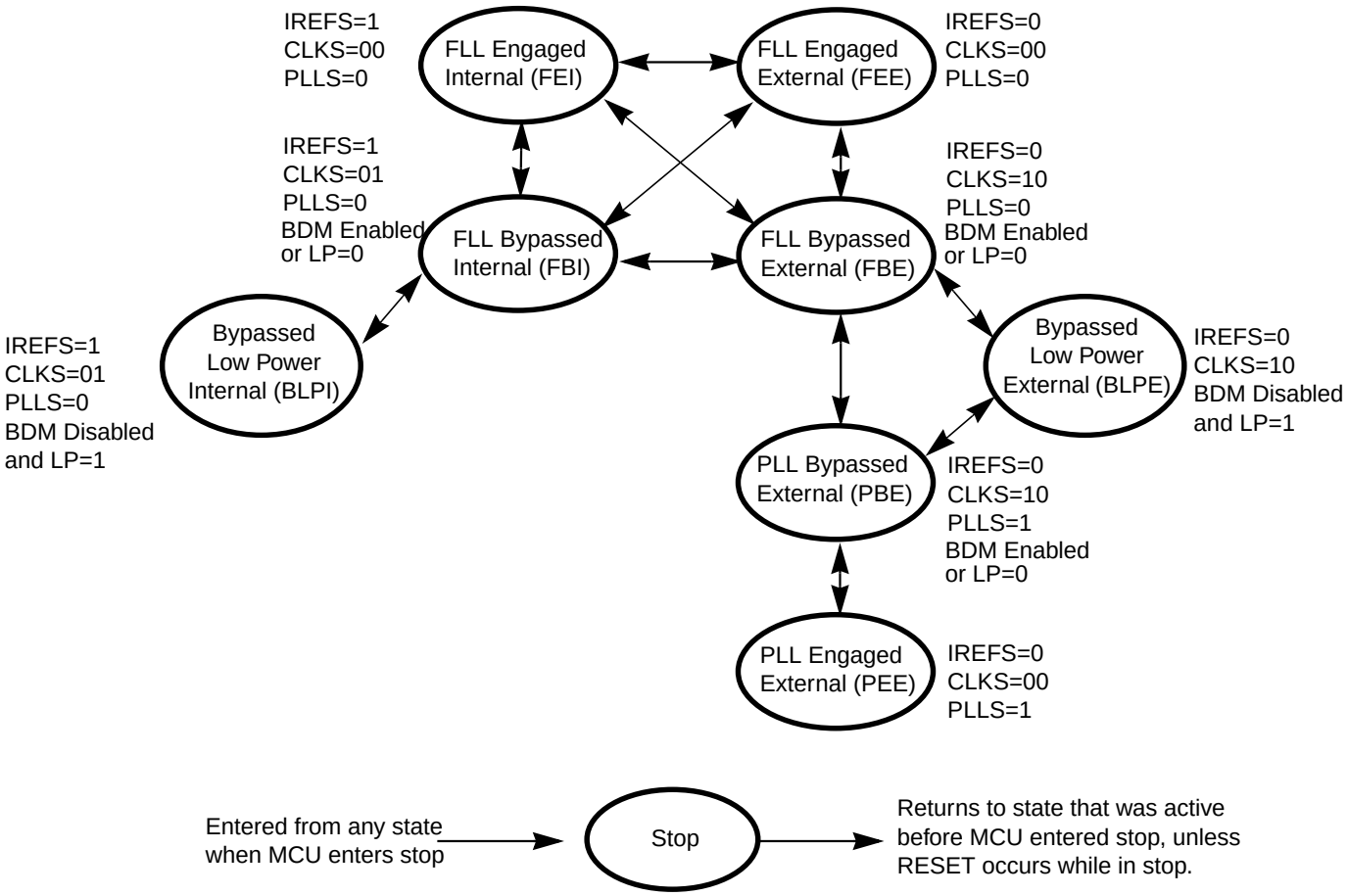



**Figure 8-8. Clock Switching Modes**

The nine states of the MCG are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

#### 8.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 1
- PLLS bit is written to 0
- RDIV bits are written to 000. Since the internal reference clock frequency should already be in the range of 31.25 kHz to 39.0625 kHz after it is trimmed, no further frequency divide is necessary.

In FLL engaged internal mode, the MCGOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

#### 8.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- PLLS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz

In FLL engaged external mode, the MCGOUT clock is derived from the FLL clock which is controlled by the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.The FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

#### 8.4.1.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal (FBI) mode, the MCGOUT clock is derived from the internal reference clock and the FLL is operational but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUT clock is driven from the internal reference clock.

The FLL bypassed internal mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- PLLS bit is written to 0
- RDIV bits are written to 000. Since the internal reference clock frequency should already be in the range of 31.25 kHz to 39.0625 kHz after it is trimmed, no further frequency divide is necessary.

- LP bit is written to 0

In FLL bypassed internal mode, the MCGOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

#### 8.4.1.4 FLL Bypassed External (FBE)

In FLL bypassed external (FBE) mode, the MCGOUT clock is derived from the external reference clock and the FLL is operational but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUT clock is driven from the external reference clock.

The FLL bypassed external mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz
- LP bit is written to 0

In FLL bypassed external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.The FLL clock is controlled by the external reference clock, and the FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

**NOTE**

It is possible to briefly operate in FBE mode with an FLL reference clock frequency that is greater than the specified maximum frequency. This can be necessary in applications that operate in PEE mode using an external crystal with a frequency above 5 MHz. Please see 8.5.2.4, “Example # 4: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 8 MHz for a detailed example.

#### 8.4.1.5 PLL Engaged External (PEE)

The PLL engaged external (PEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- PLLS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 1 MHz to 2 MHz

In PLL engaged external mode, the MCGOUT clock is derived from the PLL clock which is controlled by the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source The PLL clock frequency locks to a

multiplication factor, as selected by the VDIV bits, times the reference frequency, as selected by the RDIV bits. If BDM is enabled then the MCGLCLK is derived from the DCO (open-loop mode) divided by two. If BDM is not enabled then the FLL is disabled in a low power state.

#### 8.4.1.6 PLL Bypassed External (PBE)

In PLL bypassed external (PBE) mode, the MCGOUT clock is derived from the external reference clock and the PLL is operational but its output clock is not used. This mode is useful to allow the PLL to acquire its target frequency while the MCGOUT clock is driven from the external reference clock.

The PLL bypassed external mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 1 MHz to 2 MHz
- LP bit is written to 0

In PLL bypassed external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source. The PLL clock frequency locks to a multiplication factor, as selected by the VDIV bits, times the reference frequency, as selected by the RDIV bits. If BDM is enabled then the MCGLCLK is derived from the DCO (open-loop mode) divided by two. If BDM is not enabled then the FLL is disabled in a low power state.

#### 8.4.1.7 Bypassed Low Power Internal (BLPI)

The bypassed low power internal (BLPI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- PLLS bit is written to 0
- LP bit is written to 1
- BDM mode is not active

In bypassed low power internal mode, the MCGOUT clock is derived from the internal reference clock.

The PLL and the FLL are disabled at all times in BLPI mode and the MCGLCLK will not be available for BDC communications If the BDM becomes active the mode will switch to FLL bypassed internal (FBI) mode.

#### 8.4.1.8 Bypassed Low Power External (BLPE)

The bypassed low power external (BLPE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 0 or 1

- LP bit is written to 1
- BDM mode is not active

In bypassed low power external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.

The PLL and the FLL are disabled at all times in BLPE mode and the MCGLCLK will not be available for BDC communications. If the BDM becomes active the mode will switch to one of the bypassed external modes as determined by the state of the PLLS bit.

#### 8.4.1.9 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, the FLL and PLL are disabled and all MCG clock signals are static except in the following cases:

MCGIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN = 1
- IREFSTEN = 1

MCGERCLK will be active in stop mode when all the following conditions occur:

- ERCLKEN = 1
- EREFSTEN = 1

### 8.4.2 Mode Switching

When switching between engaged internal and engaged external modes the IREFS bit can be changed at anytime, but the RDIV bits must be changed simultaneously so that the reference frequency stays in the range required by the state of the PLLS bit (31.25 kHz to 39.0625 kHz if the FLL is selected, or 1 MHz to 2 MHz if the PLL is selected). After a change in the IREFS value the FLL or PLL will begin locking again after the switch is completed. The completion of the switch is shown by the IREFST bit .

For the special case of entering stop mode immediately after switching to FBE mode, if the external clock and the internal clock are disabled in stop mode, (EREFSTEN = 0 and IREFSTEN = 0), it is necessary to allow 100us after the IREFST bit is cleared to allow the internal reference to shutdown. For most cases the delay due to instruction execution times will be sufficient.

The CLKS bits can also be changed at anytime, but in order for the MCGLCLK to be configured correctly the RDIV bits must be changed simultaneously so that the reference frequency stays in the range required by the state of the PLLS bit (31.25 kHz to 39.0625 kHz if the FLL is selected, or 1 MHz to 2MHz if the PLL is selected). The actual switch to the newly selected clock will be shown by the CLKST bits. If the newly selected clock is not available, the previous clock will remain selected.

For details see Figure 8-8.

### 8.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency will occur immediately.

### 8.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL or PLL to be disabled and thus conserve power when these systems are not being used. However, in some applications it may be desirable to enable the FLL or PLL and allow it to lock for maximum accuracy before switching to an engaged mode. Do this by writing the LP bit to 0.

### 8.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as MCGIRCLK, which can be used as an additional clock source. The MCGIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the MCGTRM register. Writing a larger value will decrease the MCGIRCLK frequency, and writing a smaller value to the MCGTRM register will increase the MCGIRCLK frequency. The TRIM bits will effect the MCGOUT frequency if the MCG is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or bypassed low power internal (BLPI) mode. The TRIM and FTRIM value is initialized by POR but is not affected by other resets.

Until MCGIRCLK is trimmed, programming low reference divider (RDIV) factors may result in MCGOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN and IRCLKEN bits are both set, the internal reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

### 8.4.6 External Reference Clock

The MCG module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in FEE and FBE modes, 1 MHz to 16 MHz in PEE and PBE modes, and 0 to 40 MHz in BLPE mode. When ERCLKEN is set, the external reference clock signal will be presented as MCGERCLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL or PLL and will only be used as MCGERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the Device Overview chapter).

If EREFSTEN and ERCLKEN bits are both set or the MCG is in FEE, FBE, PEE, PBE or BLPE mode, the external reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

If CME bit is written to 1, the clock monitor is enabled. If the external reference falls below a certain frequency ($f_{loc_high}$ or $f_{loc_low}$ depending on the RANGE bit in the MCGC2), the MCU will reset. The LOC bit in the System Reset Status (SRS) register will be set to indicate the error.

### 8.4.7 Fixed Frequency Clock

The MCG presents the divided reference clock as MCGFFCLK for use as an additional clock source. The MCGFFCLK frequency must be no more than 1/4 of the MCGOUT frequency to be valid. Because of this requirement, the MCGFFCLK is not valid in bypass modes for the following combinations of BDIV and RDIV values:

- BDIV=00 (divide by 1), RDIV < 010
- BDIV=01 (divide by 2), RDIV < 011

When MCGFFCLK is valid then MCGFFCLKVALID is set to 1. When MCGFFCLK is not valid then MCGFFCLKVALID is set to 0.

## 8.5 Initialization / Application Information

This section describes how to initialize and configure the MCG module in application. The following sections include examples on how to initialize the MCG and properly switch between the various available modes.

### 8.5.1 MCG Module Initialization Sequence

The MCG comes out of reset configured for FEI mode with the BDIV set for divide-by-2. The internal reference will stabilize in $t_{irefst}$ microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in $t_{fll_lock}$ milliseconds.

Upon POR, the internal reference will require trimming to guarantee an accurate clock. Freescale recommends using FLASH location 0xFFAE for storing the fine trim bit, FTRIM in the MCGSC register, and 0xFFAF for storing the 8-bit trim value in the MCGTRM register. The MCU will not automatically copy the values in these FLASH locations to the respective registers. Therefore, user code must copy these values from FLASH to the registers.

**NOTE**

The BDIV value should not be changed to divide-by-1 without first trimming the internal reference. Failure to do so could result in the MCU running out of specification.

#### 8.5.1.1 Initializing the MCG

Because the MCG comes out of reset in FEI mode, the only MCG modes which can be directly switched to upon reset are FEE, FBE, and FBI modes (see Figure 8-8). Reaching any of the other modes requires first configuring the MCG for one of these three initial modes. Care must be taken to check relevant status bits in the MCGSC register reflecting all configuration changes within each mode.

To change from FEI mode to FEE or FBE modes, follow this procedure:

1. Enable the external clock source by setting the appropriate bits in MCGC2.
2. Write to MCGC1 to select the clock mode.

— If entering FEE, set RDIV appropriately, clear the IREFS bit to switch to the external reference, and leave the CLKS bits at %00 so that the output of the FLL is selected as the system clock source.

— If entering FBE, clear the IREFS bit to switch to the external reference and change the CLKS bits to %10 so that the external reference clock is selected as the system clock source. The RDIV bits should also be set appropriately here according to the external reference frequency because although the FLL is bypassed, it is still on in FBE mode.

— The internal reference can optionally be kept running by setting the IRCLKEN bit. This is useful if the application will switch back and forth between internal and external modes. For minimum power consumption, leave the internal reference disabled while in an external clock mode.

3. After the proper configuration bits have been set, wait for the affected bits in the MCGSC register to be changed appropriately, reflecting that the MCG has moved into the proper mode.

— If ERCLKEN was set in step 1 or the MCG is in FEE, FBE, PEE, PBE, or BLPE mode, and EREFS was also set in step 1, wait here for the OSCINIT bit to become set indicating that the external clock source has finished its initialization cycles and stabilized. Typical crystal startup times are given in Appendix A, “Electrical Characteristics”.

— If in FEE mode, check to make sure the IREFST bit is cleared and the LOCK bit is set before moving on.

— If in FBE mode, check to make sure the IREFST bit is cleared, the LOCK bit is set, and the CLKST bits have changed to %10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed in FBE mode, it is still on and will lock in FBE mode.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change the CLKS bits to %01 so that the internal reference clock is selected as the system clock source.
2. Wait for the CLKST bits in the MCGSC register to change to %01, indicating that the internal reference clock has been appropriately selected.

### 8.5.2 MCG Mode Switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another. Each time any of these bits are changed (PLLS, IREFS, CLKS, or EREFS), the corresponding bits in the MCGSC register (PLLST, IREFST, CLKST, or OSCINIT) must be checked before moving on in the application software.

Additionally, care must be taken to ensure that the reference clock divider (RDIV) is set properly for the mode being switched to. For instance, in PEE mode, if using a 4 MHz crystal, RDIV must be set to %001 (divide-by-2) or %010 (divide -by-4) in order to divide the external reference down to the required frequency between 1 and 2 MHz.

The RDIV and IREFS bits should always be set properly before changing the PLLS bit so that the FLL or PLL clock has an appropriate reference clock frequency to switch to.

The table below shows MCGOUT frequency calculations using RDIV, BDIV, and VDIV settings for each clock mode. The bus frequency is equal to MCGOUT divided by 2.

**Table 8-6. MCGOUT Frequency Calculation Options**

| Clock Mode | $f_{MCGOUT}$[1] | Note |
|---|---|---|
| FEI (FLL engaged internal) | ($f_{int}$ * 1024) / B | Typical $f_{MCGOUT}$ = 16 MHz immediately after reset. RDIV bits set to %000. |
| FEE (FLL engaged external) | ($f_{ext}$ / R *1024) / B | $f_{ext}$ / R must be in the range of 31.25 kHz to 39.0625 kHz |
| FBE (FLL bypassed external) | $f_{ext}$ / B | $f_{ext}$ / R must be in the range of 31.25 kHz to 39.0625 kHz |
| FBI (FLL bypassed internal) | $f_{int}$ / B | Typical $f_{int}$ = 32 kHz |
| PEE (PLL engaged external) | [($f_{ext}$ / R) * M] / B | $f_{ext}$ / R must be in the range of 1 MHz to 2 MHz |
| PBE (PLL bypassed external) | $f_{ext}$ / B | $f_{ext}$ / R must be in the range of 1 MHz to 2 MHz |
| BLPI (Bypassed low power internal) | $f_{int}$ / B | |
| BLPE (Bypassed low power external) | $f_{ext}$ / B | |

1 R is the reference divider selected by the RDIV bits, B is the bus frequency divider selected by the BDIV bits, and M is the multiplier selected by the VDIV bits.

This section will include 3 mode switching examples using a 4 MHz external crystal. If using an external clock source less than 1 MHz, the MCG should not be configured for any of the PLL modes (PEE and PBE).

### 8.5.2.1 Example # 1: Moving from FEI to PEE Mode: External Crystal = 4 MHz, Bus Frequency = 8 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE mode until the 4 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz. Because the MCG is in FEI mode out of reset, this example also shows how to initialize the MCG for PEE mode out of reset. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, FEI must transition to FBE mode:
   a) MCGC2 = 0x36 (%00110110)
      – BDIV (bits 7 and 6) set to %00, or divide-by-1
      – RANGE (bit 5) set to 1 because the frequency of 4 MHz is within the high frequency range
      – HGO (bit 4) set to 1 to configure external oscillator for high gain operation
      – EREFS (bit 2) set to 1, because a crystal is being used
      – ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
   b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.

c) MCGC1 = 0xB8 (%10111000)
   - CLKS (bits 7 and 6) set to %10 in order to select external reference clock as system clock source
   - RDIV (bits 5-3) set to %111, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
   - IREFS (bit 2) cleared to 0, selecting the external reference clock

d) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference is the current source for the reference clock

e) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT

2. Then, FBE must transition either directly to PBE mode or first through BLPE mode and then to PBE mode:

a) BLPE: If a transition through BLPE mode is desired, first set LP (bit 3) in MCGC2 to 1.

b) BLPE/PBE: MCGC1 = 0x90 (%10010000)
   - RDIV (bits 5-3) set to %010, or divide-by-4 because 4 MHz / 4 = 1 MHz which is in the 1 MHz to 2 MHz range required by the PLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the the dividers for PLL usage in PBE mode

c) BLPE/PBE: MCGC3 = 0x44 (%01000100)
   - PLLS (bit 6) set to 1, selects the PLL. In BLPE mode, changing this bit only prepares the MCG for PLL usage in PBE mode
   - VDIV (bits 3-0) set to %0100, or multiply-by-16 because 1 MHz reference * 16 = 16 MHz. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode

d) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode

e) PBE: Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL

f) PBE: Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock

3. Last, PBE mode transitions into PEE mode:

a) MCGC1 = 0x10 (%00010000)
   - CLKS (bits7 and 6) in MCGSC1 set to %00 in order to select the output of the PLL as the system clock source

b) Loop until CLKST (bits 3 and 2) in MCGSC are %11, indicating that the PLL output is selected to feed MCGOUT in the current clock mode
   - Now, With an RDIV of divide-by-4, a BDIV of divide-by-1, and a VDIV of multiply-by-16, MCGOUT = [(4 MHz / 4) * 16] / 1 = 16 MHz, and the bus frequency is MCGOUT / 2, or 8 MHz

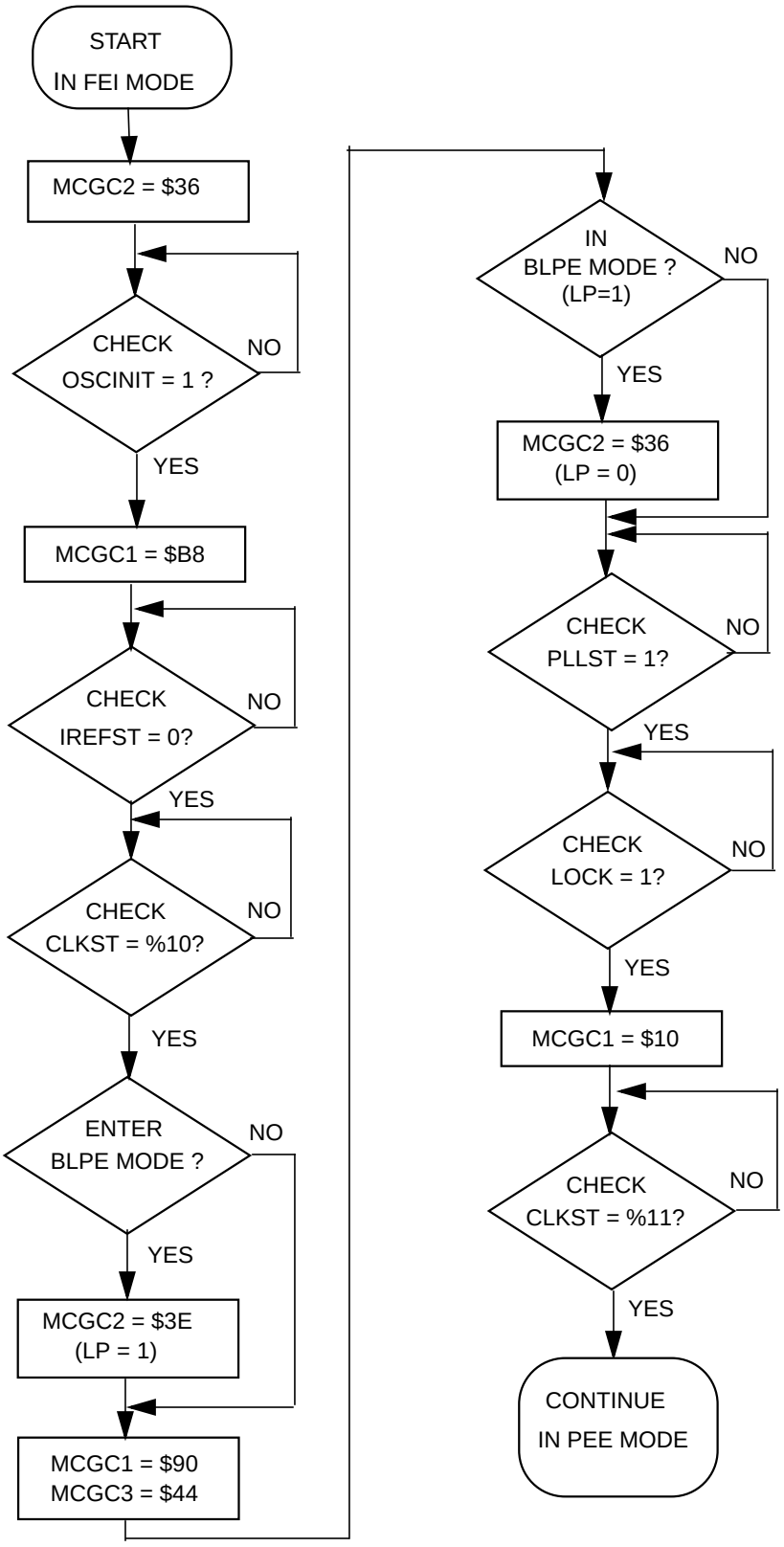



**Figure 8-9. Flowchart of FEI to PEE Mode Transition using a 4 MHz crystal**

#### 8.5.2.2 Example # 2: Moving from PEE to BLPI Mode: External Crystal = 4 MHz, Bus Frequency =16 kHz

In this example, the MCG will move through the proper operational modes from PEE mode with a 4 MHz crystal configured for an 8 MHz bus frequency (see previous example) to BLPI mode with a 16 kHz bus frequency.First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, PEE must transition to PBE mode:
   a) MCGC1 = 0x90 (%10010000)
      – CLKS (bits 7 and 6) set to %10 in order to switch the system clock source to the external reference clock
   b) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, PBE must transition either directly to FBE mode or first through BLPE mode and then to FBE mode:
   a) BLPE: If a transition through BLPE mode is desired, first set LP (bit 3) in MCGC2 to 1
   b) BLPE/FBE: MCGC1 = 0xB8 (%10111000)
      – RDIV (bits 5-3) set to %111, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the dividers for FLL usage in FBE mode
   c) BLPE/FBE: MCGC3 = 0x04 (%00000100)
      – PLLS (bit 6) clear to 0 to select the FLL. In BLPE mode, changing this bit only prepares the MCG for FLL usage in FBE mode. With PLLS = 0, the VDIV value does not matter.
   d) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to FBE mode
   e) FBE: Loop until PLLST (bit 5) in MCGSC is clear, indicating that the current source for the PLLS clock is the FLL
   f) FBE: Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has acquired lock. Although the FLL is bypassed in FBE mode, it is still enabled and running.
3. Next, FBE mode transitions into FBI mode:
   a) MCGC1 = 0x44 (%01000100)
      – CLKS (bits7 and 6) in MCGSC1 set to %01 in order to switch the system clock to the internal reference clock
      – IREFS (bit 2) set to 1 to select the internal reference clock as the reference clock source
      – RDIV (bits 5-3) set to %000, or divide-by-1 because the trimmed internal reference should be within the 31.25 kHz to 39.0625 kHz range required by the FLL
   b) Loop until IREFST (bit 4) in MCGSC is 1, indicating the internal reference clock has been selected as the reference clock source
   c) Loop until CLKST (bits 3 and 2) in MCGSC are %01, indicating that the internal reference clock is selected to feed MCGOUT

4. Lastly, FBI transitions into BLPI mode.
   a) MCGC2 = 0x08 (%00001000)
      – LP (bit 3) in MCGSC is 1

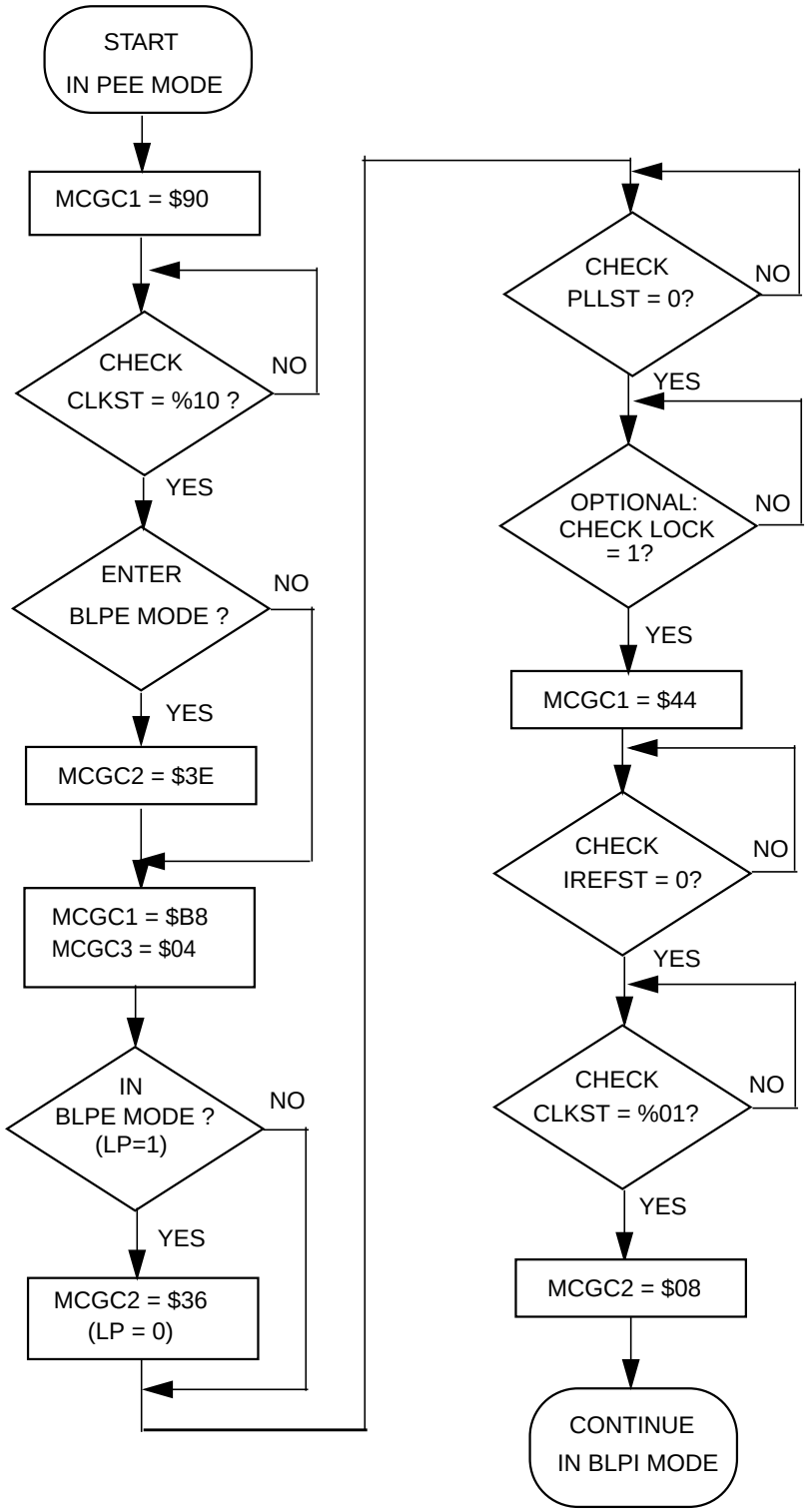



**Figure 8-10. Flowchart of PEE to BLPI Mode Transition using a 4 MHz crystal**

#### 8.5.2.3 Example #3: Moving from BLPI to FEE Mode: External Crystal = 4 MHz, Bus Frequency = 16 MHz

In this example, the MCG will move through the proper operational modes from BLPI mode at a 16 kHz bus frequency running off of the internal reference clock (see previous example) to FEE mode using a 4 MHz crystal configured for a 16 MHz bus frequency. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, BLPI must transition to FBI mode.
   a) MCGC2 = 0x00 (%00000000)
      – LP (bit 3) in MCGSC is 0
   b) Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has acquired lock. Although the FLL is bypassed in FBI mode, it is still enabled and running.
2. Next, FBI will transition to FEE mode.
   a) MCGC2 = 0x36 (%00110110)
      – RANGE (bit 5) set to 1 because the frequency of 4 MHz is within the high frequency range
      – HGO (bit 4) set to 1 to configure external oscillator for high gain operation
      – EREFS (bit 2) set to 1, because a crystal is being used
      – ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
   b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.
   c) MCGC1 = 0x38 (%00111000)
      – CLKS (bits 7 and 6) set to %00 in order to select the output of the FLL as system clock source
      – RDIV (bits 5-3) set to %111, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
      – IREFS (bit 1) cleared to 0, selecting the external reference clock
   d) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference clock is the current source for the reference clock
   e) Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has reacquired lock.
   f) Loop until CLKST (bits 3 and 2) in MCGSC are %00, indicating that the output of the FLL is selected to feed MCGOUT

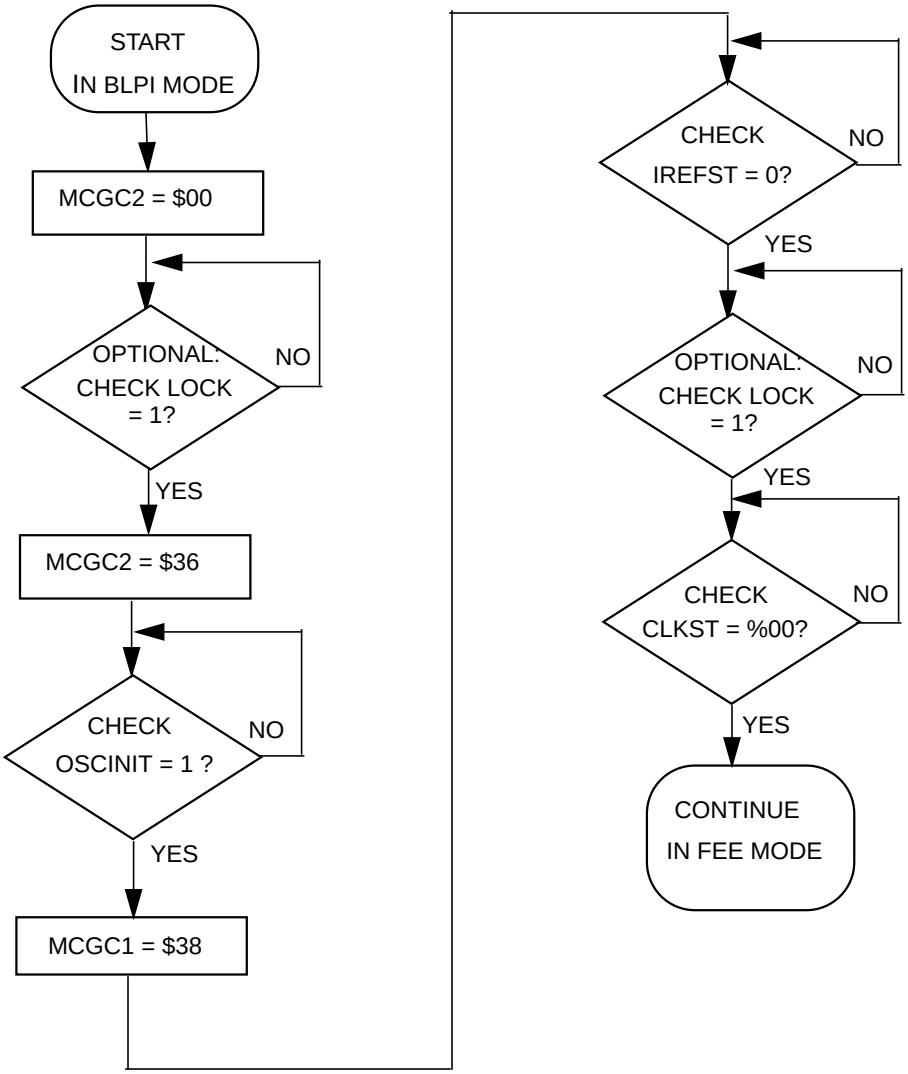



**Figure 8-11. Flowchart of BLPI to FEE Mode Transition using a 4 MHz crystal**

### 8.5.2.4 Example # 4: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 8 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE mode until the 8 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz.

This example is similar to example number one except that in this case the frequency of the external crystal is 8 MHz instead of 4 MHz. Special consideration must be taken with this case since there is a period of time along the way from FEI mode to PEE mode where the FLL operates based on a reference clock with a frequency that is greater than the maximum allowed for the FLL. This occurs because with an 8 MHz

external crystal and a maximum reference divider factor of 128, the resulting frequency of the reference clock for the FLL is 62.5 kHz (greater than the 39.0625 kHz maximum allowed).

Care must be taken in the software to minimize the amount of time spent in this state where the FLL is operating in this condition.

The following code sequence describes how to move from FEI mode to PEE mode until the 8 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz. Because the MCG is in FEI mode out of reset, this example also shows how to initialize the MCG for PEE mode out of reset. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, FEI must transition to FBE mode:
   a) MCGC2 = 0x36 (%00110110)
      – BDIV (bits 7 and 6) set to %00, or divide-by-1
      – RANGE (bit 5) set to 1 because the frequency of 8 MHz is within the high frequency range
      – HGO (bit 4) set to 1 to configure external oscillator for high gain operation
      – EREFS (bit 2) set to 1, because a crystal is being used
      – ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
   b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.
   c) Block Interrupts (If applicable by setting the interrupt bit in the CCR).
   d) MCGC1 = 0xB8 (%10111000)
      – CLKS (bits 7 and 6) set to %10 in order to select external reference clock as system clock source
      – RDIV (bits 5-3) set to %111, or divide-by-128.

**NOTE**

8 MHz / 128 = 62.5 kHz which is greater than the 31.25 kHz to 39.0625 kHz range required by the FLL. Therefore after the transition to FBE is complete, software must progress through to BLPE mode immediately by setting the LP bit in MCGC2.

      – IREFS (bit 2) cleared to 0, selecting the external reference clock
   e) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference is the current source for the reference clock
   f) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, FBE mode transitions into BLPE mode:
   a) MCGC2 = 0x3E (%00111110)
      – LP (bit 3) in MCGC2 to 1 (BLPE mode entered)

**NOTE**

There must be no extra steps (including interrupts) between steps 1d and 2a.

   b) Enable Interrupts (if applicable by clearing the interrupt bit in the CCR).

c) MCGC1 = 0x98 (%10011000)
   – RDIV (bits 5-3) set to %011, or divide-by-8 because 8 MHz / 8= 1 MHz which is in the 1 MHz to 2 MHz range required by the PLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the the dividers for PLL usage in PBE mode

d) MCGC3 = 0x44 (%01000100)
   – PLLS (bit 6) set to 1, selects the PLL. In BLPE mode, changing this bit only prepares the MCG for PLL usage in PBE mode
   – VDIV (bits 3-0) set to %0100, or multiply-by-16 because 1 MHz reference * 16 = 16 MHz. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode

e) Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL

3. Then, BLPE mode transitions into PBE mode:
   a) Clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode
   b) Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock
4. Last, PBE mode transitions into PEE mode:
   a) MCGC1 = 0x18 (%00011000)
      – CLKS (bits7 and 6) in MCGSC1 set to %00 in order to select the output of the PLL as the system clock source
   b) Loop until CLKST (bits 3 and 2) in MCGSC are %11, indicating that the PLL output is selected to feed MCGOUT in the current clock mode
      – Now, With an RDIV of divide-by-8, a BDIV of divide-by-1, and a VDIV of multiply-by-16, MCGOUT = [(8 MHz / 8) * 16] / 1 = 16 MHz, and the bus frequency is MCGOUT / 2, or 8 MHz

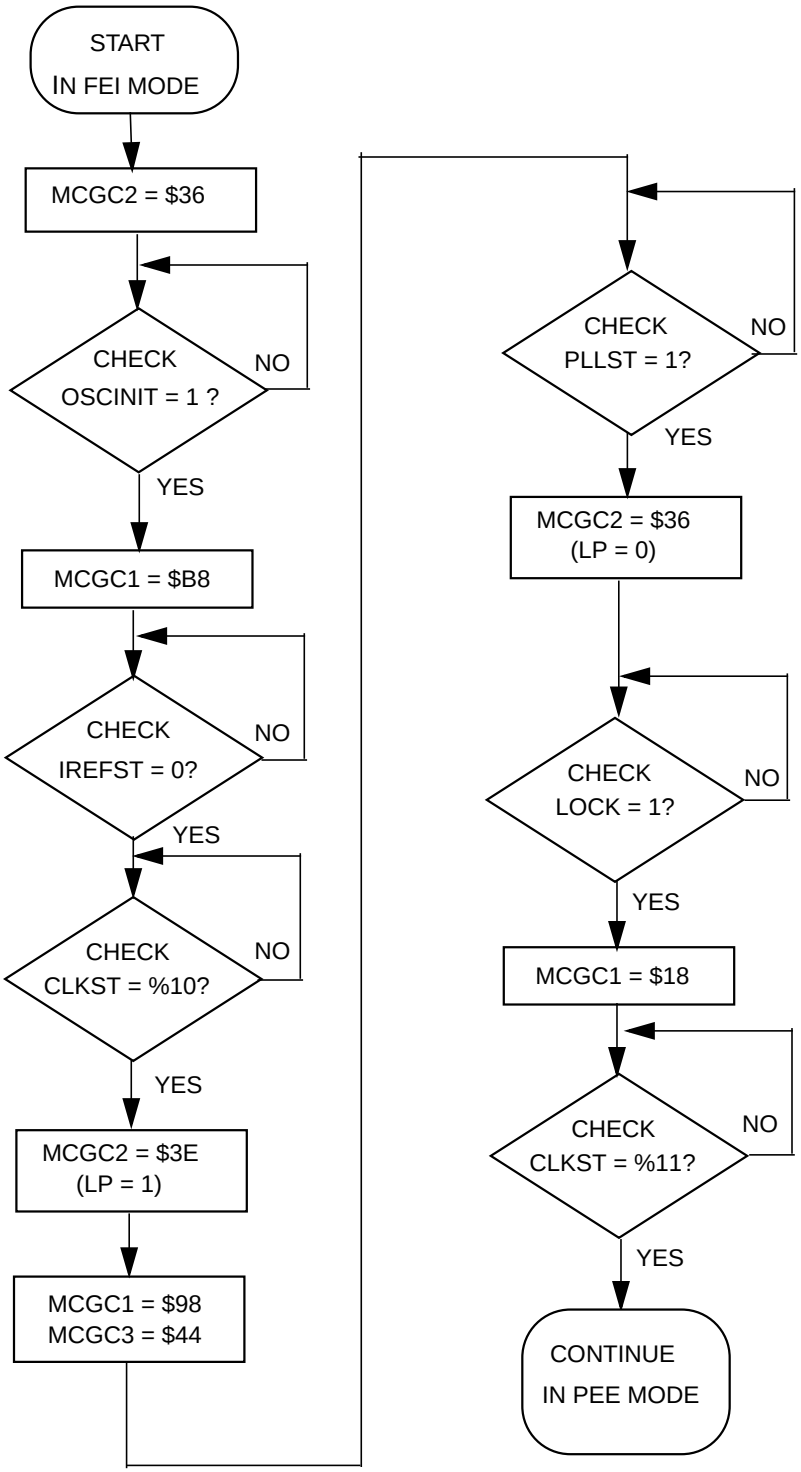



**Figure 8-12. Flowchart of FEI to PEE Mode Transition using a 8 MHz crystal**

### 8.5.3 Calibrating the Internal Reference Clock (IRC)

The IRC is calibrated by writing to the MCGTRM register first, then using the FTRIM bit to “fine tune” the frequency. We will refer to this total 9-bit value as the trim value, ranging from 0x000 to 0x1FF, where the FTRIM bit is the LSB.

The trim value after a POR is always 0x100 (MCGTRM = 0x80 and FTRIM = 0). Writing a larger value will decrease the frequency and smaller values will increase the frequency. The trim value is linear with the period, except that slight variations in wafer fab processing produce slight non-linearities between trim value and period. These non-linearities are why an iterative trimming approach to search for the best trim value is recommended. In Example #5: Internal Reference Clock Trim this approach will be demonstrated.

After a trim value has been found for a device, this value can be stored in FLASH memory to save the value. If power is removed from the device, the IRC can easily be re-trimmed by copying the saved value from FLASH to the MCG registers. Freescale identifies recommended FLASH locations for storing the trim value for each MCU. Consult the memory map in the data sheet for these locations. On devices that are factory trimmed, the factory trim value will be stored in these locations.

#### 8.5.3.1 Example #5: Internal Reference Clock Trim

For applications that require a tight frequency tolerance, a trimming procedure is provided that will allow a very accurate internal clock source. This section outlines one example of trimming the internal oscillator. Many other possible trimming procedures are valid and can be used.

In the example below, the MCG trim will be calibrated for the 9-bit MCGTRM and FTRIM collective value. This value will be referred to as TRMVAL.

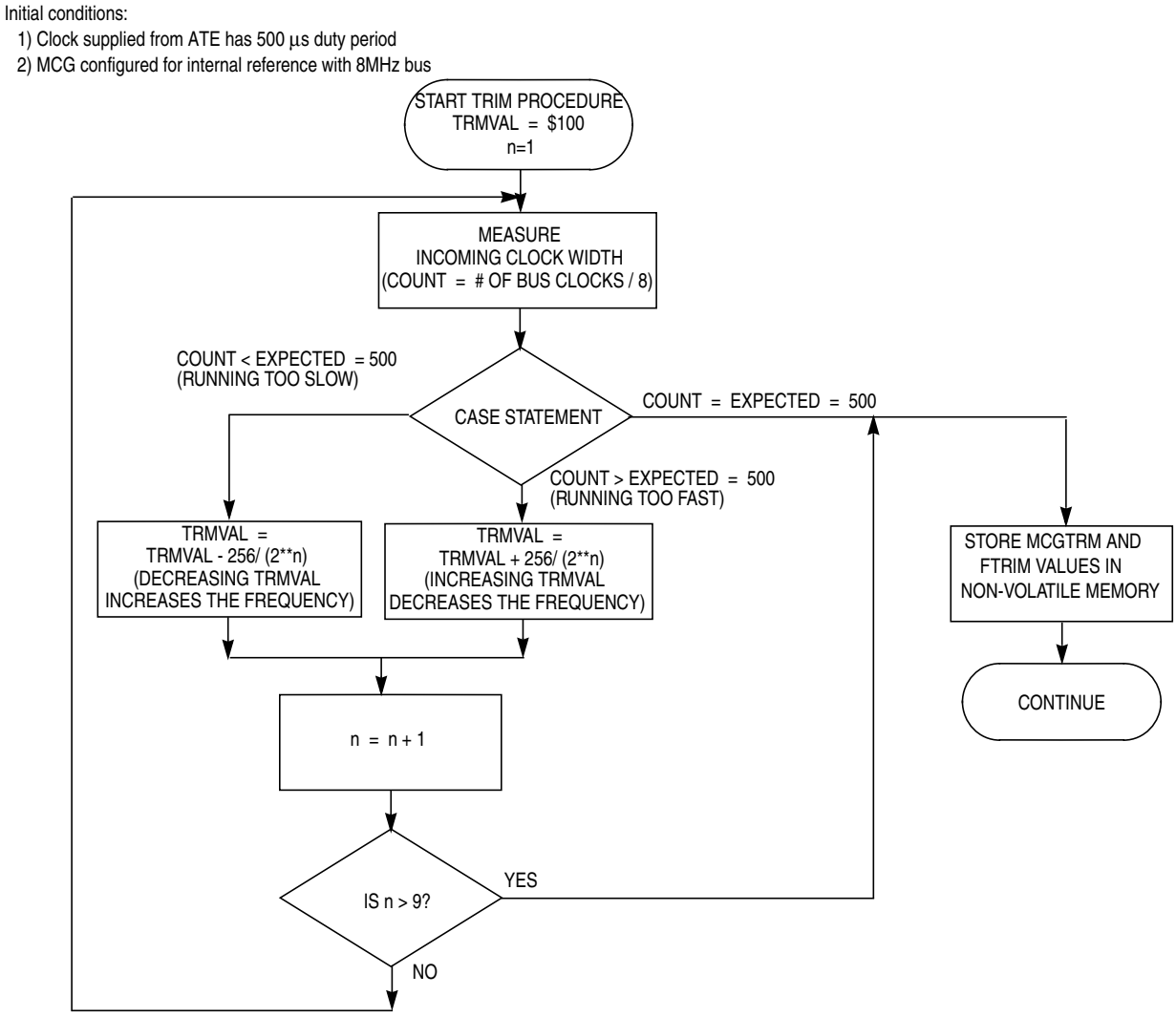



**Figure 8-13. Trim Procedure**

In this particular case, the MCU has been attached to a PCB and the entire assembly is undergoing final test with automated test equipment. A separate signal or message is provided to the MCU operating under user provided software control. The MCU initiates a trim procedure as outlined in Figure 8-13 while the tester supplies a precision reference signal.

If the intended bus frequency is near the maximum allowed for the device, it is recommended to trim using a reference divider value (RDIV setting) of twice the final value. After the trim procedure is complete, the reference divider can be restored. This will prevent accidental overshoot of the maximum clock frequency.

# Chapter 9
# Analog Comparator (S08ACMPV3)

## 9.1 Introduction

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages or for comparing one analog input voltage to an internal reference voltage. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

All MC9S08DZ60 Series MCUs have two full function ACMPs in a 64-pin package. MCUs in the 48-pin package have two ACMPs, but the output of ACMP2 is not accessible. MCUs in the 32-pin package contain one full function ACMP.

**NOTE**

MC9S08DZ60 Series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Please ignore references to stop1.

### 9.1.1 ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP+, the user must enable the bandgap buffer by setting BGBE =1 in SPMSC1 see Section 5.8.7, “System Power Management Status and Control 1 Register (SPMSC1).” For value of bandgap voltage reference see Section A.6, “DC Characteristics.”

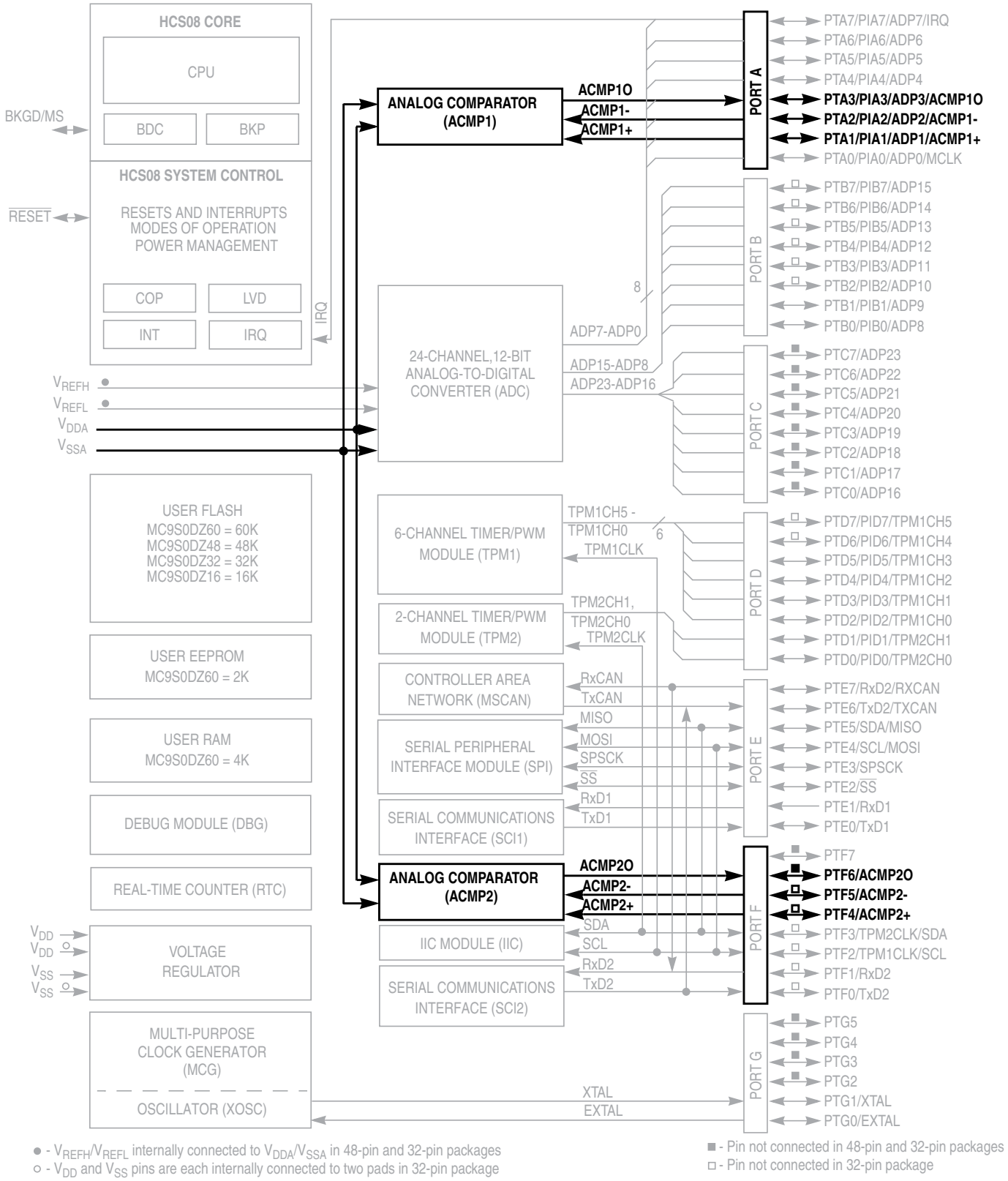



**Figure 9-1. MC9S08DZ60 Block Diagram**

## 9.1.2 Features

The ACMP has the following features:

- Full rail to rail supply operation.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Option to compare to fixed internal bandgap reference voltage.
- Option to allow comparator output to be visible on a pin, ACMPxO.

## 9.1.3 Modes of Operation

This section defines the ACMP operation in wait, stop, and background debug modes.

### 9.1.3.1 ACMP in Wait Mode

The ACMP continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the ACMP can be used to bring the MCU out of wait mode if the ACMP interrupt is enabled (ACIE is set). For lowest possible current consumption, the ACMP should be disabled by software if not required as an interrupt source during wait mode.

### 9.1.3.2 ACMP in Stop Modes

The ACMP is disabled in all stop modes, regardless of the settings before executing the stop instruction. Therefore, the ACMP cannot be used as a wake up source from stop modes.

During stop2 mode, the ACMP module is fully powered down. Upon wake-up from stop2 mode, the ACMP module is in the reset state.

During stop3 mode, clocks to the ACMP module are halted. No registers are affected. In addition, the ACMP comparator circuit enters a low-power state. No compare operation occurs while in stop3.

If stop3 is exited with a reset, the ACMP is put into its reset state. If stop3 is exited with an interrupt, the ACMP continues from the state it was in when stop3 was entered.

### 9.1.3.3 ACMP in Active Background Mode

When the microcontroller is in active background mode, the ACMP continues to operate normally.

### 9.1.4 Block Diagram

The block diagram for the analog comparator module is shown Figure 9-2.

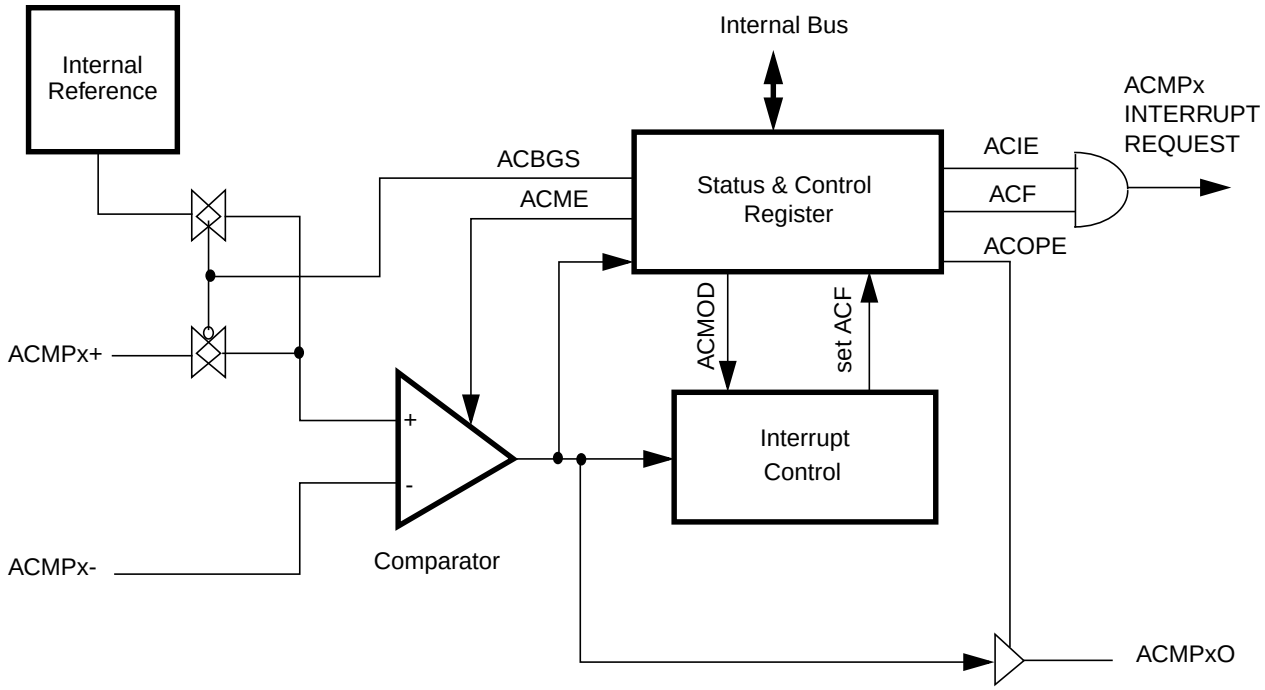



**Figure 9-2. Analog Comparator (ACMP) Block Diagram**

## 9.2 External Signal Description

The ACMP has two analog input pins, ACMPx+ and ACMPx– and one digital output pin ACMPxO. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. As shown in Figure 9-2, the ACMPx- pin is connected to the inverting input of the comparator, and the ACMPx+ pin is connected to the comparator non-inverting input if ACBGS is a 0. As shown in Figure 9-2, the ACMPxO pin can be enabled to drive an external pin.

The signal properties of ACMP are shown in Table 9-1.

**Table 9-1. Signal Properties**

| Signal | Function | I/O |
|---|---|---|
| ACMPx- | Inverting analog input to the ACMP. (Minus input) | I |
| ACMPx+ | Non-inverting analog input to the ACMP. (Positive input) | I |
| ACMPxO | Digital output of the ACMP. | O |

## 9.3 Memory Map/Register Definition

The ACMP includes one register:

- An 8-bit status and control register

Refer to the direct-page register summary in the memory section of this document for the absolute address assignments for the ACMP register.This section refers to register and control bits only by their names and relative address offsets.

Some MCUs may have more than one ACMP, so register names include placeholder characters (x) to identify which ACMP is being referenced.

**Table 9-2. ACMP Register Summary**

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| ACMPxSC | R | ACME | ACBGS | ACF | ACIE | ACO | ACOPE | ACMOD | |
| | W | | | | | | | | |

### 9.3.1 ACMPx Status and Control Register (ACMPxSC)

ACMPxSC contains the status flag and control bits used to enable and configure the ACMP.

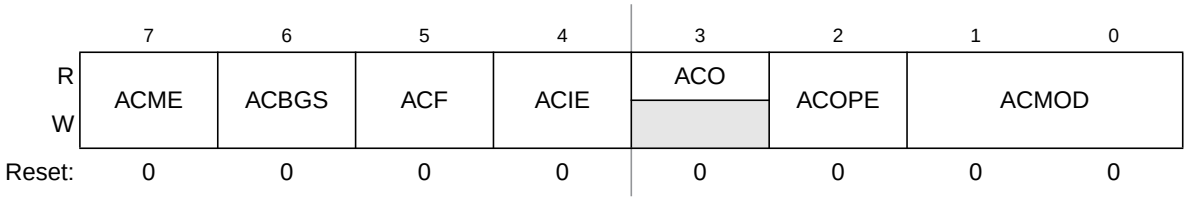



**Figure 9-3. ACMPx Status and Control Register (ACMPxSC)**

**Table 9-3. ACMPxSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ACME | Analog Comparator Module Enable. Enables the ACMP module.<br>0 ACMP not enabled<br>1 ACMP is enabled |
| 6<br>ACBGS | Analog Comparator Bandgap Select. Selects between the bandgap reference voltage or the ACMPx+ pin as the input to the non-inverting input of the analog comparator.<br>0 External pin ACMPx+ selected as non-inverting input to comparator<br>1 Internal reference select as non-inverting input to comparator |
| 5<br>ACF | Analog Comparator Flag. ACF is set when a compare event occurs. Compare events are defined by ACMOD. ACF is cleared by writing a one to it.<br>0 Compare event has not occurred<br>1 Compare event has occurred |
| 4<br>ACIE | Analog Comparator Interrupt Enable. Enables the interrupt from the ACMP. When ACIE is set, an interrupt is asserted when ACF is set.<br>0 Interrupt disabled<br>1 Interrupt enabled |

**Table 9-3. ACMPxSC Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ACO | Analog Comparator Output. Reading ACO returns the current value of the analog comparator output. ACO is reset to a 0 and reads as a 0 when the ACMP is disabled (ACME = 0). |
| 2<br>ACOPE | Analog Comparator Output Pin Enable. Enables the comparator output to be placed onto the external pin, ACMPxO.<br>0 Analog comparator output not available on ACMPxO<br>1 Analog comparator output is driven out on ACMPxO |
| 1:0<br>ACMOD | Analog Comparator Mode. ACMOD selects the type of compare event which sets ACF.<br>00 Encoding 0 — Comparator output falling edge<br>01 Encoding 1 — Comparator output rising edge<br>10 Encoding 2 — Comparator output falling edge<br>11 Encoding 3 — Comparator output rising or falling edge |

## 9.4 Functional Description

The analog comparator can compare two analog input voltages applied to ACMPx+ and ACMPx–, or it can compare an analog input voltage applied to ACMPx– with an internal bandgap reference voltage. ACBGS selects between the bandgap reference voltage or the ACMPx+ pin as the input to the non-inverting input of the analog comparator. The comparator output is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. ACMOD selects the condition that causes ACF to be set. ACF can be set on a rising edge of the comparator output, a falling edge of the comparator output, or a rising or a falling edge (toggle). The comparator output can be read directly through ACO. The comparator output can be driven onto the ACMPxO pin using ACOPE.

# Chapter 10
# Analog-to-Digital Converter (S08ADC12V1)

## 10.1 Introduction

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

**NOTE**

MC9S08DZ60 Series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Please ignore references to stop1.

### 10.1.1 Analog Power and Ground Signal Names

References to $V_{DDAD}$ and $V_{SSAD}$ in this chapter correspond to signals $V_{DDA}$ and $V_{SSA}$, respectively.

### 10.1.2 Channel Assignments

**NOTE**

The ADC channel assignments for the MC9S08DZ60 Series devices are shown in Table 10-1. Reserved channels convert to an unknown value.

This chapter shows bits for all S08ADC12V1 channels. MC9S08DZ60 Series MCUs do not use all of these channels. All bits corresponding to channels that are not available on a device are reserved.

**Table 10-1. ADC Channel Assignment**

| ADCH | Channel | Input |
|---|---|---|
| 00000 | AD0 | PTA0/ADP0/MCLK |
| 00001 | AD1 | PTA1/ADP1/ACMP1+ |
| 00010 | AD2 | PTA2/ADP2/ACMP1P- |
| 00011 | AD3 | PTA3/ADP3/ACMP1O |
| 00100 | AD4 | PTA4/ADP4 |
| 00101 | AD5 | PTA5/ADP5 |
| 00110 | AD6 | PTA6/ADP6 |
| 00111 | AD7 | PTA7/ADP7 |
| 01000 | AD8 | PTB0/ADP8 |
| 01001 | AD9 | PTB1/ADP9 |
| 01010 | AD10 | PTB2/ADP10 |
| 01011 | AD11 | PTB3/ADP11 |
| 01100 | AD12 | PTB4/ADP12 |
| 01101 | AD13 | PTB5/ADP13 |
| 01110 | AD14 | PTB6/ADP14 |
| 01111 | AD15 | PTB7/ADP15 |
| 10000 | AD16 | PTC0/ADP16 |
| 10001 | AD17 | PTC1/ADP17 |
| 10010 | AD18 | PTC2/ADP18 |
| 10011 | AD19 | PTC3/ADP19 |
| 10100 | AD20 | PTC4/ADP20 |
| 10101 | AD21 | PTC5/ADP21 |
| 10110 | AD22 | PTC6/ADP22 |
| 10111 | AD23 | PTC7/ADP23 |
| 11000–11001 | AD24 through AD25 | Reserved |
| 11010 | AD26 | Temperature Sensor[1] |
| 11011 | AD27 | Internal Bandgap[2] |
| 11100 | Reserved | Reserved |
| 11101 | $V_{REFH}$ | $V_{REFH}$ |

## 10.1.3 Alternate Clock

The ADC module is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock, ALTCLK. The alternate clock for the MC9S08DZ60 Series MCU devices is the external reference clock (MCGERCLK).

The selected clock source must run at a frequency such that the ADC conversion clock (ADCK) runs at a frequency within its specified range ($f_{ADCK}$) after being divided down from the ALTCLK input as determined by the ADIV bits.

ALTCLK is active while the MCU is in wait mode provided the conditions described above are met. This allows ALTCLK to be used as the conversion clock source for the ADC while the MCU is in wait mode.

ALTCLK cannot be used as the ADC conversion clock source while the MCU is in either stop2 or stop3.

## 10.1.4 Hardware Trigger

The ADC hardware trigger, ADHWT, is the output from the real time counter (RTC). The RTC counter can be clocked by either MCGERCLK or a nominal 1 kHz clock source.

The period of the RTC is determined by the input clock frequency, the RTCPS bits, and the RTCMOD register. When the ADC hardware trigger is enabled, a conversion is initiated upon an RTC counter overflow.

The RTC can be configured to cause a hardware trigger in MCU run, wait, and stop3.

### 10.1.5 Temperature Sensor

To use the on-chip temperature sensor, the user must perform the following:

- Configure ADC for long sample with a maximum of 1 MHz clock
- Convert the bandgap voltage reference channel (AD27)
  - By converting the digital value of the bandgap voltage reference channel using the value of $V_{BG}$ the user can determine $V_{DD}$. For value of bandgap voltage, see Section A.6, “DC Characteristics”.
- Convert the temperature sensor channel (AD26)
  - By using the calculated value of $V_{DD}$, convert the digital value of AD26 into a voltage, $V_{TEMP}$

Equation 10-1 provides an approximate transfer function of the temperature sensor.

$$Temp = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 10-1}$$

where:

- $V_{TEMP}$ is the voltage of the temperature sensor channel at the ambient temperature.
- $V_{TEMP25}$ is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the $V_{TEMP25}$ and m values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates $V_{TEMP}$, and compares to $V_{TEMP25}$. If $V_{TEMP}$ is greater than $V_{TEMP25}$ the cold slope value is applied in Equation 10-1. If $V_{TEMP}$ is less than $V_{TEMP25}$ the hot slope value is applied in Equation 10-1. To improve accuracy the user should calibrate the bandgap voltage reference and temperature sensor.

Calibrating at 25°C will improve accuracy to ± 4.5°C.

Calibration at three points, -40°C, 25°C, and 125°C will improve accuracy to ± 2.5°C. Once calibration has been completed, the user will need to calculate the slope for both hot and cold. In application code, the user would then calculate the temperature using Equation 10-1 as detailed above and then determine if the temperature is above or below 25°C. Once determined if the temperature is above or below 25°C, the user can recalculate the temperature using the hot or cold slope value obtained during calibration.

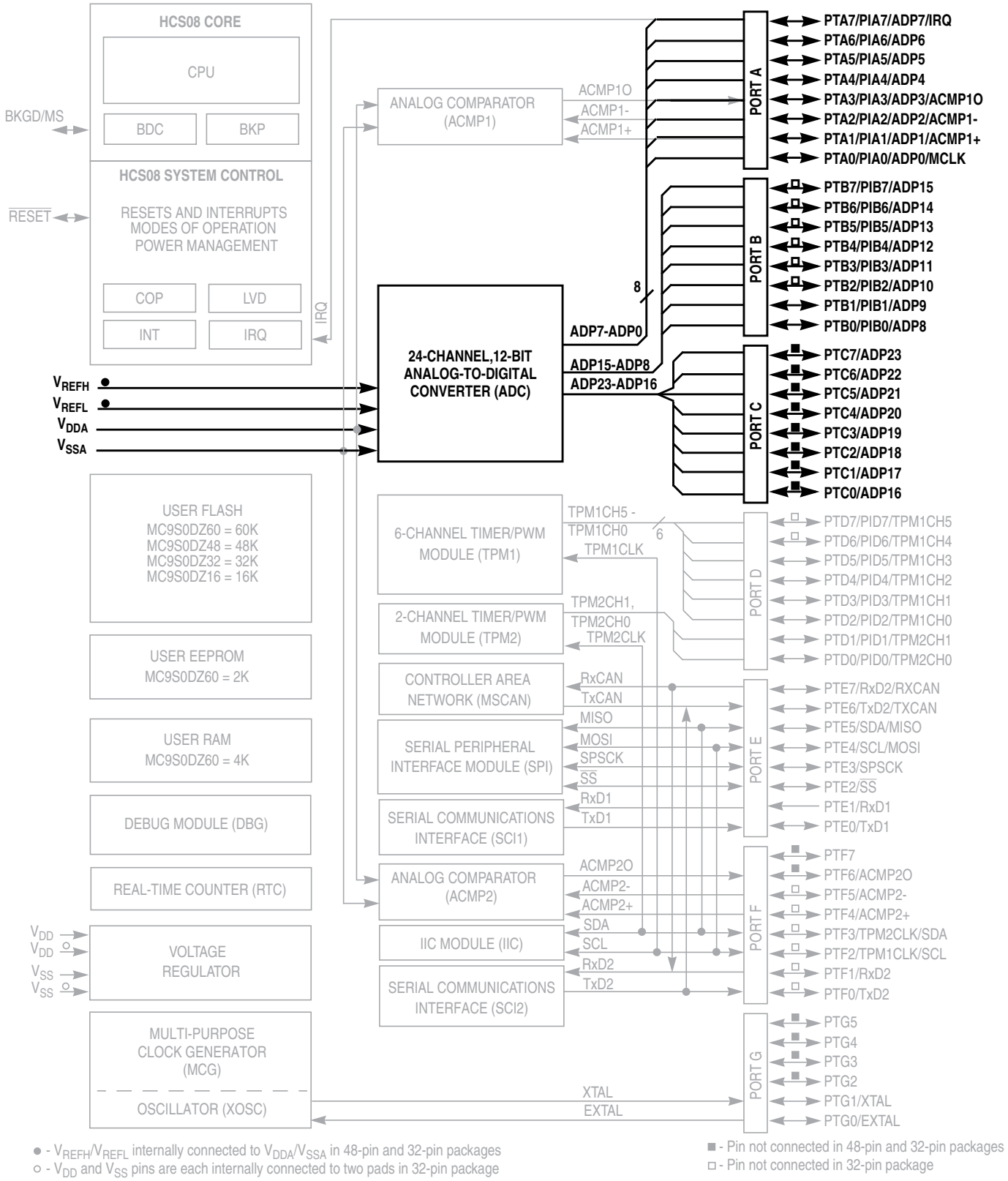



**Figure 10-1. MC9S08DZ60 Block Diagram Emphasizing the ADC Module and Pins**

### 10.1.6 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bit resolution
- Up to 28 analog inputs
- Output formatted in 12-, 10-, or 8-bit right-justified unsigned format
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in wait or stop3 modes for lower noise operation
- Asynchronous clock source for lower noise operation
- Selectable asynchronous hardware conversion trigger
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value
- Temperature sensor

### 10.1.7 ADC Module Block Diagram

Figure 10-2 provides a block diagram of the ADC module.

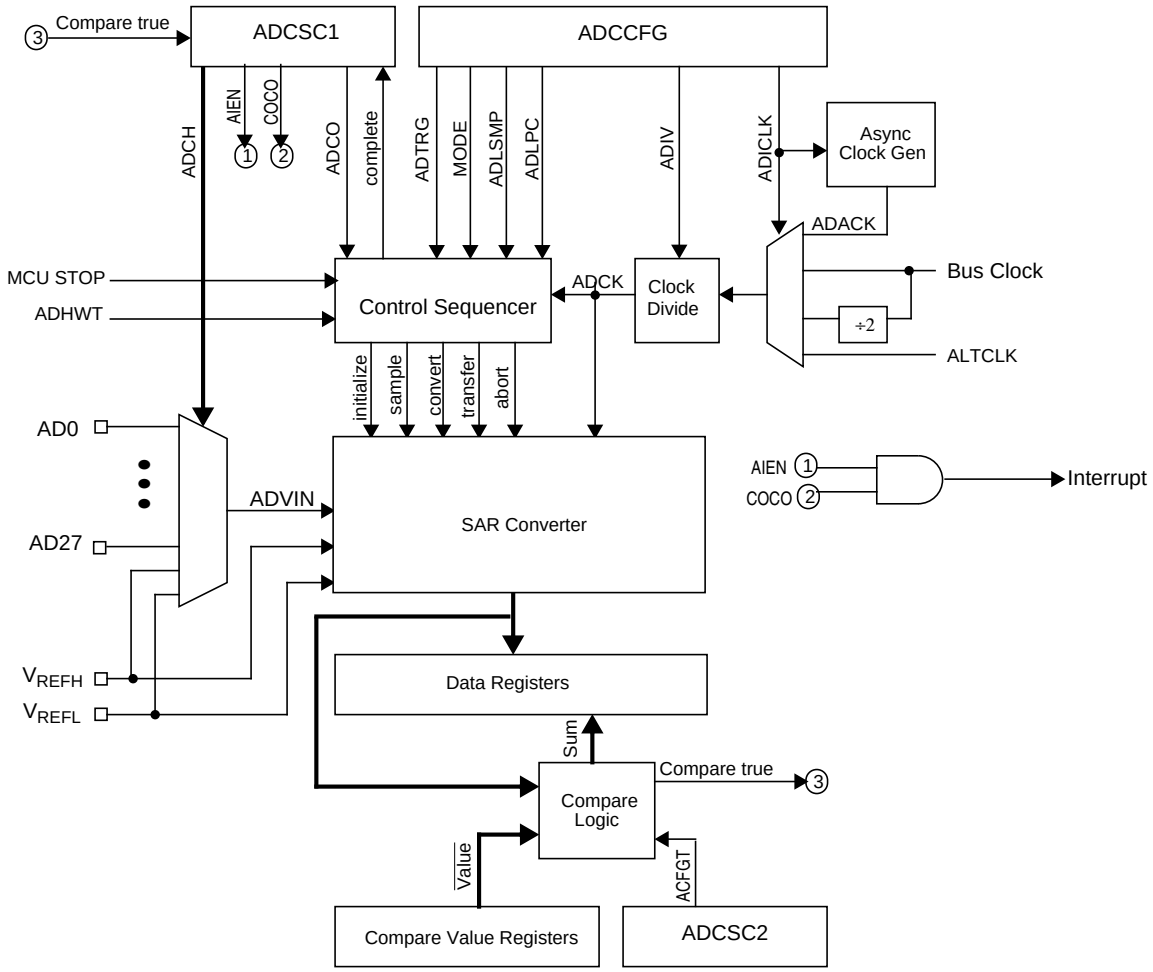



**Figure 10-2. ADC Block Diagram**

## 10.2 External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

**Table 10-2. Signal Properties**

| Name | Function |
|---|---|
| AD27–AD0 | Analog Channel inputs |
| $V_{REFH}$ | High reference voltage |
| $V_{REFL}$ | Low reference voltage |
| $V_{DDAD}$ | Analog power supply |
| $V_{SSAD}$ | Analog ground |

### 10.2.1 Analog Power ($V_{DDAD}$)

The ADC analog portion uses $V_{DDAD}$ as its power connection. In some packages, $V_{DDAD}$ is connected internally to $V_{DD}$. If externally available, connect the $V_{DDAD}$ pin to the same voltage potential as $V_{DD}$. External filtering may be necessary to ensure clean $V_{DDAD}$ for good results.

### 10.2.2 Analog Ground ($V_{SSAD}$)

The ADC analog portion uses $V_{SSAD}$ as its ground connection. In some packages, $V_{SSAD}$ is connected internally to $V_{SS}$. If externally available, connect the $V_{SSAD}$ pin to the same voltage potential as $V_{SS}$.

### 10.2.3 Voltage Reference High ($V_{REFH}$)

$V_{REFH}$ is the high reference voltage for the converter. In some packages, $V_{REFH}$ is connected internally to $V_{DDAD}$. If externally available, $V_{REFH}$ may be connected to the same potential as $V_{DDAD}$ or may be driven by an external source between the minimum $V_{DDAD}$ spec and the $V_{DDAD}$ potential ($V_{REFH}$ must never exceed $V_{DDAD}$).

### 10.2.4 Voltage Reference Low ($V_{REFL}$)

$V_{REFL}$ is the low-reference voltage for the converter. In some packages, $V_{REFL}$ is connected internally to $V_{SSAD}$. If externally available, connect the $V_{REFL}$ pin to the same voltage potential as $V_{SSAD}$.

### 10.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

## 10.3 Register Definition

These memory-mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin control registers, APCTL1, APCTL2, APCTL3

### 10.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

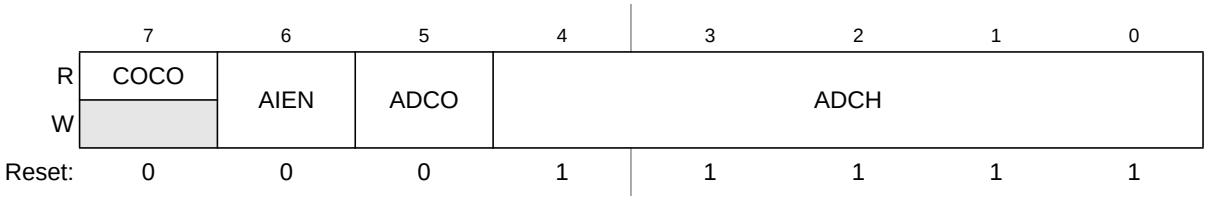



**Figure 10-3. Status and Control Register (ADCSC1)**

**Table 10-3. ADCSC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COCO | Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared when ADCSC1 is written or when ADCRL is read.<br>0 Conversion not completed<br>1 Conversion completed |
| 6<br>AIEN | Interrupt Enable AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted.<br>0 Conversion complete interrupt disabled<br>1 Conversion complete interrupt enabled |
| 5<br>ADCO | Continuous Conversion Enable. ADCO enables continuous conversions.<br>0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected.<br>1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected. |
| 4:0<br>ADCH | Input Channel Select. The ADCH bits form a 5-bit field that selects one of the input channels. The input channels are detailed in Table 10-4.<br>The successive approximation converter subsystem is turned off when the channel select bits are all set. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes. |

**Table 10-4. Input Channel Select**

| ADCH | Input Select |
|---|---|
| 00000–01111 | AD0–15 |
| 10000–11011 | AD16–27 |
| 11100 | Reserved |
| 11101 | $V_{REFH}$ |
| 11110 | $V_{REFL}$ |
| 11111 | Module disabled |

## 10.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register controls the compare function, conversion trigger, and conversion active of the ADC module.

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

**Figure 10-4. Status and Control Register 2 (ADCSC2)**

**Table 10-5. ADCSC2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADACT | Conversion Active. Indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted.<br>0 Conversion not in progress<br>1 Conversion in progress |
| 6<br>ADTRG | Conversion Trigger Select. Selects the type of trigger used for initiating a conversion. Two types of triggers are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input.<br>0 Software trigger selected<br>1 Hardware trigger selected |
| 5<br>ACFE | Compare Function Enable. Enables the compare function.<br>0 Compare function disabled<br>1 Compare function enabled |
| 4<br>ACFGT | Compare Function Greater Than Enable. Configures the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value.<br>0 Compare triggers when input is less than compare value<br>1 Compare triggers when input is greater than or equal to compare value |

## 10.3.3 Data Result High Register (ADCRH)

In 12-bit operation, ADCRH contains the upper four bits of the result of a 12-bit conversion. In 10-bit mode, ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 10-bit mode, ADR[11:10] are cleared. When configured for 8-bit mode, ADR[11:8] are cleared.

In 12-bit and 10-bit mode, ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. When a compare event does occur, the value is the addition of the conversion result and the two’s complement of the compare value. In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, the intermediate conversion result is lost. In 8-bit mode, there is no interlocking with ADCRL.

If the MODE bits are changed, any data in ADCRH becomes invalid.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | ADR11 | ADR10 | ADR9 | ADR8 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-5. Data Result High Register (ADCRH)**

## 10.3.4 Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of the result of a 12-bit or 10-bit conversion, and all eight bits of an 8-bit conversion. This register is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. When a compare event does occur, the value is the addition of the conversion result and the two's complement of the compare value. In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until the after next conversion is completed, the intermediate conversion results are lost. In 8-bit mode, there is no interlocking with ADCRH. If the MODE bits are changed, any data in ADCRL becomes invalid.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-6. Data Result Low Register (ADCRL)**

## 10.3.5 Compare Value High Register (ADCCVH)

In 12-bit mode, the ADCCVH register holds the upper four bits of the 12-bit compare value. When the compare function is enabled, these bits are compared to the upper four bits of the result following a conversion in 12-bit mode.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | ADCV11 | ADCV10 | ADCV9 | ADCV8 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-7. Compare Value High Register (ADCCVH)**

In 10-bit mode, the ADCCVH register holds the upper two bits of the 10-bit compare value (ADCV[9:8]). These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.

In 8-bit mode, ADCCVH is not used during compare.

## 10.3.6 Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 12-bit or 10-bit compare value or all 8 bits of the 8-bit compare value. When the compare function is enabled, bits ADCV[7:0] are compared to the lower 8 bits of the result following a conversion in 12-bit, 10-bit or 8-bit mode.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ADCV7 | ADCV6 | ADCV5 | ADCV4 | ADCV3 | ADCV2 | ADCV1 | ADCV0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-8. Compare Value Low Register (ADCCVL)**

## 10.3.7 Configuration Register (ADCCFG)

ADCCFG selects the mode of operation, clock source, clock divide, and configures for low power and long sample time.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ADLPC | ADIV | | ADLSMP | MODE | | ADICLK | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-9. Configuration Register (ADCCFG)**

**Table 10-6. ADCCFG Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADLPC | Low-Power Configuration. ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required.<br>0 High speed configuration<br>1 Low power configuration: The power is reduced at the expense of maximum clock speed. |
| 6:5<br>ADIV | Clock Divide Select. ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 10-7 shows the available clock configurations. |
| 4<br>ADLSMP | Long Sample Time Configuration. ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required.<br>0 Short sample time<br>1 Long sample time |

**Table 10-6. ADCCFG Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3:2<br>MODE | Conversion Mode Selection. MODE bits are used to select between 12-, 10-, or 8-bit operation. See Table 10-8. |
| 1:0<br>ADICLK | Input Clock Select. ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 10-9. |

**Table 10-7. Clock Divide Select**

| ADIV | Divide Ratio | Clock Rate |
|---|---|---|
| 00 | 1 | Input clock |
| 01 | 2 | Input clock ÷ 2 |
| 10 | 4 | Input clock ÷ 4 |
| 11 | 8 | Input clock ÷ 8 |

**Table 10-8. Conversion Modes**

| MODE | Mode Description |
|---|---|
| 00 | 8-bit conversion (N=8) |
| 01 | 12-bit conversion (N=12) |
| 10 | 10-bit conversion (N=10) |
| 11 | Reserved |

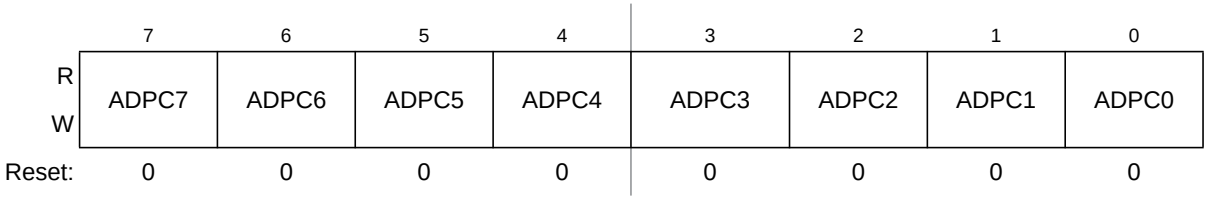
**Table 10-9. Input Clock Select**

| ADICLK | Selected Clock Source |
|---|---|
| 00 | Bus clock |
| 01 | Bus clock divided by 2 |
| 10 | Alternate clock (ALTCLK) |
| 11 | Asynchronous clock (ADACK) |

## 10.3.8 Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 is

used to control the pins associated with channels 0–7 of the ADC module.



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ADPC7 | ADPC6 | ADPC5 | ADPC4 | ADPC3 | ADPC2 | ADPC1 | ADPC0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

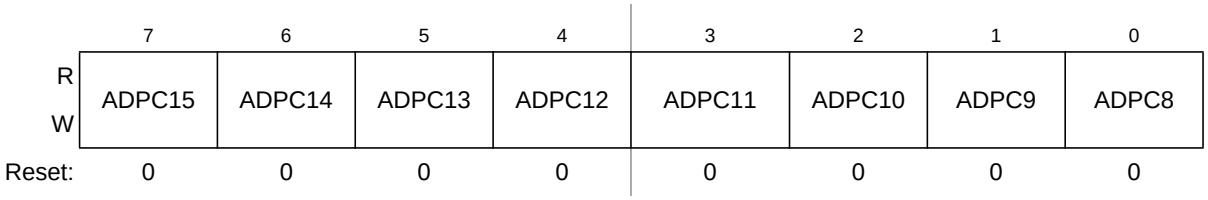
**Figure 10-10. Pin Control 1 Register (APCTL1)**

**Table 10-10. APCTL1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC7 | ADC Pin Control 7. ADPC7 controls the pin associated with channel AD7.<br>0 AD7 pin I/O control enabled<br>1 AD7 pin I/O control disabled |
| 6<br>ADPC6 | ADC Pin Control 6. ADPC6 controls the pin associated with channel AD6.<br>0 AD6 pin I/O control enabled<br>1 AD6 pin I/O control disabled |
| 5<br>ADPC5 | ADC Pin Control 5. ADPC5 controls the pin associated with channel AD5.<br>0 AD5 pin I/O control enabled<br>1 AD5 pin I/O control disabled |
| 4<br>ADPC4 | ADC Pin Control 4. ADPC4 controls the pin associated with channel AD4.<br>0 AD4 pin I/O control enabled<br>1 AD4 pin I/O control disabled |
| 3<br>ADPC3 | ADC Pin Control 3. ADPC3 controls the pin associated with channel AD3.<br>0 AD3 pin I/O control enabled<br>1 AD3 pin I/O control disabled |
| 2<br>ADPC2 | ADC Pin Control 2. ADPC2 controls the pin associated with channel AD2.<br>0 AD2 pin I/O control enabled<br>1 AD2 pin I/O control disabled |
| 1<br>ADPC1 | ADC Pin Control 1. ADPC1 controls the pin associated with channel AD1.<br>0 AD1 pin I/O control enabled<br>1 AD1 pin I/O control disabled |
| 0<br>ADPC0 | ADC Pin Control 0. ADPC0 controls the pin associated with channel AD0.<br>0 AD0 pin I/O control enabled<br>1 AD0 pin I/O control disabled |

## 10.3.9 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ADPC15 | ADPC14 | ADPC13 | ADPC12 | ADPC11 | ADPC10 | ADPC9 | ADPC8 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-11. Pin Control 2 Register (APCTL2)**

**Table 10-11. APCTL2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC15 | ADC Pin Control 15. ADPC15 controls the pin associated with channel AD15.<br>0 AD15 pin I/O control enabled<br>1 AD15 pin I/O control disabled |
| 6<br>ADPC14 | ADC Pin Control 14. ADPC14 controls the pin associated with channel AD14.<br>0 AD14 pin I/O control enabled<br>1 AD14 pin I/O control disabled |
| 5<br>ADPC13 | ADC Pin Control 13. ADPC13 controls the pin associated with channel AD13.<br>0 AD13 pin I/O control enabled<br>1 AD13 pin I/O control disabled |
| 4<br>ADPC12 | ADC Pin Control 12. ADPC12 controls the pin associated with channel AD12.<br>0 AD12 pin I/O control enabled<br>1 AD12 pin I/O control disabled |
| 3<br>ADPC11 | ADC Pin Control 11. ADPC11 controls the pin associated with channel AD11.<br>0 AD11 pin I/O control enabled<br>1 AD11 pin I/O control disabled |
| 2<br>ADPC10 | ADC Pin Control 10. ADPC10 controls the pin associated with channel AD10.<br>0 AD10 pin I/O control enabled<br>1 AD10 pin I/O control disabled |
| 1<br>ADPC9 | ADC Pin Control 9. ADPC9 controls the pin associated with channel AD9.<br>0 AD9 pin I/O control enabled<br>1 AD9 pin I/O control disabled |
| 0<br>ADPC8 | ADC Pin Control 8. ADPC8 controls the pin associated with channel AD8.<br>0 AD8 pin I/O control enabled<br>1 AD8 pin I/O control disabled |

## 10.3.10 Pin Control 3 Register (APCTL3)

APCTL3 controls channels 16–23 of the ADC module.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ADPC23 | ADPC22 | ADPC21 | ADPC20 | ADPC19 | ADPC18 | ADPC17 | ADPC16 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-12. Pin Control 3 Register (APCTL3)**

**Table 10-12. APCTL3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC23 | ADC Pin Control 23. ADPC23 controls the pin associated with channel AD23.<br>0 AD23 pin I/O control enabled<br>1 AD23 pin I/O control disabled |
| 6<br>ADPC22 | ADC Pin Control 22. ADPC22 controls the pin associated with channel AD22.<br>0 AD22 pin I/O control enabled<br>1 AD22 pin I/O control disabled |
| 5<br>ADPC21 | ADC Pin Control 21. ADPC21 controls the pin associated with channel AD21.<br>0 AD21 pin I/O control enabled<br>1 AD21 pin I/O control disabled |
| 4<br>ADPC20 | ADC Pin Control 20. ADPC20 controls the pin associated with channel AD20.<br>0 AD20 pin I/O control enabled<br>1 AD20 pin I/O control disabled |
| 3<br>ADPC19 | ADC Pin Control 19. ADPC19 controls the pin associated with channel AD19.<br>0 AD19 pin I/O control enabled<br>1 AD19 pin I/O control disabled |
| 2<br>ADPC18 | ADC Pin Control 18. ADPC18 controls the pin associated with channel AD18.<br>0 AD18 pin I/O control enabled<br>1 AD18 pin I/O control disabled |
| 1<br>ADPC17 | ADC Pin Control 17. ADPC17 controls the pin associated with channel AD17.<br>0 AD17 pin I/O control enabled<br>1 AD17 pin I/O control disabled |
| 0<br>ADPC16 | ADC Pin Control 16. ADPC16 controls the pin associated with channel AD16.<br>0 AD16 pin I/O control enabled<br>1 AD16 pin I/O control disabled |

## 10.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. In 12-bit and 10-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 12-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). In 10-bit mode, the result is rounded to 10 bits and placed in the data registers (ADCRH and ADCRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ACFE bit and operates with any of the conversion modes and configurations.

### 10.4.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by two. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. When selected as the clock source, this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC do not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

### 10.4.2 Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) disable the I/O port control of the pins used as analog inputs.When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

### 10.4.3 Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

### 10.4.4 Conversion Control

Conversions can be performed in 12-bit mode, 10-bit mode, or 8-bit mode as determined by the MODE bits. Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be

configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

#### 10.4.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

#### 10.4.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRH and ADCRL. This is indicated by the setting of COCO. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 12-bit or 10-bit MODE (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

#### 10.4.4.3 Aborting Conversions

Any conversion in progress is aborted when:

- A write to ADCSC1 occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCSC2, ADCCFG, ADCCVH, or ADCCVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL, are not altered. However, they continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset, ADCRH and ADCRL return to their reset states.

#### 10.4.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADLPC. This results in a lower maximum value for $f_{ADCK}$ (see the electrical specifications).

#### 10.4.4.5 Sample Time and Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock ($f_{ADCK}$). After the module becomes active, sampling of the input begins. ADLSMP selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times.When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

If the bus frequency is less than the $f_{ADCK}$ frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). If the bus frequency is less than 1/11th of the $f_{ADCK}$ frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in Table 10-13.

**Table 10-13. Total Conversion Time vs. Control Conditions**

| Conversion Type | ADICLK | ADLSMP | Max Total Conversion Time |
|---|---|---|---|
| Single or first continuous 8-bit | 0x, 10 | 0 | 20 ADCK cycles + 5 bus clock cycles |
| Single or first continuous 10-bit or 12-bit | 0x, 10 | 0 | 23 ADCK cycles + 5 bus clock cycles |
| Single or first continuous 8-bit | 0x, 10 | 1 | 40 ADCK cycles + 5 bus clock cycles |
| Single or first continuous 10-bit or 12-bit | 0x, 10 | 1 | 43 ADCK cycles + 5 bus clock cycles |
| Single or first continuous 8-bit | 11 | 0 | 5 µs + 20 ADCK + 5 bus clock cycles |
| Single or first continuous 10-bit or 12-bit | 11 | 0 | 5 µs + 23 ADCK + 5 bus clock cycles |
| Single or first continuous 8-bit | 11 | 1 | 5 µs + 40 ADCK + 5 bus clock cycles |
| Single or first continuous 10-bit or 12-bit | 11 | 1 | 5 µs + 43 ADCK + 5 bus clock cycles |
| Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$ | xx | 0 | 17 ADCK cycles |
| Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$ | xx | 0 | 20 ADCK cycles |
| Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$ | xx | 1 | 37 ADCK cycles |

**Table 10-13. Total Conversion Time vs. Control Conditions**

| Conversion Type | ADICLK | ADLSMP | Max Total Conversion Time |
|---|---|---|---|
| Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$ | xx | 1 | 40 ADCK cycles |

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. For example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion is:

$$\text{Conversion time} = \frac{\text{23 ADCK Cyc}}{\text{8 MHz/1}} + \frac{\text{5 bus Cyc}}{\text{8 MHz}} = 3.5 \text{ ms}$$

Number of bus cycles = 3.5 ms x 8 MHz = 28 cycles

**NOTE**

The ADCK frequency must be between $f_{ADCK}$ minimum and $f_{ADCK}$ maximum to meet ADC specifications.

## 10.4.5 Automatic Compare Function

The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the result is added to the two's complement of the compare value (ADCCVH and ADCCVL). When comparing to an upper limit (ACFGT = 1), if the result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. The value generated by the addition of the conversion result and the two's complement of the compare value is transferred to ADCRH and ADCRL.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

**NOTE**

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

## 10.4.6 MCU Wait Mode Operation

Wait mode is a lower power-consumption standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of

ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (AIEN = 1).

## 10.4.7 MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

### 10.4.7.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

### 10.4.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

**NOTE**

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the data transfer blocking mechanism (discussed in Section 10.4.4.2, "Completing Conversions") is cleared when entering stop3 and continuing ADC conversions.

## 10.4.8 MCU Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters stop2 mode. All module registers contain their reset values following exit from stop2. Therefore, the module must be re-enabled and re-configured following exit from stop2.

## 10.5 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 10-7, Table 10-8, and Table 10-9 for information used in this example.

**NOTE**

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

### 10.5.1 ADC Module Initialization Example

#### 10.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

#### 10.5.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

**ADCCFG = 0x98 (%10011000)**

```
Bit 7    ADLPC    1        Configures for low power (lowers maximum clock speed)
Bit 6:5  ADIV     00       Sets the ADCK to the input clock ÷ 1
Bit 4    ADLSMP   1        Configures for long sample time
Bit 3:2  MODE     10       Sets mode at 10-bit conversions
Bit 1:0  ADICLK   00       Selects bus clock as input clock source
```

**ADCSC2 = 0x00 (%00000000)**

```
Bit 7    ADACT    0        Flag indicates if a conversion is in progress
Bit 6    ADTRG    0        Software trigger selected
Bit 5    ACFE     0        Compare function disabled
Bit 4    ACFGT    0        Not used in this example
Bit 3:2           00       Reserved, always reads zero
Bit 1:0           00       Reserved for Freescale's internal use; always write zero
```

### ADCSC1 = 0x41 (%01000001)

```
Bit 7    COCO     0        Read-only flag which is set when a conversion completes
Bit 6    AIEN     1        Conversion complete interrupt enabled
Bit 5    ADCO     0        One conversion only (continuous conversions disabled)
Bit 4:0  ADCH     00001    Input channel 1 selected as ADC input channel
```

### ADCRH/L = 0xxx

Holds results of conversion. Read high byte (ADCRH) before low byte (ADCRL) so that conversion data cannot be overwritten with data from the next conversion.

### ADCCVH/L = 0xxx

Holds compare value when compare function enabled

### APCTL1=0x02

AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins

### APCTL2=0x00

All other AD pins remain general purpose I/O pins

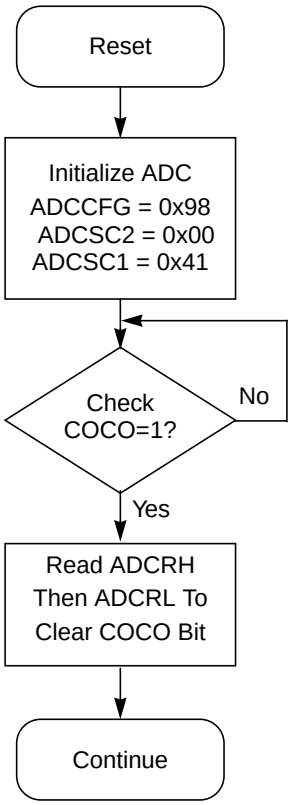



**Figure 10-13. Initialization Flowchart for Example**

## 10.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

### 10.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

#### 10.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies ($V_{DDAD}$ and $V_{SSAD}$) available as separate pins on some devices. $V_{SSAD}$ is shared on the same pin as the MCU digital $V_{SS}$ on some devices. On other devices, $V_{SSAD}$ and $V_{DDAD}$ are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both $V_{DDAD}$ and $V_{SSAD}$ must be connected to the same voltage potential as their corresponding MCU digital supply ($V_{DD}$ and $V_{SS}$) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the $V_{SSAD}$ pin. This should be the only ground connection between these supplies if possible. The $V_{SSAD}$ pin makes a good single point ground location.

#### 10.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is $V_{REFH}$, which may be shared on the same pin as $V_{DDAD}$ on some devices. The low reference is $V_{REFL}$, which may be shared on the same pin as $V_{SSAD}$ on some devices.

When available on a separate pin, $V_{REFH}$ may be connected to the same potential as $V_{DDAD}$, or may be driven by an external source between the minimum $V_{DDAD}$ spec and the $V_{DDAD}$ potential ($V_{REFH}$ must never exceed $V_{DDAD}$). When available on a separate pin, $V_{REFL}$ must be connected to the same voltage potential as $V_{SSAD}$. $V_{REFH}$ and $V_{REFL}$ must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the $V_{REFH}$ and $V_{REFL}$ loop. The best external component to meet this current demand is a 0.1 µF capacitor with good high frequency characteristics. This capacitor is connected between $V_{REFH}$ and $V_{REFL}$ and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum (parasitic only).

#### 10.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at $V_{DD}$ or $V_{SS}$. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 µF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to $V_{SSA}$.

For proper conversion, the input voltage must fall between $V_{REFH}$ and $V_{REFL}$. If the input is equal to or exceeds $V_{REFH}$, the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than $V_{REFL}$, the converter circuit converts it to 0x000. Input voltages between $V_{REFH}$ and $V_{REFL}$ are straight-line linear conversions. There is a brief current associated with $V_{REFL}$ when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

### 10.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

#### 10.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source ($R_{AS}$) is kept below 2 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

#### 10.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance ($R_{AS}$) is high. If this error cannot be tolerated by the application, keep $R_{AS}$ lower than $V_{DDAD} / (2^N * I_{LEAK})$ for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

### 10.6.2.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 µF low-ESR capacitor from $V_{REFH}$ to $V_{REFL}$.
- There is a 0.1 µF low-ESR capacitor from $V_{DDAD}$ to $V_{SSAD}$.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from $V_{DDAD}$ to $V_{SSAD}$.
- $V_{SSAD}$ (and $V_{REFL}$, if connected) is connected to $V_{SS}$ at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces $V_{DD}$ noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive $V_{DD}$ noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor ($C_{AS}$) on the selected input channel to $V_{REFL}$ or $V_{SSAD}$ (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

### 10.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\ lsb = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 10-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is –1 lsb to 0 lsb and the code width of each step is 1 lsb.

### 10.6.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error ($E_{ZS}$) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error ($E_{FS}$) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

### 10.6.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around 1/2lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 10.6.2.3 reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

# Chapter 11
# Inter-Integrated Circuit (S08IICV2)

## 11.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. 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.

All MC9S08DZ60 Series MCUs feature the IIC, as shown in the following block diagram.

**NOTE**

Drive strength must be disabled (DSE=0) for the IIC pins when using the IIC module for correct operation.

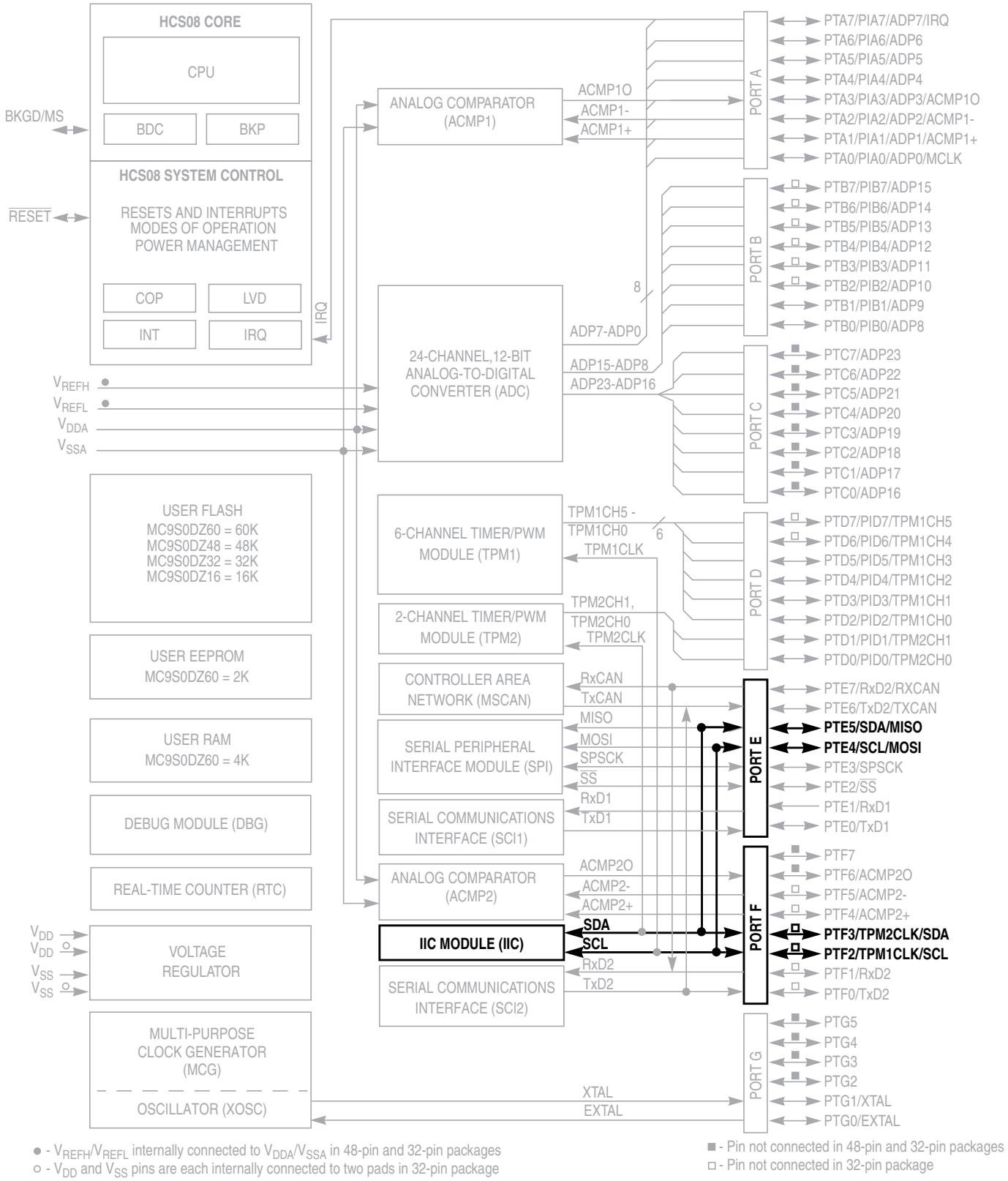



**Figure 11-1. MC9S08DZ60 Block Diagram**

### 11.1.1 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 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 recognition
- 10-bit address extension

### 11.1.2 Modes of Operation

A brief description of the IIC in the various MCU modes is given here.

- **Run mode** — This is the basic mode of operation. To conserve power in this mode, disable the module.
- **Wait mode** — The module continues to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- **Stop mode** — The IIC is inactive in stop3 mode for reduced power consumption. The stop instruction does not affect IIC register states. Stop2 resets the register contents.

### 11.1.3 Block Diagram

Figure 11-2 is a block diagram of the IIC.

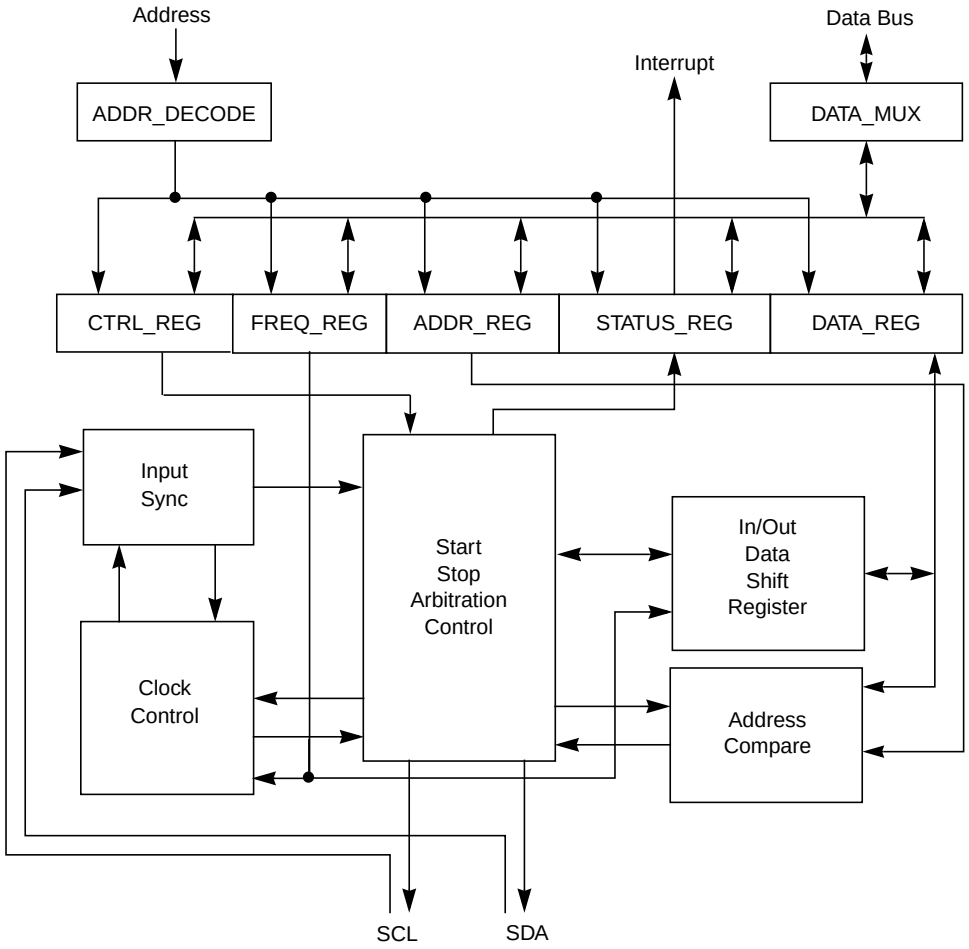



**Figure 11-2. IIC Functional Block Diagram**

## 11.2 External Signal Description

This section describes each user-accessible pin signal.

### 11.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

### 11.2.2 SDA — Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

## 11.3 Register Definition

This section consists of the IIC register descriptions in address order.

Refer to the direct-page register summary in the memory chapter of this document for the absolute address assignments for all IIC registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

## 11.3.1 IIC Address Register (IICA)

**Figure 11-3. IIC Address Register (IICA)**

**Table 11-1. IICA Field Descriptions**

| Field | Description |
|---|---|
| 7–1<br>AD[7:1] | **Slave Address.** The AD[7:1] field contains the slave address to be used by the IIC module. This field is used on the 7-bit address scheme and the lower seven bits of the 10-bit address scheme. |

## 11.3.2 IIC Frequency Divider Register (IICF)

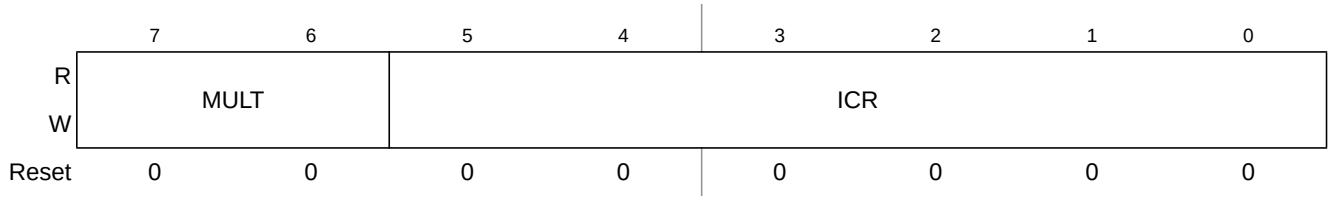



**Figure 11-4. IIC Frequency Divider Register (IICF)**

**Table 11-2. IICF Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>MULT | **IIC Multiplier Factor**. The MULT bits define the multiplier factor, mul. This factor, along with the SCL divider, generates the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below.<br>00 mul = 01<br>01 mul = 02<br>10 mul = 04<br>11 Reserved |
| 5–0<br>ICR | **IIC Clock Rate**. The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits determine the IIC baud rate, the SDA hold time, the SCL Start hold time, and the SCL Stop hold time. Table 11-4 provides the SCL divider and hold values for corresponding values of the ICR.<br><br>The SCL divider multiplied by multiplier factor mul generates IIC baud rate.<br><br>$$\text{IIC baud rate} = \frac{\text{bus speed (Hz)}}{\text{mul} \times \text{SCLdivider}}$$ ***Eqn. 11-1***<br><br>SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data).<br><br>$$\text{SDA hold time} = \text{bus period (s)} \times \text{mul} \times \text{SDA hold value}$$ Eqn. 11-2<br><br>SCL start hold time is the delay from the falling edge of SDA (IIC data) while SCL is high (Start condition) to the falling edge of SCL (IIC clock).<br><br>$$\text{SCL Start hold time} = \text{bus period (s)} \times \text{mul} \times \text{SCL Start hold value}$$ Eqn. 11-3<br><br>SCL stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA<br>SDA (IIC data) while SCL is high (Stop condition).<br><br>$$\text{SCL Stop hold time} = \text{bus period (s)} \times \text{mul} \times \text{SCL Stop hold value}$$ Eqn. 11-4 |

For example, if the bus speed is 8 MHz, the table below shows the possible hold time values with different ICR and MULT selections to achieve an IIC baud rate of 100kbps.

**Table 11-3. Hold Time Values for 8 MHz Bus Speed**

| MULT | ICR | Hold Times (μs) | | |
|---|---|---|---|---|
| | | SDA | SCL Start | SCL Stop |
| 0x2 | 0x00 | 3.500 | 3.000 | 5.500 |
| 0x1 | 0x07 | 2.500 | 4.000 | 5.250 |
| 0x1 | 0x0B | 2.250 | 4.000 | 5.250 |
| 0x0 | 0x14 | 2.125 | 4.250 | 5.125 |
| 0x0 | 0x18 | 1.125 | 4.750 | 5.125 |

**Table 11-4. IIC Divider and Hold Values**

| ICR (hex) | SCL Divider | SDA Hold Value | SCL Hold (Start) Value | SDA Hold (Stop) Value |
|---|---|---|---|---|
| **00** | 20 | 7 | 6 | 11 |
| **01** | 22 | 7 | 7 | 12 |
| **02** | 24 | 8 | 8 | 13 |
| **03** | 26 | 8 | 9 | 14 |
| **04** | 28 | 9 | 10 | 15 |
| **05** | 30 | 9 | 11 | 16 |
| **06** | 34 | 10 | 13 | 18 |
| **07** | 40 | 10 | 16 | 21 |
| **08** | 28 | 7 | 10 | 15 |
| **09** | 32 | 7 | 12 | 17 |
| **0A** | 36 | 9 | 14 | 19 |
| **0B** | 40 | 9 | 16 | 21 |
| **0C** | 44 | 11 | 18 | 23 |
| **0D** | 48 | 11 | 20 | 25 |
| **0E** | 56 | 13 | 24 | 29 |
| **0F** | 68 | 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 |

| ICR (hex) | SCL Divider | SDA Hold Value | SCL Hold (Start) Value | SCL Hold (Stop) Value |
|---|---|---|---|---|
| **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 |
| **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 |

## 11.3.3 IIC Control Register (IICC1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IICEN | IICIE | MST | TX | TXAK | 0 | 0 | 0 |
| W | | | | | | RSTA | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-5. IIC Control Register (IICC1)**

**Table 11-5. IICC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IICEN | **IIC Enable.** The IICEN bit determines whether the IIC module is enabled.<br>0 IIC is not enabled<br>1 IIC is enabled |
| 6<br>IICIE | **IIC Interrupt Enable.** The IICIE bit determines whether an IIC interrupt is requested.<br>0 IIC interrupt request not enabled<br>1 IIC interrupt request enabled |
| 5<br>MST | **Master Mode Select.** The MST bit changes from a 0 to a 1 when a start signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a stop signal is generated and the mode of operation changes from master to slave.<br>0 Slave mode<br>1 Master mode |
| 4<br>TX | **Transmit Mode Select.** The TX bit selects the direction of master and slave transfers. In master mode, this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. When addressed as a slave, this bit should be set by software according to the SRW bit in the status register.<br>0 Receive<br>1 Transmit |
| 3<br>TXAK | **Transmit Acknowledge Enable.** This bit specifies the value driven onto the SDA during data acknowledge cycles for master and slave receivers.<br>0 An acknowledge signal is sent out to the bus after receiving one data byte<br>1 No acknowledge signal response is sent |
| 2<br>RSTA | **Repeat start.** Writing a 1 to this bit generates a repeated start condition provided it is the current master. This bit is always read as cleared. Attempting a repeat at the wrong time results in loss of arbitration. |

## 11.3.4 IIC Status Register (IICS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-6. IIC Status Register (IICS)**

**Table 11-6. IICS Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TCF | **Transfer Complete Flag.** This bit is set on the completion of a byte transfer. This bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IICD register in receive mode or writing to the IICD in transmit mode.<br>0 Transfer in progress<br>1 Transfer complete |
| 6<br>IAAS | **Addressed as a Slave.** The IAAS bit is set when the calling address matches the programmed slave address or when the GCAEN bit is set and a general call is received. Writing the IICC register clears this bit.<br>0 Not addressed<br>1 Addressed as a slave |
| 5<br>BUSY | **Bus Busy.** The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a start signal is detected and cleared when a stop signal is detected.<br>0 Bus is idle<br>1 Bus is busy |
| 4<br>ARBL | **Arbitration Lost.** This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software by writing a 1 to it.<br>0 Standard bus operation<br>1 Loss of arbitration |
| 2<br>SRW | **Slave Read/Write.** When addressed as a slave, the SRW bit indicates the value of the R/W command bit of the calling address sent to the master.<br>0 Slave receive, master writing to slave<br>1 Slave transmit, master reading from slave |
| 1<br>IICIF | **IIC Interrupt Flag.** The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit:<br>• One byte transfer completes<br>• Match of slave address to calling address<br>• Arbitration lost<br>0 No interrupt pending<br>1 Interrupt pending |
| 0<br>RXAK | **Receive Acknowledge**. When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected.<br>0 Acknowledge received<br>1 No acknowledge received |

## 11.3.5 IIC Data I/O Register (IICD)

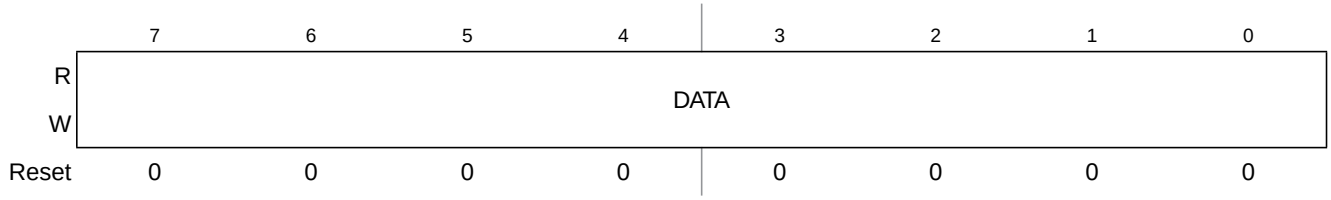



**Figure 11-7. IIC Data I/O Register (IICD)**

**Table 11-7. IICD Field Descriptions**

| Field | Description |
|---|---|
| 7–0<br>DATA | **Data** — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data. |

**NOTE**

When transitioning out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

The TX bit in IICC 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, reading the IICD does not initiate the receive.

Reading the IICD returns the last byte received while the IIC is configured in master receive or slave receive modes. The IICD does not reflect every byte transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

## 11.3.6 IIC Control Register 2 (IICC2)

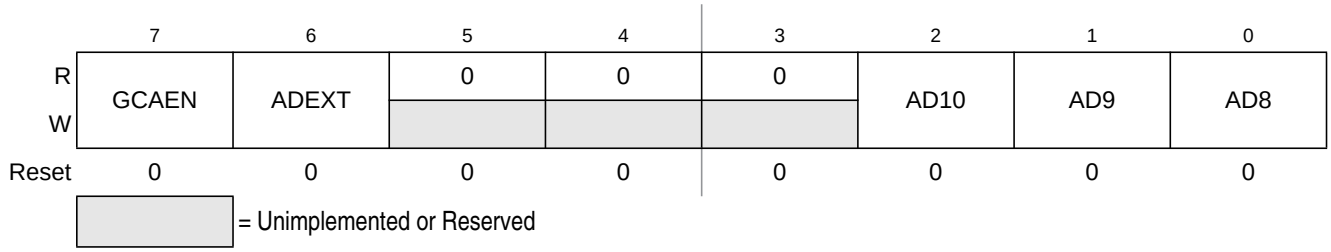



**Figure 11-8. IIC Control Register (IICC2)**

**Table 11-8. IICC2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>GCAEN | **General Call Address Enable.** The GCAEN bit enables or disables general call address.<br>0 General call address is disabled<br>1 General call address is enabled |
| 6<br>ADEXT | **Address Extension.** The ADEXT bit controls the number of bits used for the slave address.<br>0 7-bit address scheme<br>1 10-bit address scheme |
| 2–0<br>AD[10:8] | Slave Address. The AD[10:8] field contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set. |

## 11.4 Functional Description

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

### 11.4.1 IIC 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. A 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
- Stop signal

The stop signal should not be confused with the CPU stop instruction. The IIC bus system communication is described briefly in the following sections and illustrated in Figure 11-9.

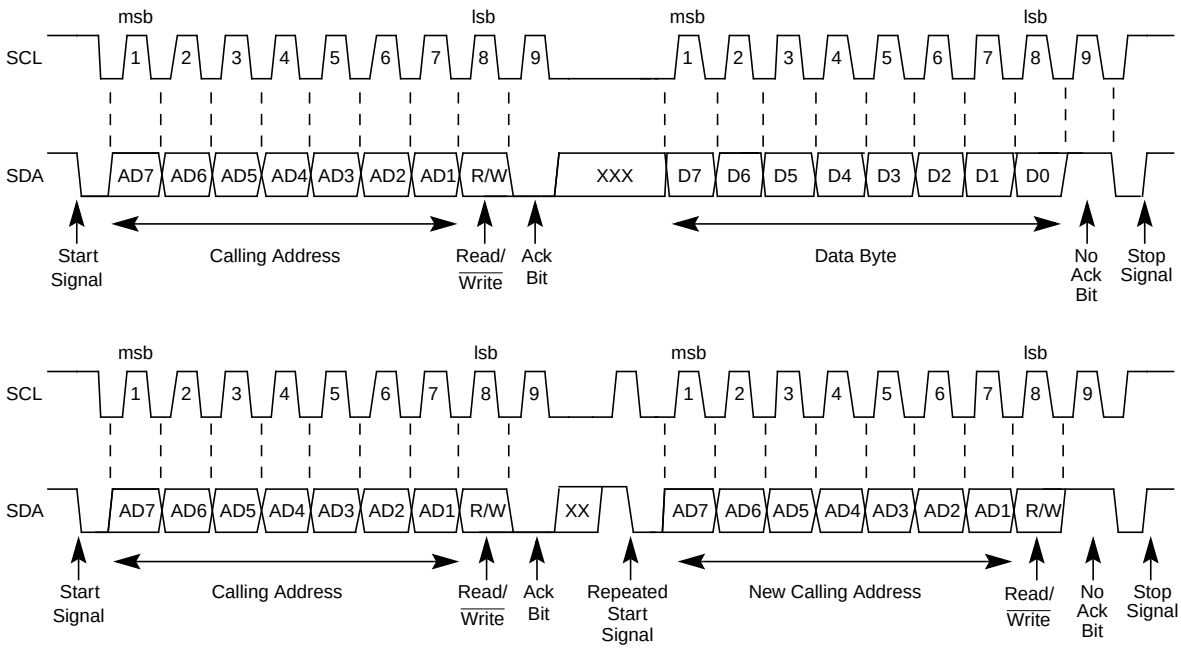



**Figure 11-9. IIC Bus Transmission Signals**

### 11.4.1.1 Start Signal

When the bus is free, no master device is engaging the bus (SCL and SDA lines are at logical high), a master may initiate communication by sending a start signal. As shown in Figure 11-9, 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.

### 11.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the start signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ 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.

Only the slave with a calling address that matches the one transmitted by the master responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 11-9).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC reverts to slave mode and operates correctly even if it is being addressed by another master.

#### 11.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{W}$ 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 11-9. There is one clock pulse on SCL for each data bit, the msb being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a stop signal.
- Commences a new calling by generating a repeated start signal.

#### 11.4.1.4 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 11-9).

The master can generate a stop even if the slave has generated an acknowledge at which point the slave must release the bus.

#### 11.4.1.5 Repeated Start Signal

As shown in Figure 11-9, 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.

#### 11.4.1.6 Arbitration Procedure

The IIC 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.

### 11.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after 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 still 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 11-10). 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.

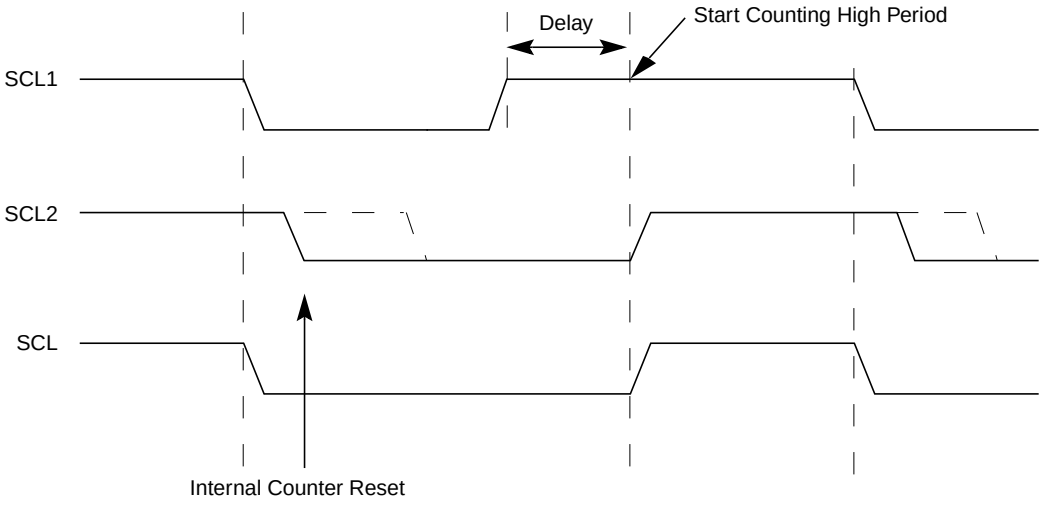



**Figure 11-10. IIC Clock Synchronization**

### 11.4.1.8 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 a case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

### 11.4.1.9 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.

## 11.4.2 10-bit Address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

### 11.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 11-9). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

| S | Slave Address 1st 7 bits<br>11110 + AD10 + AD9 | R/W<br>0 | A1 | Slave Address 2nd byte<br>AD[8:1] | A2 | Data | A | ... | Data | A/A | P |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Table 11-9. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address**

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

### 11.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit (see Table 11-10). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

After a repeated start condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices) or the 11110XX slave address (for 7-bit devices) does not match.

| S | Slave Address<br>1st 7 bits<br>11110 + AD10 + AD9 | R/W<br>0 | A1 | Slave Address<br>2nd byte<br>AD[8:1] | A2 | Sr | Slave Address<br>1st 7 bits<br>11110 + AD10 + AD9 | R/W<br>1 | A3 | Data | A | ... | Data | A | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Table 11-10. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address**

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

### 11.4.3 General Call Address

General calls can be requested in 7-bit address or 10-bit address. If the GCAEN bit is set, the IIC matches the general call address as well as its own slave address. When the IIC responds to a general call, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the IICD register after the first byte transfer to determine whether the address matches is its own slave address or a general call. If the value is 00, the match is a general call. If the GCAEN bit is clear, the IIC ignores any data supplied from a general call address by not issuing an acknowledgement.

## 11.5 Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

## 11.6 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 11-11 occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. You can determine the interrupt type by reading the status register.

**Table 11-11. Interrupt Summary**

| Interrupt Source | Status | Flag | Local Enable |
|---|---|---|---|
| Complete 1-byte transfer | TCF | IICIF | IICIE |
| Match of received calling address | IAAS | IICIF | IICIE |
| Arbitration Lost | ARBL | IICIF | IICIE |

### 11.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the ninth clock to indicate the completion of byte transfer.

### 11.6.2 Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the status register is set. The CPU is interrupted, provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

### 11.6.3 Arbitration Lost Interrupt

The IIC 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, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A start cycle is attempted when the bus is busy.
- A repeated start cycle is requested in slave mode.
- A stop condition is detected when the master did not request it.

This bit must be cleared by software writing a 1 to it.

# 11.7 Initialization/Application Information

**Module Initialization (Slave)**

1. Write: IICC2
   — to enable or disable general call
   — to select 10-bit or 7-bit addressing mode
2. Write: IICA
   — to set the slave address
3. Write: IICC1
   — to enable IIC and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in Figure 11-12

**Module Initialization (Master)**

1. Write: IICF
   — to set the IIC baud rate (example provided in this chapter)
2. Write: IICC1
   — to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in Figure 11-12
5. Write: IICC1
   — to enable TX
6. Write: IICC1
   — to enable MST (master mode)
7. Write: IICD
   — with the address of the target slave. (The lsb of this byte determines whether the communication is master receive or transmit.)

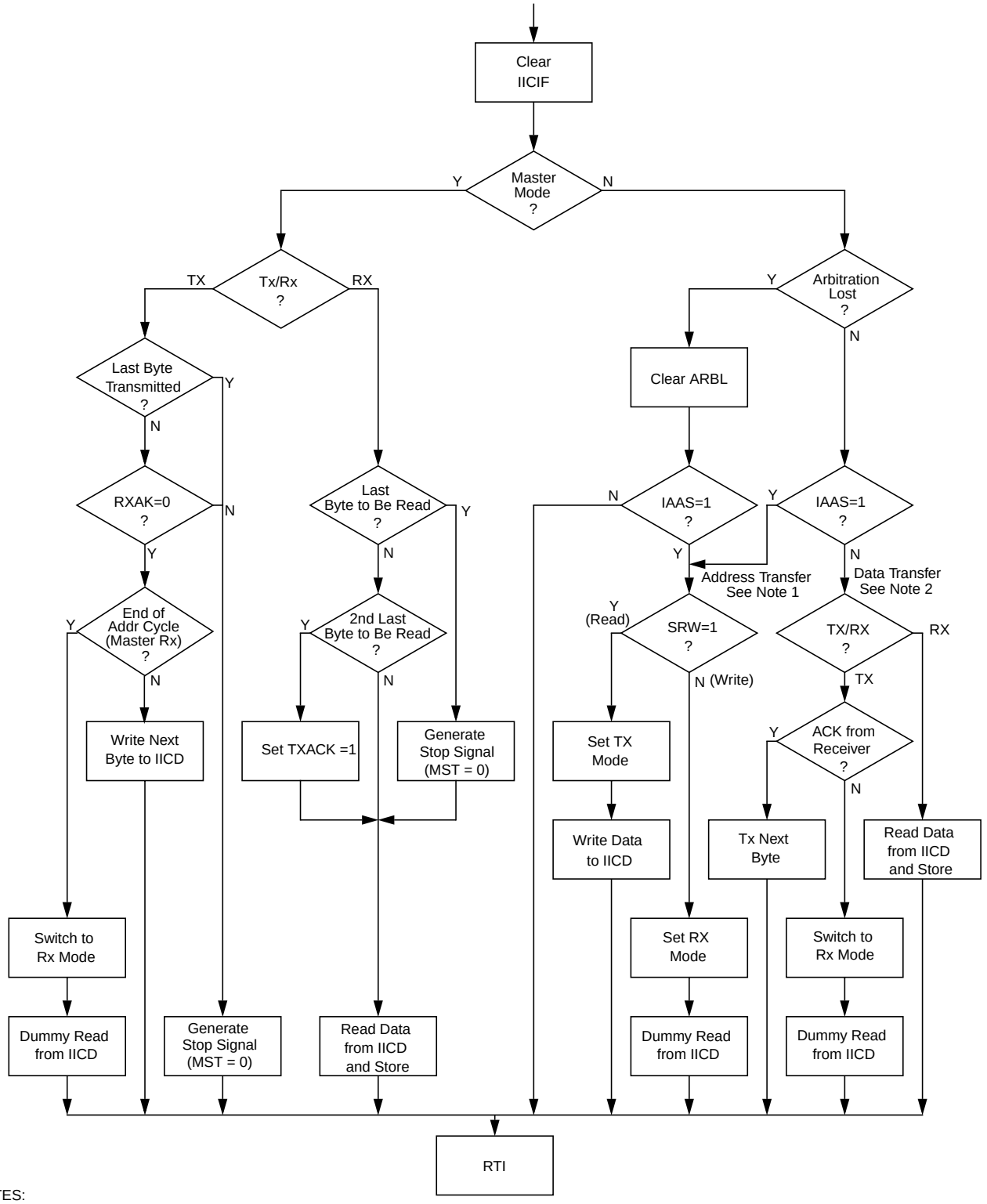
**Module Use**

The routine shown in Figure 11-12 can handle both master and slave IIC operations. For slave operation, an incoming IIC message that contains the proper address begins IIC communication. For master operation, communication must be initiated by writing to the IICD register.

**Register Model**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IICA | AD[7:1] | | | | | | | 0 |
| | When addressed as a slave (in slave mode), the module responds to this address | | | | | | | |
| IICF | MULT | | ICR | | | | | |
| | Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER)) | | | | | | | |
| IICC1 | IICEN | IICIE | MST | TX | TXAK | RSTA | 0 | 0 |
| | Module configuration | | | | | | | |
| IICS | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| | Module status flags | | | | | | | |
| IICD | DATA | | | | | | | |
| | Data register; Write to transmit IIC data read to read IIC data | | | | | | | |
| IICC2 | GCAEN | ADEXT | 0 | 0 | 0 | AD10 | AD9 | AD8 |
| | Address configuration | | | | | | | |

**Figure 11-11. IIC Module Quick Start**




NOTES:
1. If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.
2. When 10-bit addressing is used to address a slave, the slave sees an interrupt following the first byte of the extended address. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as a valid data transfer

**Figure 11-12. Typical IIC Interrupt Routine**

# Chapter 12
# Freescale Controller Area Network (S08MSCANV1)

## 12.1 Introduction

The Freescale controller area network (MSCAN) is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991. To fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to gain familiarity 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.

The MSCAN module is available in all devices in the MC9S08DZ60 Series.

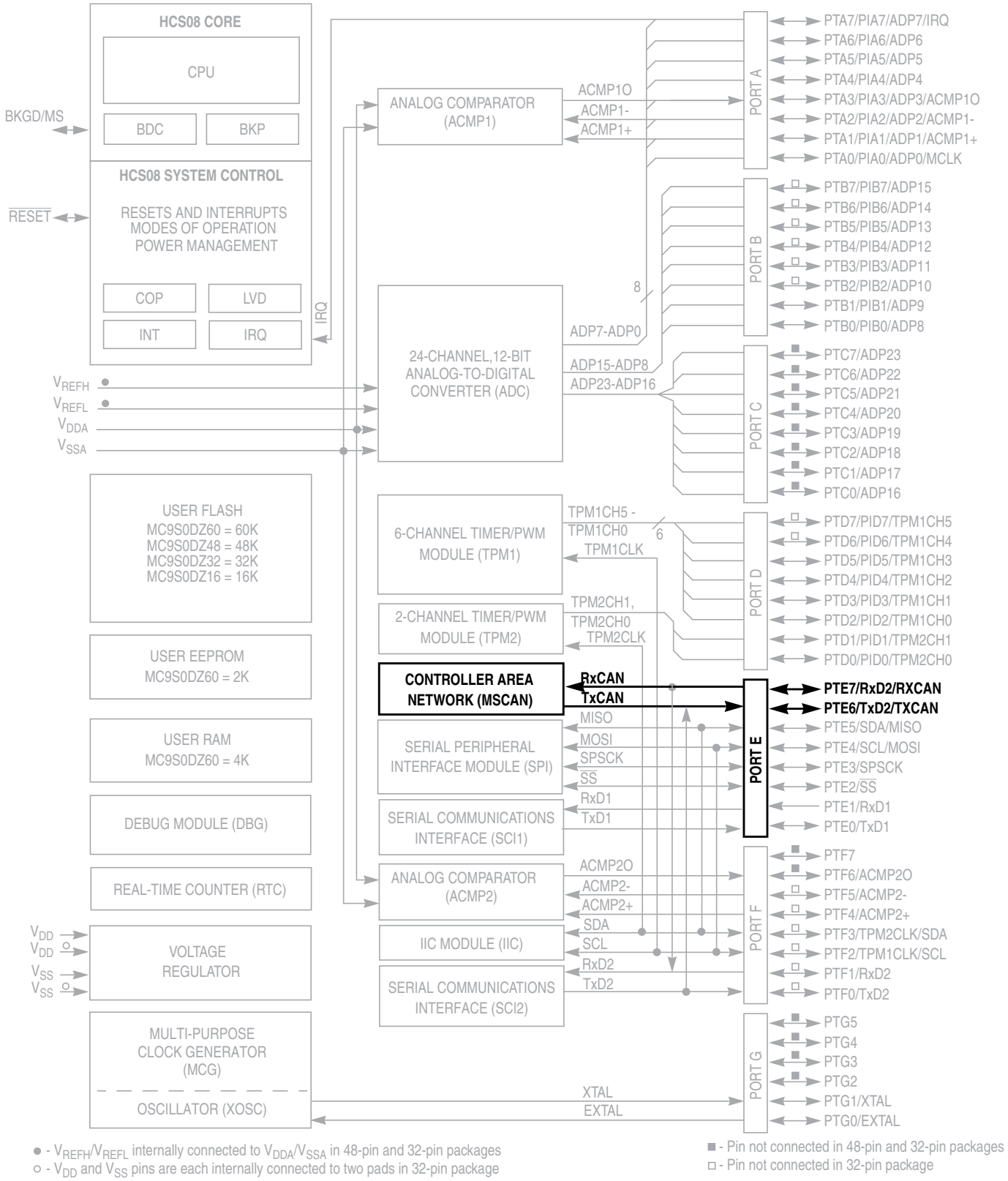



**Figure 12-1. MC9S08DZ60 Block Diagram**

### 12.1.1 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 wakeup 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

### 12.1.2 Modes of Operation

The following modes of operation are specific to the MSCAN. See Section 12.5, "Functional Description," for details.

- Listen-Only Mode
- MSCAN Sleep Mode
- MSCAN Initialization Mode
- MSCAN Power Down Mode
- Loopback Self Test Mode

1. Depending on the actual bit timing and the clock jitter of the PLL.

### 12.1.3 Block Diagram

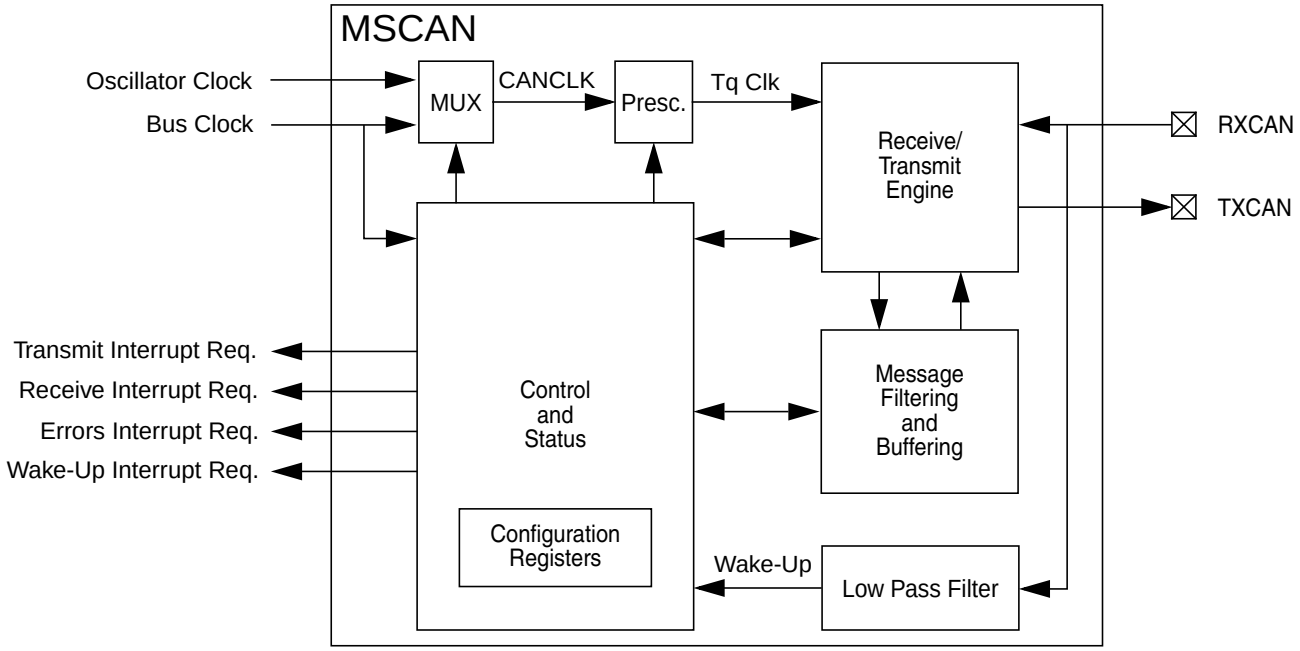



**Figure 12-2. MSCAN Block Diagram**

## 12.2 External Signal Description

The MSCAN uses two external pins:

### 12.2.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

### 12.2.2 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

### 12.2.3 CAN System

A typical CAN system with MSCAN is shown in Figure 12-3. Each CAN node 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 nodes.

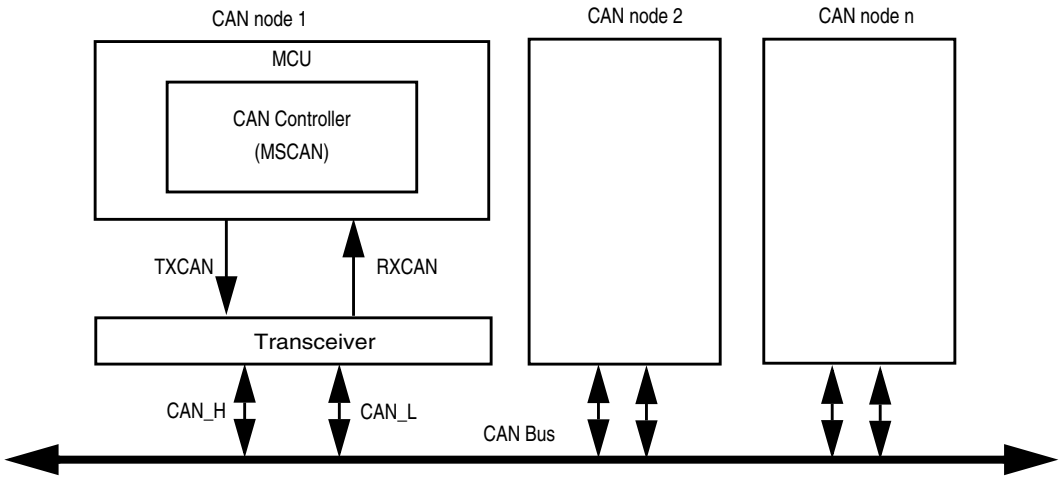



**Figure 12-3. CAN System**

## 12.3 Register Definition

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.

### 12.3.1 MSCAN Control Register 0 (CANCTL0)

The CANCTL0 register provides various control bits of the MSCAN module as described below.

| | 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 12-4. MSCAN Control Register 0 (CANCTL0)**

**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).

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).

**Table 12-1. CANCTL0 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RXFRM[1] | **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. The flag is controlled by the receiver front end. This bit is not valid in loopback mode.<br>0 MSCAN is transmitting or idle[2]<br>1 MSCAN is receiving a message (including when arbitration is lost)[2] |
| 5<br>CSWAI[3] | **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. As soon as a message is acknowledged on the CAN bus, the time stamp will be written to the highest bytes (0x000E, 0x000F) in the appropriate buffer (see Section 12.4, "Programmer's Model of Message Storage"). 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[4] | **Wake-Up Enable** — This configuration bit allows the MSCAN to restart from sleep mode when traffic on CAN is detected (see Section 12.5.5.4, "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 12-1. CANCTL0 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>SLPRQ[5] | **Sleep Mode Request** — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see Section 12.5.5.4, "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 12.3.2, "MSCAN Control Register 1 (CANCTL1)"). SLPRQ cannot be set while the WUPIF flag is set (see Section 12.3.4.1, "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[6,7] | **Initialization Mode Request** — When this bit is set by the CPU, the MSCAN skips to initialization mode (see Section 12.5.5.5, "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 12.3.2, "MSCAN Control Register 1 (CANCTL1)").<br>The following registers enter their hard reset state and restore their default values: CANCTL0[8], CANRFLG[9], CANRIER[10], 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 The MSCAN must be in normal mode for this bit to become set.

2 See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.

3 In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see Section 12.5.5.2, "Operation in Wait Mode" and Section 12.5.5.3, "Operation in Stop Mode").

4 The CPU has to make sure that the WUPE bit and the WUPIE wake-up interrupt enable bit (see Section 12.3.5, "MSCAN Receiver Interrupt Enable Register (CANRIER)) is enabled, if the recovery mechanism from stop or wait is required.

5 The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPAK = 1).

6 The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).

7 In order to protect from accidentally violating the CAN protocol, the TXCAN pin 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.

8 Not including WUPE, INITRQ, and SLPRQ.

9 TSTAT1 and TSTAT0 are not affected by initialization mode.

10 RSTAT1 and RSTAT0 are not affected by initialization mode.

## 12.3.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

| | 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 12-5. MSCAN Control Register 1(CANCTL1)**

Read: Anytime
Write: Anytime when 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 12-2. 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 12.5.3.3, "Clock System," and Section Figure 12-42., "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.Section 12.5.4.6, "Loopback Self Test Mode.<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 12.5.4.4, "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 bits configures the bus-off state recovery mode of the MSCAN. Refer to Section 12.6.2, "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 12.5.5.4, "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 12-2. 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 12.5.5.4, "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.CPU clearing the SLPRQ bit will also reset the SLPAK bit.<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 12.5.5.5, "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 is in initialization mode |

## 12.3.3 MSCAN Bus Timing Register 0 (CANBTR0)

The CANBTR0 register configures various CAN bus timing parameters of the MSCAN module.

| | 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 12-6. MSCAN Bus Timing Register 0 (CANBTR0)**

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 12-3. 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 12-4). |
| 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 12-5). |

**Table 12-4. 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 12-5. 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 |

## 12.3.4 MSCAN Bus Timing Register 1 (CANBTR1)

The CANBTR1 register configures various CAN bus timing parameters of the MSCAN module.

| | 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 12-7. MSCAN Bus Timing Register 1 (CANBTR1)**

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 12-6. 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 12-43). Time segment 2 (TSEG2) values are programmable as shown in Table 12-7. |
| 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 12-43). Time segment 1 (TSEG1) values are programmable as shown in Table 12-8. |

[1] In this case, PHASE_SEG1 must be at least 2 time quanta (Tq).

**Table 12-7. 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 12-35 for valid settings.

**Table 12-8. 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 cycles[1] |
| 0 | 0 | 1 | 0 | 3 Tq clock cycles[1] |
| 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 12-35 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 12-7 and Table 12-8).

***Eqn. 12-1***

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{CANCLK}} \bullet (1 + \text{TimeSegment1} + \text{TimeSegment2})$$

### 12.3.4.1 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.

| | 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 12-8. MSCAN Receiver Flag Register (CANRFLG)**

**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).

Read: Anytime
Write: Anytime when out of 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.

**Table 12-9. 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 12.5.5.4, "MSCAN Sleep Mode,") and WUPE = 1 in CANTCTL0 (see Section 12.3.1, "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 12.3.5, "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 ≤ 127<br>10 RxERR: 127 < receive error counter<br>11 Bus-off[1]: transmit error counter > 255 |
| 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 ≤ 127<br>10 TxERR: 127 < transmit error counter ≤ 255<br>11 Bus-Off: transmit error counter > 255 |

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

**Table 12-9. 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.

## 12.3.5 MSCAN Receiver Interrupt Enable Register (CANRIER)

This register contains the interrupt enable bits for the interrupt flags described in the CANRFLG register.

| | 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 12-9. MSCAN Receiver Interrupt Enable Register (CANRIER)**

**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.

Read: Anytime
Write: Anytime when not in initialization mode

**Table 12-10. 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 12.3.1, "MSCAN Control Register 0 (CANCTL0)") must both be enabled if the recovery mechanism from stop or wait is required.

2 Bus-off state is defined by the CAN standard (see Bosch CAN 2.0A/B protocol specification: for only transmitters. 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 12.3.4.1, "MSCAN Receiver Flag Register (CANRFLG)").

## 12.3.6 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

| | 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 12-10. MSCAN Transmitter Flag Register (CANTFLG)**

**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).

Read: Anytime
Write: Anytime for TXEx flags when not in initialization mode; write of 1 clears flag, write of 0 is ignored

**Table 12-11. 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 12.3.8, "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 12.3.9, "MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)"). When a TXEx flag is set, the corresponding ABTRQx bit is cleared (see Section 12.3.8, "MSCAN Transmitter Message Abort Request Register (CANTARQ)").<br>When listen-mode is active (see Section 12.3.2, "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 are 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) |

## 12.3.7 MSCAN Transmitter Interrupt Enable Register (CANTIER)

This register contains the interrupt enable bits for the transmit buffer empty interrupt flags.

| | 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 12-11. MSCAN Transmitter Interrupt Enable Register (CANTIER)**

**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).

Read: Anytime
Write: Anytime when not in initialization mode

**Table 12-12. 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. See Section 12.5.2.2, "Transmit Structures" for details. |

## 12.3.8 MSCAN Transmitter Message Abort Request Register (CANTARQ)

The CANTARQ register allows abort request of messages queued for transmission.

| | 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 12-12. MSCAN Transmitter Message Abort Request Register (CANTARQ)**

**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).

Read: Anytime
Write: Anytime when not in initialization mode

**Table 12-13. 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 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)") and abort acknowledge flags (ABTAK, see Section 12.3.9, "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 |

## 12.3.9 MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)

The CANTAAK register indicates the successful abort of messages queued for transmission, if requested by the appropriate bits in the CANTARQ register.

| | 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 12-13. MSCAN Transmitter Message Abort Acknowledge Register (CANTAAK)**

**NOTE**

The CANTAAK register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

Read: Anytime
Write: Unimplemented for ABTAKx flags

**Table 12-14. CANTAAK Register Field Descriptions**

| Field | Description |
|---|---|
| 2:0<br>ABTAK[2:0] | **Abort Acknowledge** — This flag acknowledges that a message was aborted due to a pending transmission 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. |

## 12.3.10 MSCAN Transmit Buffer Selection Register (CANTBSEL)

The CANTBSEL selections of the actual transmit message buffer, which is accessible in the CANTXFG register space.

| | 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 12-14. MSCAN Transmit Buffer Selection Register (CANTBSEL)**

**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).

Read: Find the lowest ordered bit set to 1, all other bits will be read as 0
Write: Anytime when not in initialization mode

**Table 12-15. 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 12.3.6, "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 the selection of the next available Tx buffer.

- LDD CANTFLG; value read is 0b0000_0110
- STD CANTBSEL; value written is 0b0000_0110
- LDD CANTBSEL; value read is 0b0000_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG buffer register.

## 12.3.11 MSCAN Identifier Acceptance Control Register (CANIDAC)

The CANIDAC register is used for identifier filter acceptance control as described below.

| | 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 12-15. MSCAN Identifier Acceptance Control Register (CANIDAC)**

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHITx, which are read-only

**Table 12-16. 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 12.5.3, "Identifier Acceptance Filter"). Table 12-17 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 12.5.3, "Identifier Acceptance Filter"). Table 12-18 summarizes the different settings. |

**Table 12-17. 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 12-18. 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.

## 12.3.12 MSCAN Miscellaneous Register (CANMISC)

This register provides additional features.

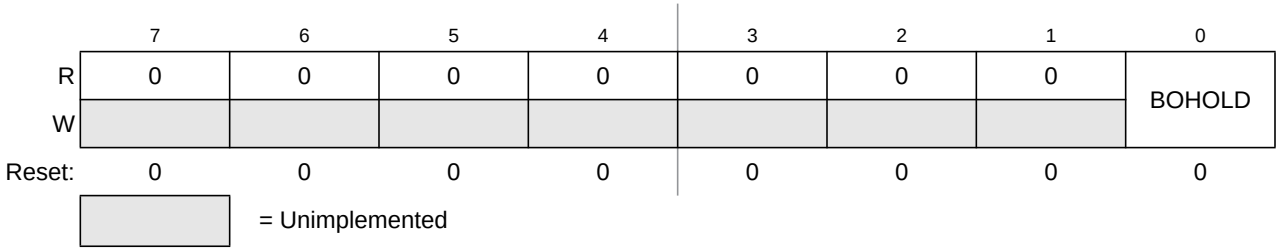



**Figure 12-16. MSCAN Miscellaneous Register (CANMISC)**

Read: Anytime
Write: Anytime; write of '1' clears flag; write of '0' ignored

**Table 12-19. CANMISC Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>BOHOLD | **Bus-off State Hold Until User Request** — If BORM is set in Section 12.3.2, "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 12.6.2, "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 |

## 12.3.13 MSCAN Receive Error Counter (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

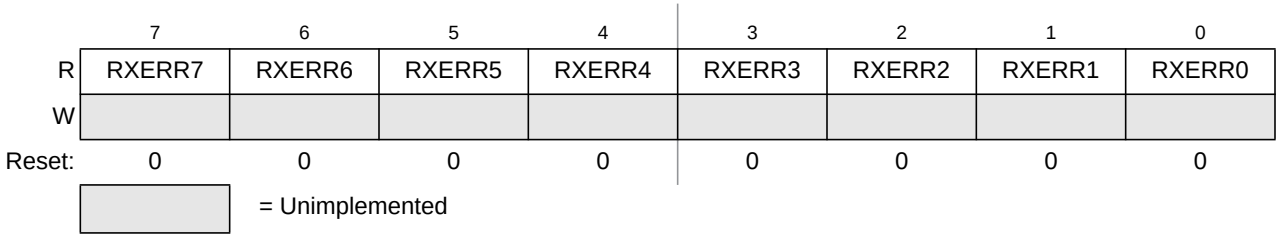



**Figure 12-17. MSCAN Receive Error Counter (CANRXERR)**

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.

Writing to this register when in special modes can alter the MSCAN functionality.

## 12.3.14 MSCAN Transmit Error Counter (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

| | 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 12-18. MSCAN Transmit Error Counter (CANTXERR)**

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.

Writing to this register when in special modes can alter the MSCAN functionality.

## 12.3.15 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 12.4.1, "Identifier Registers (IDR0–IDR3)") of incoming messages in a bit by bit manner (see Section 12.5.3, "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.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-19. MSCAN Identifier Acceptance Registers (First Bank) — CANIDAR0–CANIDAR3**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 12-20. 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. |

| | 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 12-20. MSCAN Identifier Acceptance Registers (Second Bank) — CANIDAR4–CANIDAR7**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 12-21. 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. |

## 12.3.16 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."

| | 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 12-21. MSCAN Identifier Mask Registers (First Bank) — CANIDMR0–CANIDMR3**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 12-22. 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 (don't care) |

| | 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 12-22. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7**

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 12-23. 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 (don't care) |

## 12.4 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 12.3.1, "MSCAN Control Register 0 (CANCTL0)").

The time stamp register is written by the MSCAN. The CPU can only read these registers.

**Table 12-24. Message Buffer Organization**

| Offset Address | Register | Access |
|---|---|---|
| 0x00X0 | Identifier Register 0 | |
| 0x00X1 | Identifier Register 1 | |
| 0x00X2 | Identifier Register 2 | |
| 0x00X3 | Identifier Register 3 | |
| 0x00X4 | Data Segment Register 0 | |
| 0x00X5 | Data Segment Register 1 | |
| 0x00X6 | Data Segment Register 2 | |
| 0x00X7 | Data Segment Register 3 | |
| 0x00X8 | Data Segment Register 4 | |
| 0x00X9 | Data Segment Register 5 | |
| 0x00XA | Data Segment Register 6 | |
| 0x00XB | Data Segment Register 7 | |
| 0x00XC | Data Length Register | |
| 0x00XD | Transmit Buffer Priority Register[1] | |
| 0x00XE | Time Stamp Register (High Byte)[2] | |
| 0x00XF | Time Stamp Register (Low Byte)[3] | |

1 Not applicable for receive buffers

2 Read-only for CPU

3 Read-only for CPU

Figure 12-23 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 12-24.

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 priority registers are 0 out of reset.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| IDR0 | R<br>W | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| IDR1 | R<br>W | ID20 | ID19 | ID18 | SRR(1) | IDE(1) | ID17 | ID16 | ID15 |
| IDR2 | R<br>W | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| IDR3 | R<br>W | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR2 |
| DSR0 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR1 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR2 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR3 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR4 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR5 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR6 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DSR7 | R<br>W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| DLR | R<br>W | | | | | DLC3 | DLC2 | DLC1 | DLC0 |

= Unused, always read 'x'

**Figure 12-23. Receive/Transmit Message Buffer — Extended Identifier Mapping**

1 SRR and IDE are both 1s.

2 The position of RTR differs between extended and standard indentifier mapping.

Read: For transmit buffers, anytime when TXEx flag is set (see Section 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see

Section 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). For receive buffers, only when RXF flag is set (see Section 12.3.4.1, "MSCAN Receiver Flag Register (CANRFLG)").

Write: For transmit buffers, anytime when TXEx flag is set (see Section 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). Unimplemented for receive buffers.

Reset: Undefined (0x00XX) because of RAM-based implementation

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| IDR0 | R | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| | W | | | | | | | | |
| IDR1 | R | ID2 | ID1 | ID0 | RTR[1] | IDE[2] | | | |
| | W | | | | | | | | |
| IDR2 | R | | | | | | | | |
| | W | | | | | | | | |
| IDR3 | R | | | | | | | | |
| | W | | | | | | | | |

= Unused, always read 'x'

**Figure 12-24. Receive/Transmit Message Buffer — Standard Identifier Mapping**

1 The position of RTR differs between extended and standard indentifier mapping.

2 IDE is 0.

## 12.4.1 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 bits. The identifier registers for a standard format identifier consist of a total of 13 bits; ID[10:0], RTR, and IDE bits.

### 12.4.1.1 IDR0–IDR3 for Extended Identifier Mapping

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| W | | | | | | | | |
| Reset: | x | x | x | x | x | x | x | x |

**Figure 12-25. Identifier Register 0 (IDR0) — Extended Identifier Mapping**

**Table 12-25. 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. |

| | 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 12-26. Identifier Register 1 (IDR1) — Extended Identifier Mapping**

1 SRR and IDE are both 1s.

**Table 12-26. 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. |

| | 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 12-27. Identifier Register 2 (IDR2) — Extended Identifier Mapping**

**Table 12-27. 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. |

| | 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 12-28. Identifier Register 3 (IDR3) — Extended Identifier Mapping**

**Table 12-28. 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 arbithation 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 |

## 12.4.2 IDR0–IDR3 for Standard Identifier Mapping

| | 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 12-29. Identifier Register 0 — Standard Mapping**

**Table 12-29. 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 12-30. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ID2 | ID1 | ID0 | RTR | IDE[1] | | | |
| Reset: | x | x | x | x | x | x | x | x |

= Unused; always read 'x'

**Figure 12-30. Identifier Register 1 — Standard Mapping**

1 IDE is 0.

**Table 12-30. 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 12-29. |
| 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) |

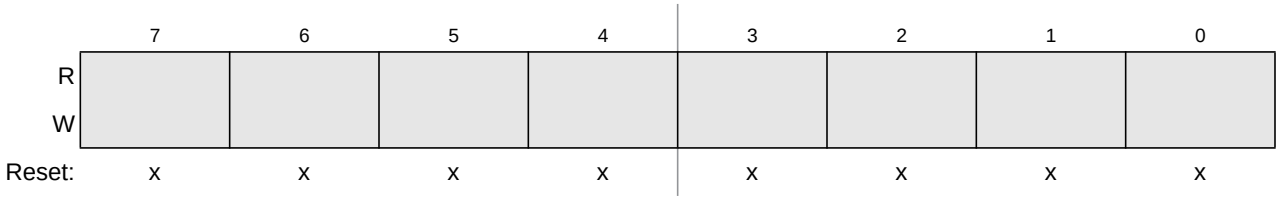



**Figure 12-31. Identifier Register 2 — Standard Mapping**

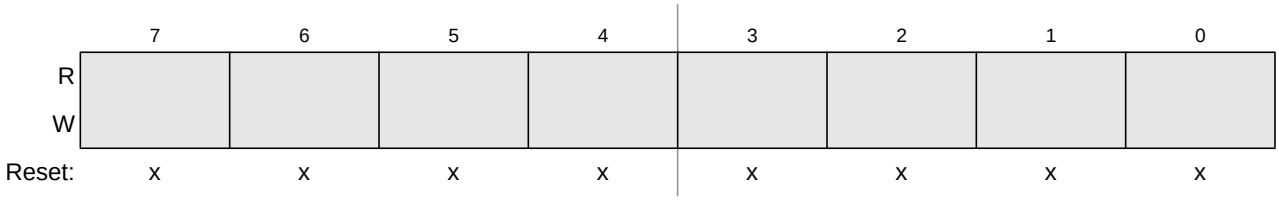



**Figure 12-32. Identifier Register 3 — Standard Mapping**

## 12.4.3 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.

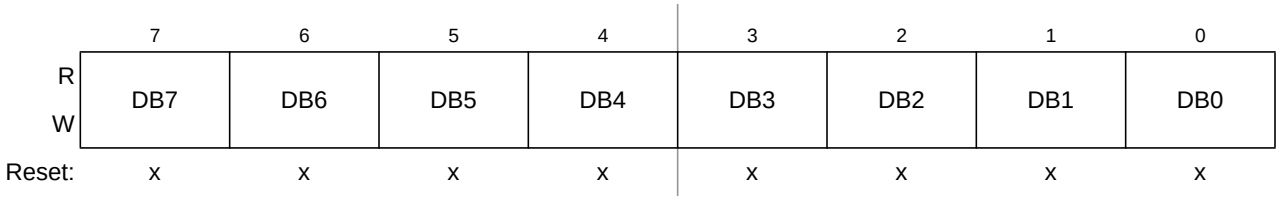



**Figure 12-33. Data Segment Registers (DSR0–DSR7) — Extended Identifier Mapping**

**Table 12-31. DSR0–DSR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>DB[7:0] | **Data bits 7:0** |

## 12.4.4 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

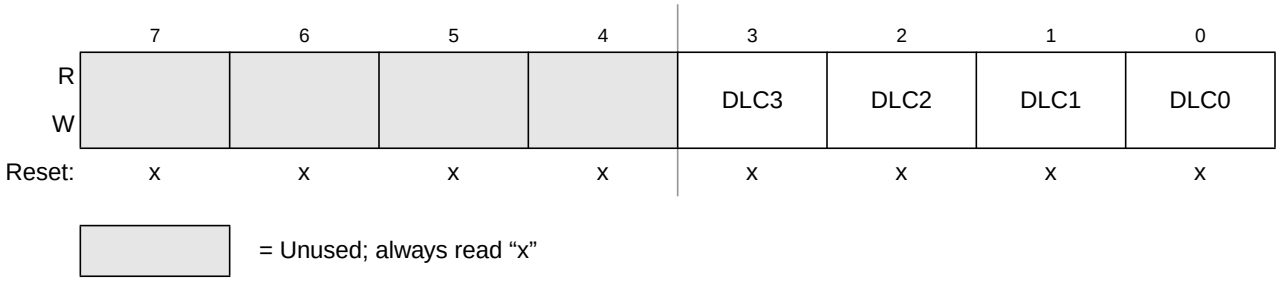



**Figure 12-34. Data Length Register (DLR) — Extended Identifier Mapping**

**Table 12-32. 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 12-33 shows the effect of setting the DLC bits. |

**Table 12-33. 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 |

## 12.4.5 Transmit Buffer Priority Register (TBPR)

This register defines the local priority of the associated message transmit 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.

| | 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 12-35. Transmit Buffer Priority Register (TBPR)**

Read: Anytime when TXEx flag is set (see Section 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").

Write: Anytime when TXEx flag is set (see Section 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").

## 12.4.6 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 as soon as a message has been acknowledged on the CAN bus (see

Section 12.3.1, "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.

| | 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 12-36. Time Stamp Register — High Byte (TSRH)**

| | 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 12-37. Time Stamp Register — Low Byte (TSRL)**

Read: Anytime when TXEx flag is set (see Section 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)") and the corresponding transmit buffer is selected in CANTBSEL (see Section 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)").

Write: Unimplemented

## 12.5 Functional Description

### 12.5.1 General

This section provides a complete functional description of the MSCAN. It describes each of the features and modes listed in the introduction.

## 12.5.2 Message Storage

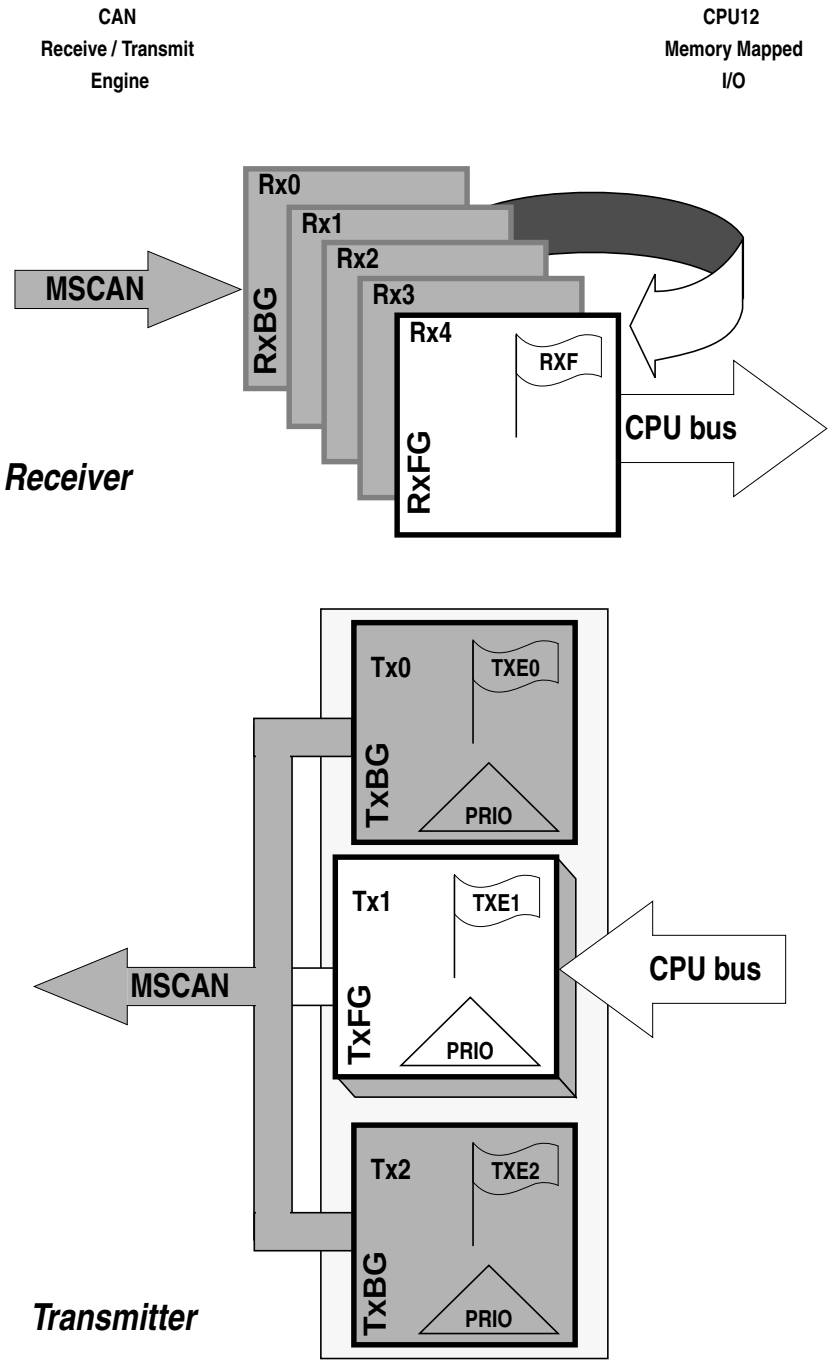



**Figure 12-38. User Model for Message Buffer Organization**

MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

### 12.5.2.1 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 12.5.2.2, "Transmit Structures."

### 12.5.2.2 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 12-38.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 12.4, "Programmer's Model of Message Storage"). An additional Section 12.4.5, "Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO) (see Section 12.4.5, "Transmit Buffer Priority Register (TBPR)"). The remaining two bytes are used for time stamping of a message, if required (see Section 12.4.6, "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 12.3.6, "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 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). This makes the respective buffer accessible within the CANTXFG address space (see Section 12.4, "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 12.5.7.2, "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 12.3.8, "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).

#### 12.5.2.3 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 12-38). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 12-38). 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 12.4, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 12.3.4.1, "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 12.5.3, "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[2], sets the RXF flag, and generates a receive interrupt (see Section 12.5.7.3, "Receive Interrupt") to the CPU[3]. 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

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.
2. Only if the RXF flag is not set.
3. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

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 12.3.2, "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 12.5.7.5, "Error Interrupt"). The MSCAN remains able to transmit messages while the receiver FIFO is full, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

### 12.5.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see Section 12.3.11, "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 12.3.16, "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 12.3.11, "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 (see Bosch CAN 2.0A/B protocol specification):

- 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[1]. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Figure 12-39 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.

1. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

- Four identifier acceptance filters, each to be applied to
  - a) the 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages or
  - b) the 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 12-40 shows how the first 32-bit filter bank (CANIDAR0–CANIDA3, CANIDMR0–3CANIDMR) 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 12-41 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.

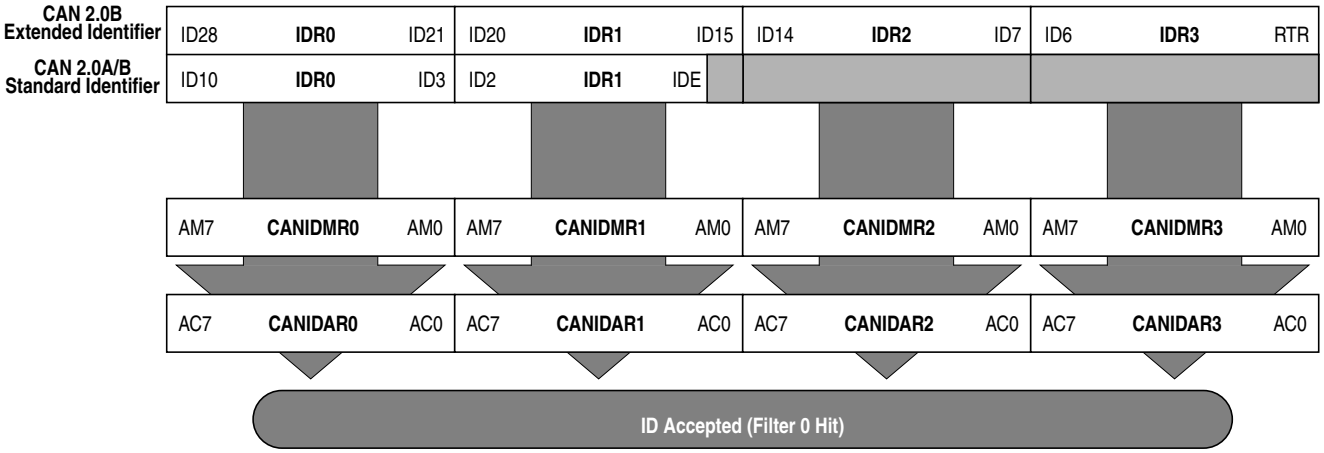



**Figure 12-39. 32-bit Maskable Identifier Acceptance Filter**

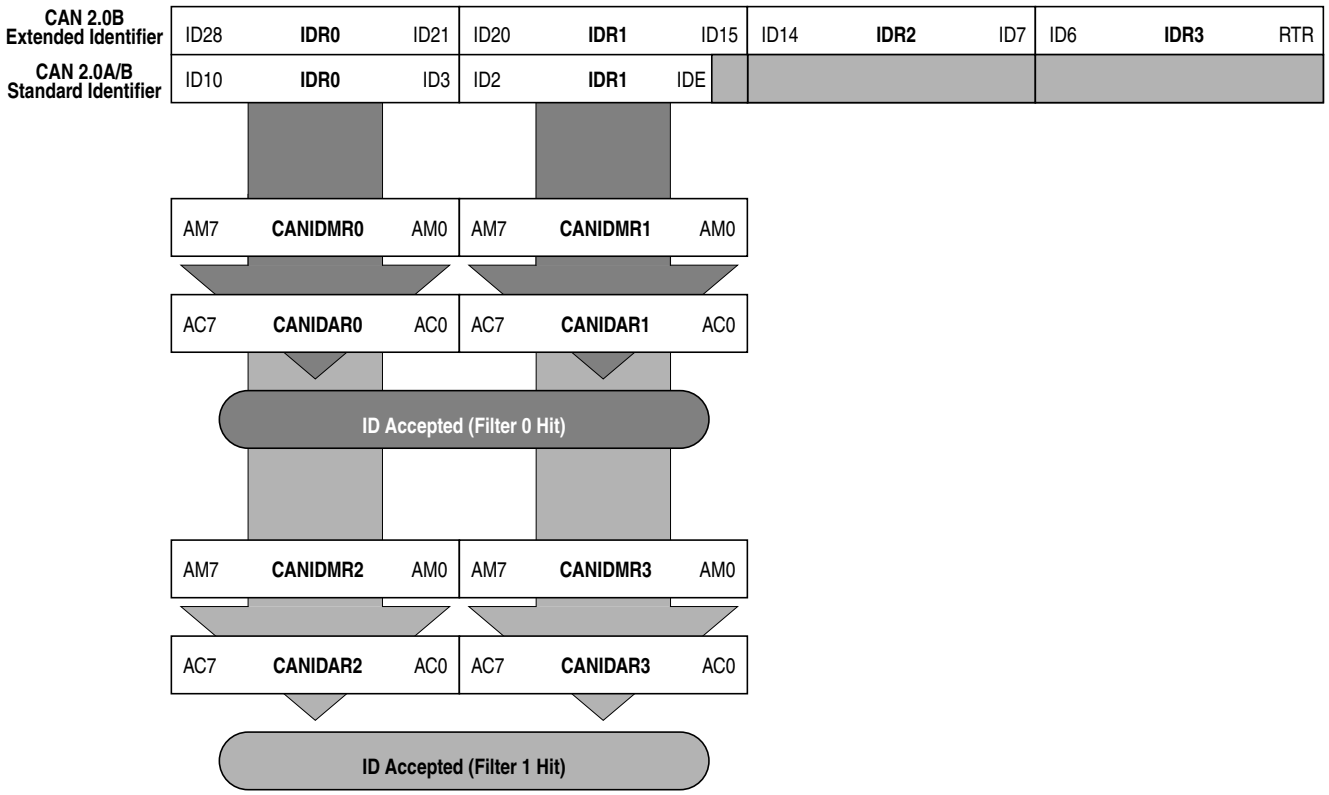



**Figure 12-40. 16-bit Maskable Identifier Acceptance Filters**

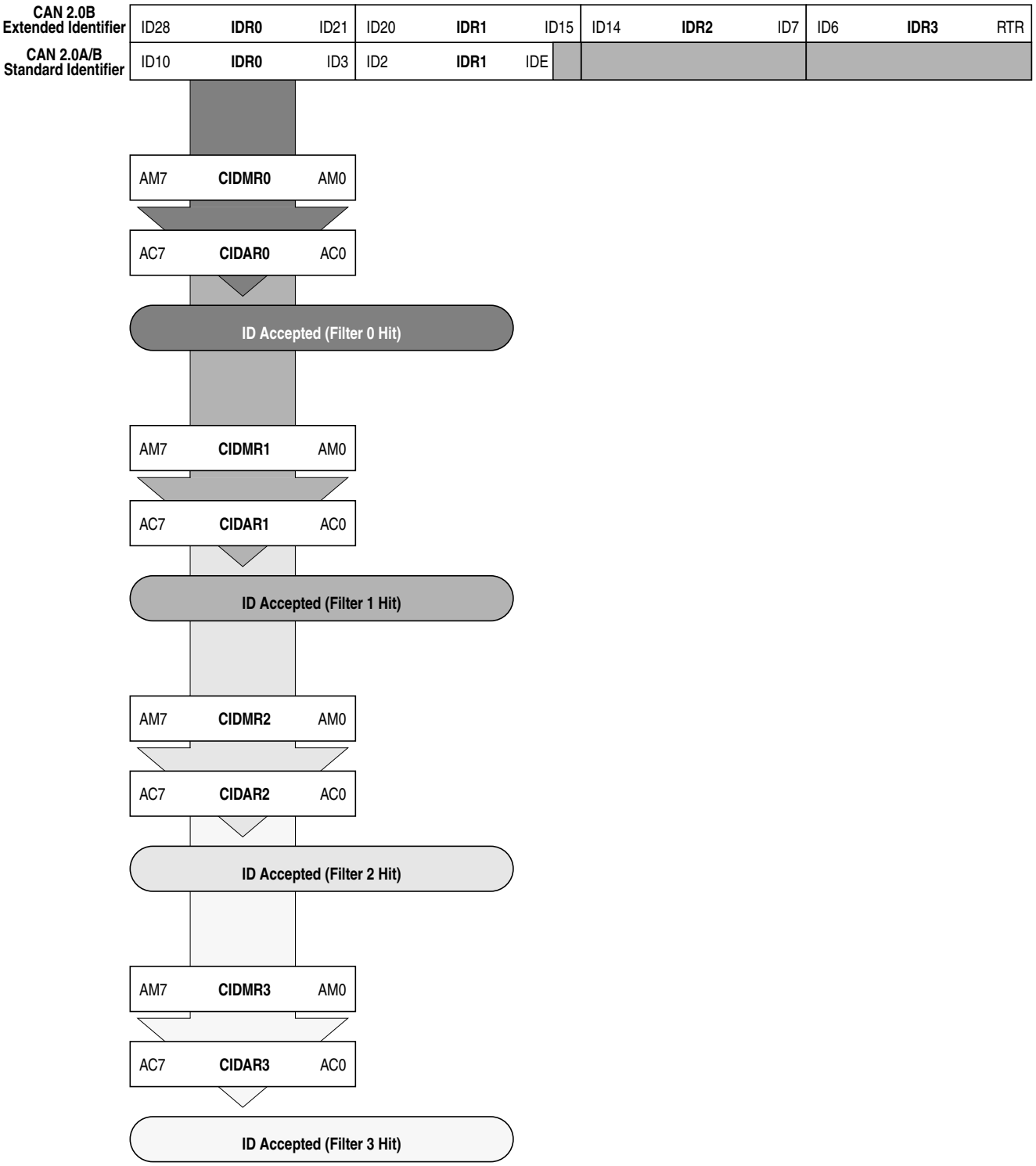



**Figure 12-41. 8-bit Maskable Identifier Acceptance Filters**

MSCAN filter uses three sets of registers to provide the filter configuration. Firstly, the CANIDAC register determines the configuration of the banks into filter sizes and number of filters. Secondly, registers CANIDMR0/1/2/3 determine those bits on which the filter will operate by placing a '0' at the appropriate

bit position in the filter register. Finally, registers CANIDAR0/1/2/3 determine the value of those bits determined by CANIDMR0/1/2/3.

For instance in the case of the filter value of:

0001x1001x0

The CANIDMR0/1/2/3 register would be configured as:

00001000010

and so all message identifier bits except bit 1 and bit 6 would be compared against the CANIDAR0/1/2/3 registers. These would be configured as:

00010100100

In this case bits 1 and 6 are set to '0', but since they are ignored it is equally valid to set them to '1'.

#### 12.5.3.1 Identifier Acceptance Filters example

As described above, filters work by comparisons to individual bits in the CAN message identifier field. The filter will check each one of the eleven bits of a standard CAN message identifier. Suppose a filter value of 0001x1001x0. In this simple example, there are only three possible CAN messages.

Filter value: 0001x1001x0

Message 1: 00011100110

Message 2: 00110100110

Message 3: 00010100100

Message 2 will be rejected since its third most significant bit is not '0' - 001. The filter is simply a convenient way of defining the set of messages that the CPU must receive. For full 29-bits of an extended CAN message identifier, the filter identifies two sets of messages: one set that it receives and one set that it rejects. Alternatively, the filter may be split into two. This allows the MSCAN to examine only the first 16 bits of a message identifier, but allows two separate filters to perform the checking. See the example below:

Filter value A: 0001x1001x0

Filter value B: 00x101x01x0

Message 1: 00011100110

Message 2: 00110100110

Message 3: 00010100100

MSCAN will accept all three messages. Filter A will accept messages 1 and 3 as before and filter B will accept message 2. In practice, it is unimportant which filter accepts the message - messages accepted by either will be placed in the input buffer. A message may be accepted by more than one filter.

### 12.5.3.2 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 12.3.1, "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 pin is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode (see Section 12.5.5.6, "MSCAN Power Down Mode," and Section 12.5.5.5, "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.

### 12.5.3.3 Clock System

Figure 12-42 shows the structure of the MSCAN clock generation circuitry.

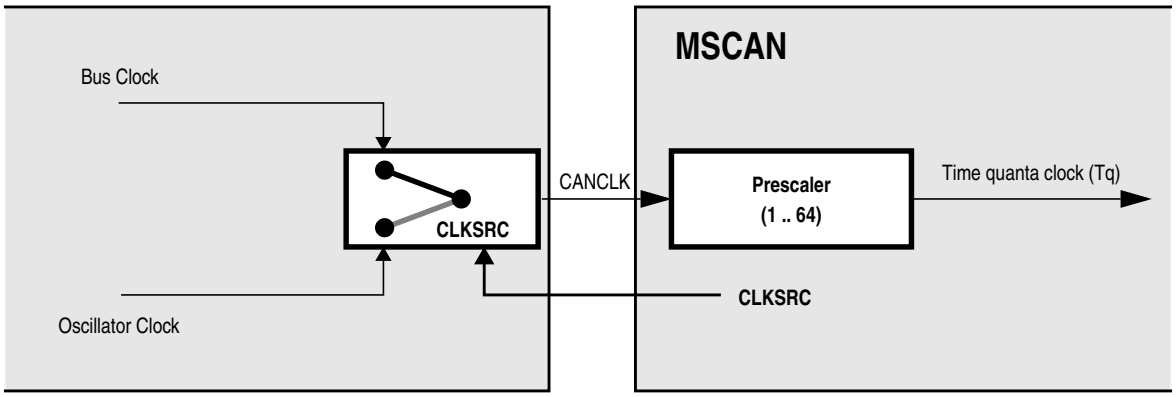



**Figure 12-42. MSCAN Clocking Scheme**

The clock source bit (CLKSRC) in the CANCTL1 register (12.3.2/-226) 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. PLL lock may also be too wide to ensure adequate clock tolerance.

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. 12-2***

$$Tq = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments as described in the Bosch CAN specification. (see Figure 12-43):

- 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. 12-3***

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

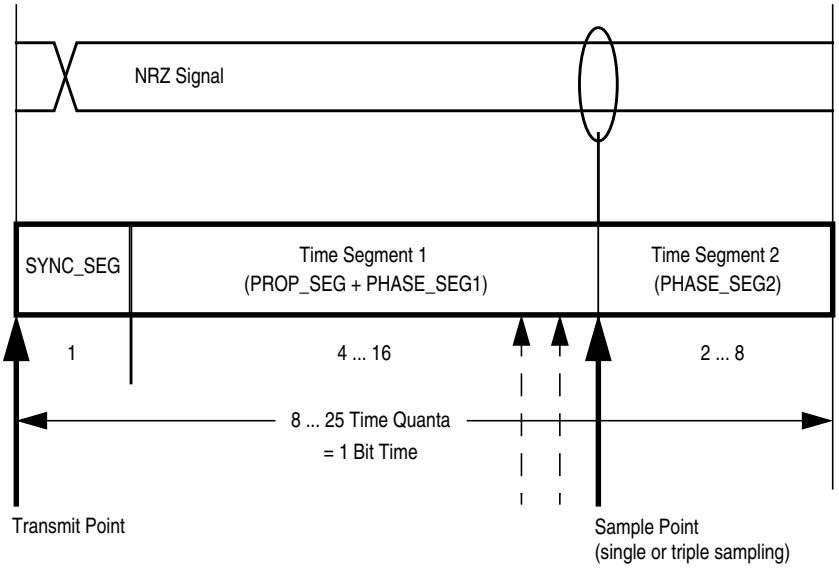



**Figure 12-43. Segments within the Bit Time**

**Table 12-34. 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 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 12.3.3, "MSCAN Bus Timing Register 0 (CANBTR0)" and Section 12.3.4, "MSCAN Bus Timing Register 1 (CANBTR1)").

Table 12-35 gives an overview of the CAN 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 12-35. CAN Standard 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 |

## 12.5.4 Modes of Operation

### 12.5.4.1 Normal Modes

The MSCAN module behaves as described within this specification in all normal system operation modes.

### 12.5.4.2 Special Modes

The MSCAN module behaves as described within this specification in all special system operation modes.

#### 12.5.4.3 Emulation Modes

In all emulation modes, the MSCAN module behaves just like normal system operation modes as described within this specification.

#### 12.5.4.4 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 transmision. 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.

#### 12.5.4.5 Security Modes

The MSCAN module has no security features.

#### 12.5.4.6 Loopback Self Test Mode

Loopback self test mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input. The RXCAN input pin 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.

### 12.5.5 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 12-36 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.

For all modes, an MSCAN wake-up interrupt can occur only if the MSCAN is in sleep mode (SLPRQ = 1 and SLPAK = 1), wake-up functionality is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

**Table 12-36. 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 |
| **Stop3** | | CSWAI = X[2]<br>SLPRQ = 1<br>SLPAK = 1 | CSWAI = X<br>SLPRQ = 0<br>SLPAK = 0 | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |
| **Stop1 or 2** | | | CSWAI = X<br>SLPRQ = X<br>SLPAK = X | CSWAI = X<br>SLPRQ = X<br>SLPAK = X |

1 'X' means don't care.

2 For a safe wake up from Sleep mode, SLPRQ and SLPAK must be set to 1 before going into Stop3 mode.

### 12.5.5.1 Operation in Run Mode

As shown in Table 12-36, only MSCAN sleep mode is available as low power option when the CPU is in run mode.

### 12.5.5.2 Operation in Wait Mode

The WAIT 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 its internal controllers 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). The MSCAN can also operate in any of the low-power modes depending on the values of the SLPRQ/SLPAK and CSWAI bits as seen in Table 12-36.

### 12.5.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop1 or stop2 modes, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK. In stop3 mode, power down or sleep modes are determined by the SLPRQ/SLPAK values set prior to entering stop3. CSWAI bit has no function in any of the stop modes.Table 12-36.

### 12.5.5.4 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.

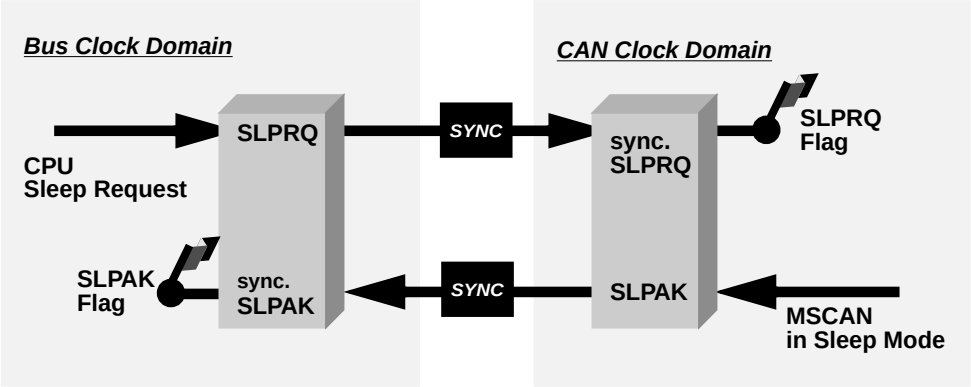



**Figure 12-44. Sleep Request / Acknowledge Cycle**

**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 12-44). 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. The TXCAN pin 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 CANCLT0 is not asserted, the MSCAN will mask any activity it detects on CAN. The RXCAN pin is therefore held internally in a recessive state. This locks the MSCAN in sleep mode (Figure 12-45). 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.

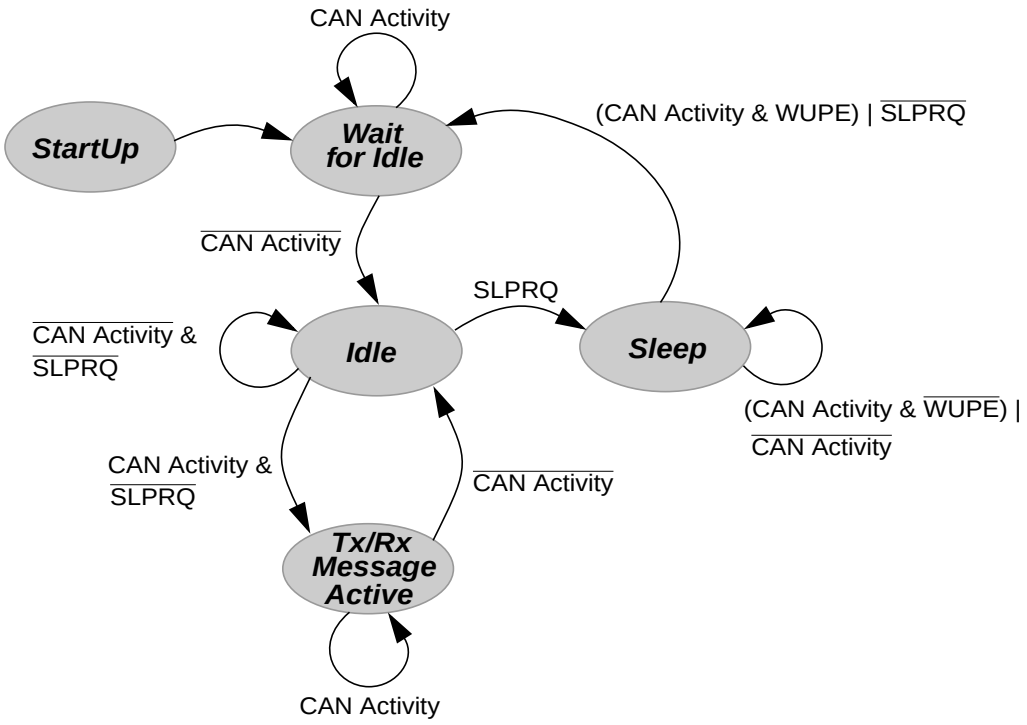



**Figure 12-45. Simplified State Transitions for Entering/Leaving Sleep Mode**

### 12.5.5.5 MSCAN Initialization Mode

In initialization mode, 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 the TXCAN pin 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 12.3.1, "MSCAN Control Register 0 (CANCTL0)," for a detailed description of the initialization mode.

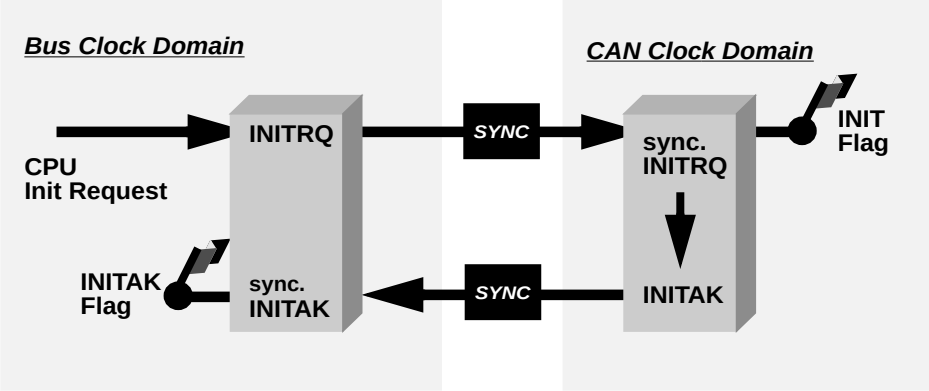



**Figure 12-46. Initialization Request/Acknowledge Cycle**

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 Section Figure 12-46., "Initialization Request/Acknowledge Cycle").

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.

#### 12.5.5.6 MSCAN Power Down Mode

The MSCAN is in power down mode (Table 12-36) 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 the TXCAN pin 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 WAIT 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.

#### 12.5.5.7 Programmable Wake-Up Function

The MSCAN can be programmed to wake up the MSCAN as soon as CAN bus activity is detected (see control bit WUPE in Section 12.3.1, "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 while in sleep mode (see control bit WUPM in Section 12.3.2, "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.

### 12.5.6 Reset Initialization

The reset state of each individual bit is listed in Section 12.3, "Register Definition," which details all the registers and their bit-fields.

### 12.5.7 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.

#### 12.5.7.1 Description of Interrupt Operation

The MSCAN supports four interrupt vectors (see Table 12-37), any of which can be individually masked (for details see sections from Section 12.3.5, "MSCAN Receiver Interrupt Enable Register (CANRIER)," to Section 12.3.7, "MSCAN Transmitter Interrupt Enable Register (CANTIER)").

**NOTE**

The dedicated interrupt vector addresses are defined in the Resets and Interrupts chapter.

**Table 12-37. 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]) |

#### 12.5.7.2 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.

#### 12.5.7.3 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.

#### 12.5.7.4 Wake-Up Interrupt

A wake-up interrupt is generated if activity on the CAN bus occurs during MSCAN internal sleep mode. WUPE (see Section 12.3.1, "MSCAN Control Register 0 (CANCTL0)") must be enabled.

#### 12.5.7.5 Error Interrupt

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurrs. Section 12.3.4.1, "MSCAN Receiver Flag Register (CANRFLG) indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in Section 12.5.2.3, "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 12.3.4.1, "MSCAN Receiver Flag Register (CANRFLG)" and Section 12.3.5, "MSCAN Receiver Interrupt Enable Register (CANRIER)").

#### 12.5.7.6 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the Section 12.3.4.1, "MSCAN Receiver Flag Register (CANRFLG)" or the Section 12.3.6, "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.

#### 12.5.7.7 Recovery from Stop or Wait

The MSCAN can recover from stop or wait via the wake-up interrupt. 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).

## 12.6 Initialization/Application Information

### 12.6.1 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 and enter normal mode

If the configuration of registers which are writable in initialization mode needs to be changed only when the MSCAN module is in normal mode:

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 in normal mode

### 12.6.2 Bus-Off Recovery

The bus-off recovery is user configurable. The bus-off state can either be exited 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 specification for details).

If the MSCAN is configured for user request (BORM set in Section 12.3.2, "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 Section 12.3.12, "MSCAN Miscellaneous Register (CANMISC) has been cleared by the user

These two events may occur in any order.

# Chapter 13
# Serial Peripheral Interface (S08SPIV3)

## 13.1 Introduction

The serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, memories, etc.

The SPI runs at a baud rate up to the bus clock divided by two in master mode and bus clock divided by four in slave mode.

All devices in the MC9S08DZ60 Series MCUs contain one SPI module, as shown in the following block diagram.

**NOTE**

Ensure that the SPI should not be disabled (SPE=0) at the same time as a bit change to the CPHA bit. These changes should be performed as separate operations or unexpected behavior may occur.

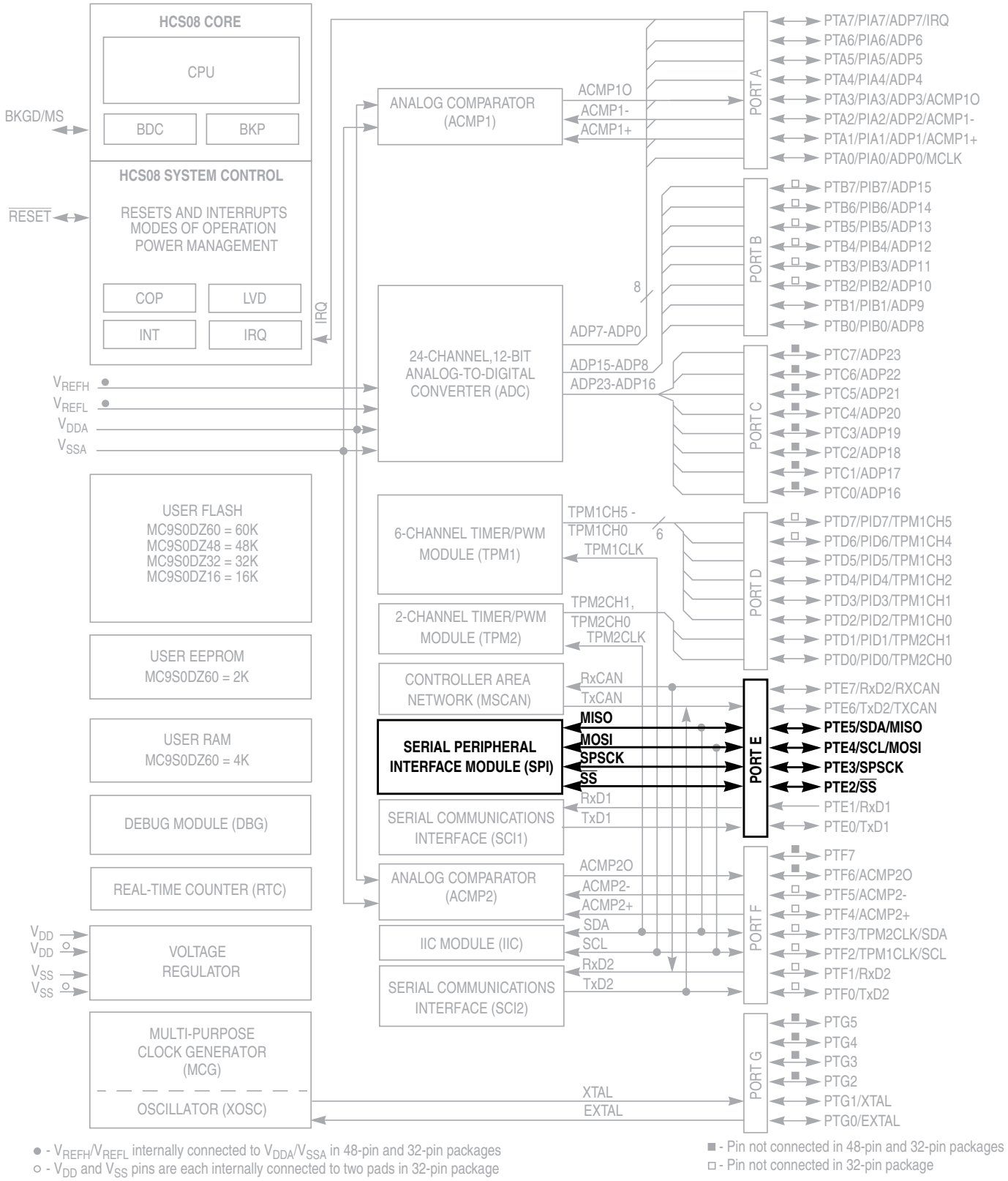



**Figure 13-1. MC9S08DZ60 Block Diagram**

### 13.1.1 Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- Programmable transmit bit rate
- Double-buffered transmit and receive
- Serial clock phase and polarity options
- Slave select output
- Selectable MSB-first or LSB-first shifting

### 13.1.2 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

#### 13.1.2.1 SPI System Block Diagram

Figure 13-2 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.

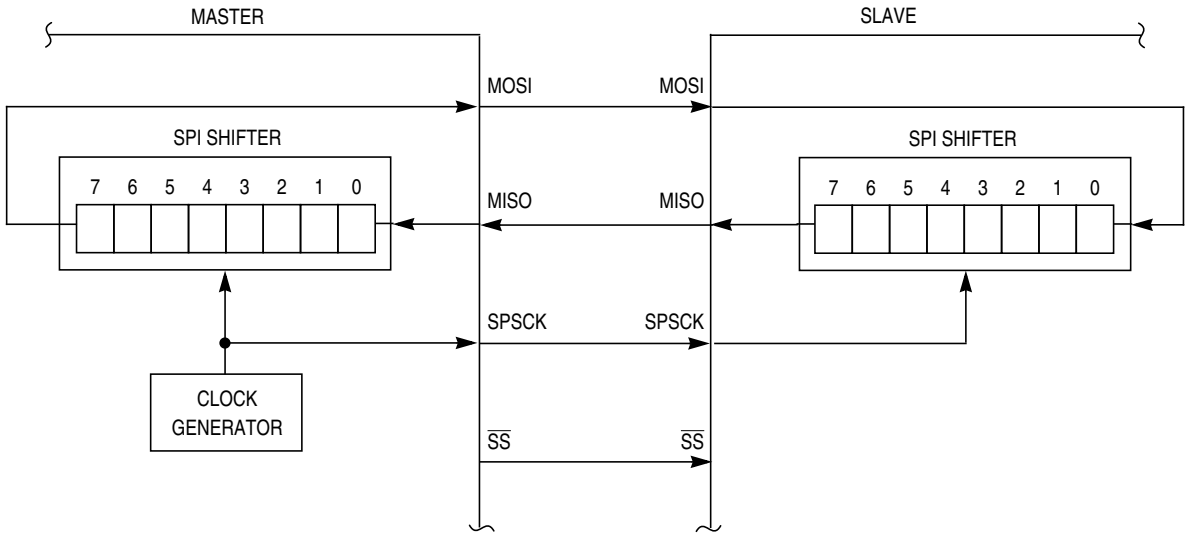



**Figure 13-2. SPI System Connections**

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although Figure 13-2 shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

### 13.1.2.2 SPI Module Block Diagram

Figure 13-3 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPID) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

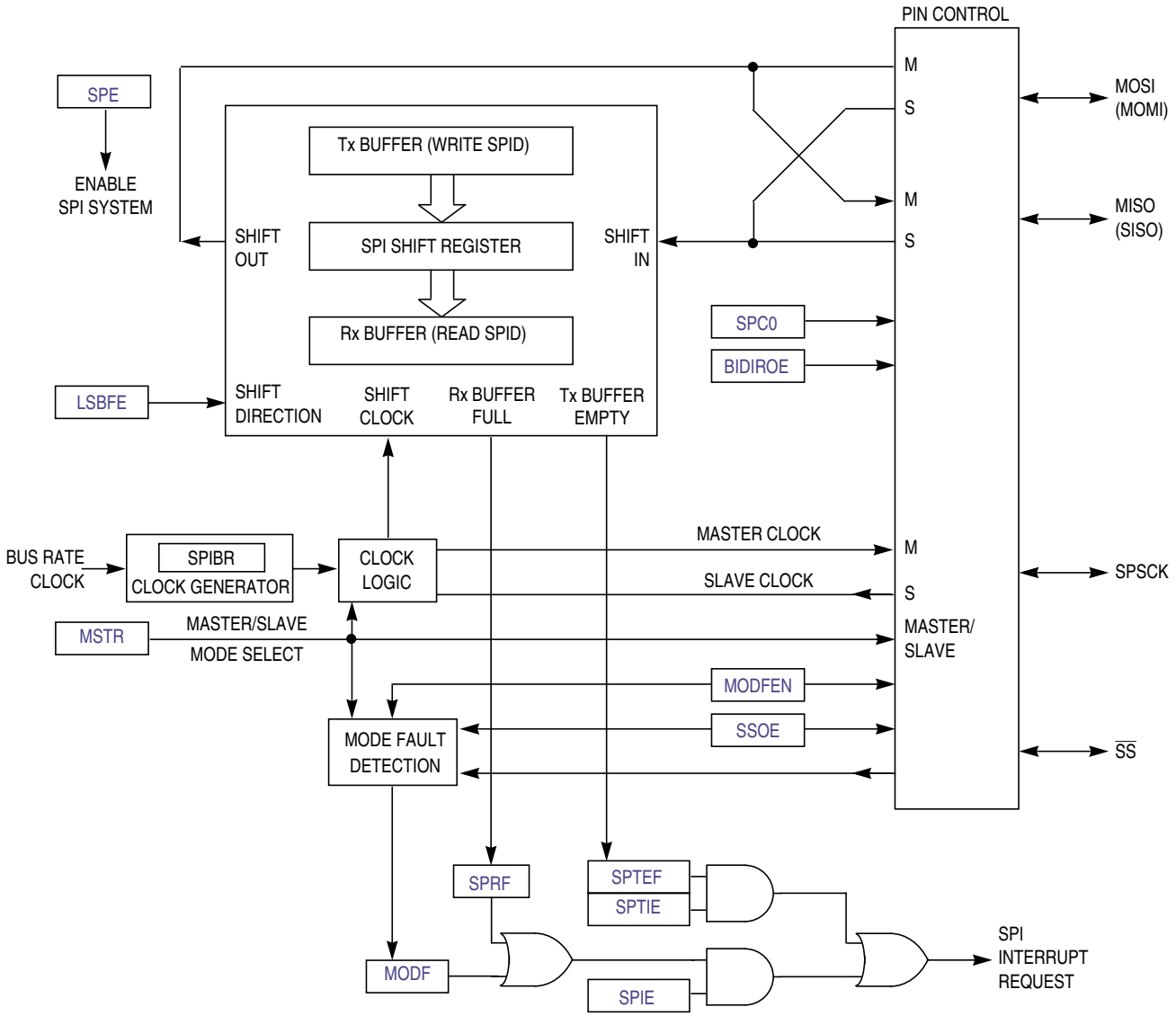



**Figure 13-3. SPI Module Block Diagram**

## 13.1.3 SPI Baud Rate Generation

As shown in Figure 13-4, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI master mode bit-rate clock.

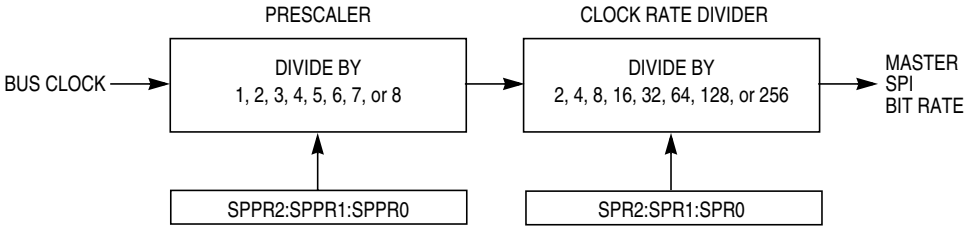



**Figure 13-4. SPI Baud Rate Generation**

## 13.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 13.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 13.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 13.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 13.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

## 13.3 Modes of Operation

### 13.3.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

## 13.4 Register Definition

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 13.4.1 SPI Control Register 1 (SPIC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

| | 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 13-5. SPI Control Register 1 (SPIC1)**

**Table 13-1. SPIC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIE | **SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events.<br>0 Interrupts from SPRF and MODF inhibited (use polling)<br>1 When SPRF or MODF is 1, request a hardware interrupt |
| 6<br>SPE | **SPI System Enable** — Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty.<br>0 SPI system inactive<br>1 SPI system enabled |
| 5<br>SPTIE | **SPI Transmit Interrupt Enable** — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF).<br>0 Interrupts from SPTEF inhibited (use polling)<br>1 When SPTEF is 1, hardware interrupt requested |

**Table 13-1. SPIC1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>MSTR | **Master/Slave Mode Select**<br>0 SPI module configured as a slave SPI device<br>1 SPI module configured as a master SPI device |
| 3<br>CPOL | **Clock Polarity** — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 13.5.1, "SPI Clock Formats" for more details.<br>0 Active-high SPI clock (idles low)<br>1 Active-low SPI clock (idles high) |
| 2<br>CPHA | **Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 13.5.1, "SPI Clock Formats" for more details.<br>0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer<br>1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer |
| 1<br>SSOE | **Slave Select Output Enable** — This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in Table 13-2. |
| 0<br>LSBFE | **LSB First (Shifter Direction)**<br>0 SPI serial data transfers start with most significant bit<br>1 SPI serial data transfers start with least significant bit |

**Table 13-2. $\overline{SS}$ Pin Function**

| MODFEN | SSOE | Master Mode | Slave Mode |
|---|---|---|---|
| 0 | 0 | General-purpose I/O (not SPI) | Slave select input |
| 0 | 1 | General-purpose I/O (not SPI) | Slave select input |
| 1 | 0 | $\overline{SS}$ input for mode fault | Slave select input |
| 1 | 1 | Automatic $\overline{SS}$ output | Slave select input |

**NOTE**

Ensure that the SPI should not be disabled (SPE=0) at the same time as a bit change to the CPHA bit. These changes should be performed as separate operations or unexpected behavior may occur.

## 13.4.2 SPI Control Register 2 (SPIC2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

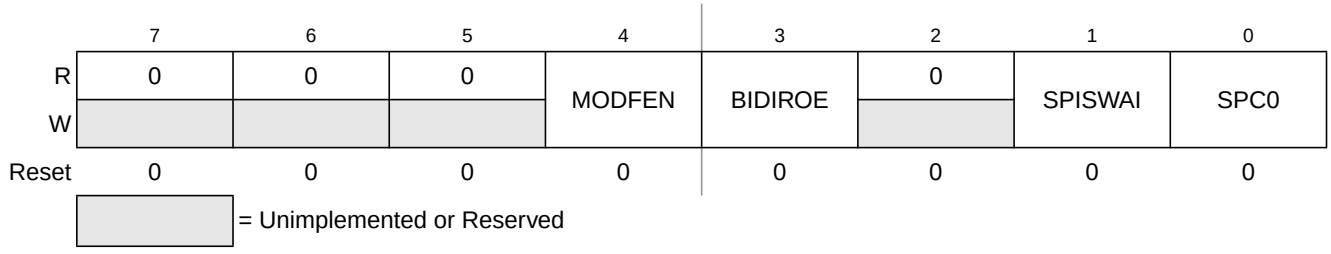



**Figure 13-6. SPI Control Register 2 (SPIC2)**

**Table 13-3. SPIC2 Register Field Descriptions**

| Field | Description |
|---|---|
| 4<br>MODFEN | **Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 13-2 for more details).<br>0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI<br>1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output |
| 3<br>BIDIROE | **Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect.<br>0 Output driver disabled so SPI data I/O pin acts as an input<br>1 SPI I/O pin enabled as an output |
| 1<br>SPISWAI | **SPI Stop in Wait Mode**<br>0 SPI clocks continue to operate in wait mode<br>1 SPI clocks stop when the MCU enters wait mode |
| 0<br>SPC0 | **SPI Pin Control 0** — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin.<br>0 SPI uses separate pins for data input and data output<br>1 SPI configured for single-wire bidirectional operation |

## 13.4.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

| | 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 13-7. SPI Baud Rate Register (SPIBR)**

**Table 13-4. SPIBR Register Field Descriptions**

| Field | Description |
|---|---|
| 6:4<br>SPPR[2:0] | **SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 13-5. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 13-4). |
| 2:0<br>SPR[2:0] | **SPI Baud Rate Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 13-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 13-4). The output of this divider is the SPI bit rate clock for master mode. |

**Table 13-5. SPI Baud Rate Prescaler Divisor**

| SPPR2:SPPR1:SPPR0 | Prescaler Divisor |
|---|---|
| 0:0:0 | 1 |
| 0:0:1 | 2 |
| 0:1:0 | 3 |
| 0:1:1 | 4 |
| 1:0:0 | 5 |
| 1:0:1 | 6 |
| 1:1:0 | 7 |
| 1:1:1 | 8 |

**Table 13-6. SPI Baud Rate Divisor**

| SPR2:SPR1:SPR0 | Rate Divisor |
|---|---|
| 0:0:0 | 2 |
| 0:0:1 | 4 |
| 0:1:0 | 8 |
| 0:1:1 | 16 |
| 1:0:0 | 32 |
| 1:0:1 | 64 |
| 1:1:0 | 128 |
| 1:1:1 | 256 |

## 13.4.4 SPI Status Register (SPIS)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0. Writes have no meaning or effect.

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

= Unimplemented or Reserved

**Figure 13-8. SPI Status Register (SPIS)**

**Table 13-7. SPIS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPRF | **SPI Read Buffer Full Flag** — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPID). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register.<br>0 No data available in the receive data buffer<br>1 Data available in the receive data buffer |
| 5<br>SPTEF | **SPI Transmit Buffer Empty Flag** — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIS with SPTEF set, followed by writing a data value to the transmit buffer at SPID. SPIS must be read with SPTEF = 1 before writing data to SPID or the SPID write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPIC1 is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPID is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter.<br>0 SPI transmit buffer not empty<br>1 SPI transmit buffer empty |
| 4<br>MODF | **Master Mode Fault Flag** — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIC1).<br>0 No mode fault error<br>1 Mode fault error detected |

## 13.4.5 SPI Data Register (SPID)

| | 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 13-9. SPI Data Register (SPID)**

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPID any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

## 13.5 Functional Description

An SPI transfer is initiated by checking for the SPI transmit buffer empty flag (SPTEF = 1) and then writing a byte of data to the SPI data register (SPID) in the master SPI device. When the SPI shift register is available, this byte of data is moved from the transmit data buffer to the shifter, SPTEF is set to indicate there is room in the buffer to queue another transmit character if desired, and the SPI serial transfer starts.

During the SPI transfer, data is sampled (read) on the MISO pin at one SPSCK edge and shifted, changing the bit value on the MOSI pin, one-half SPSCK cycle later. After eight SPSCK cycles, the data that was in the shift register of the master has been shifted out the MOSI pin to the slave while eight bits of data were shifted in the MISO pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading SPID. If another byte of data is waiting in the transmit buffer at the end of a transfer, it is moved into the shifter, SPTEF is set, and a new transfer is started.

Normally, SPI data is transferred most significant bit (MSB) first. If the least significant bit first enable (LSBFE) bit is set, SPI data is shifted LSB first.

When the SPI is configured as a slave, its $\overline{SS}$ pin must be driven low before a transfer starts and $\overline{SS}$ must stay low throughout the transfer. If a clock format where CPHA = 0 is selected, $\overline{SS}$ must be driven to a logic 1 between successive transfers. If CPHA = 1, $\overline{SS}$ may remain low between successive transfers. See Section 13.5.1, "SPI Clock Formats" for more details.

Because the transmitter and receiver are double buffered, a second byte, in addition to the byte currently being shifted out, can be queued into the transmit data buffer, and a previously received character can be in the receive data buffer while a new character is being shifted in. The SPTEF flag indicates when the transmit buffer has room for a new character. The SPRF flag indicates when a received character is available in the receive data buffer. The received character must be read out of the receive buffer (read SPID) before the next transfer is finished or a receive overrun error results.

In the case of a receive overrun, the new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for such an overrun condition so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

### 13.5.1 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 13-10 shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output

pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

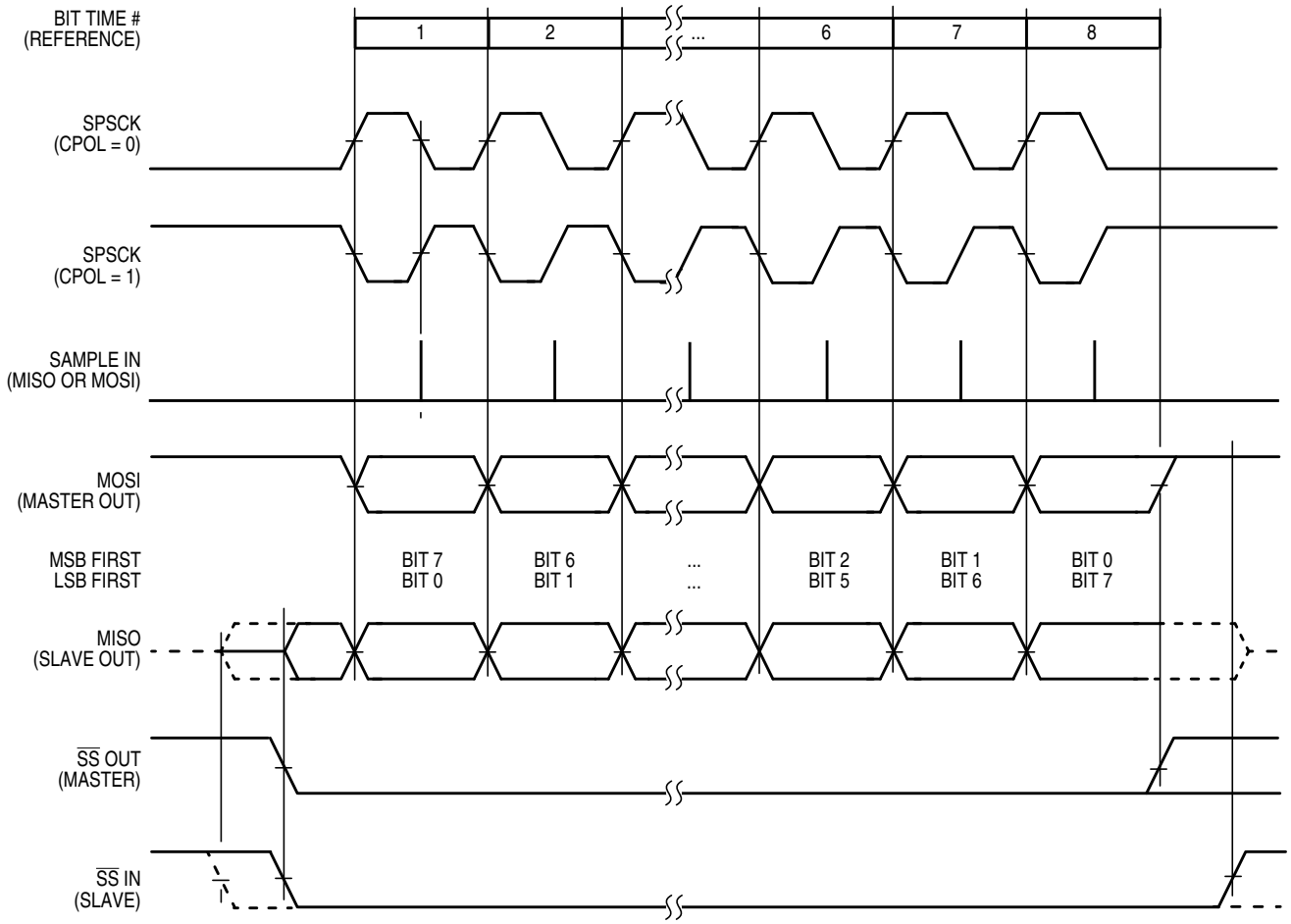



**Figure 13-10. SPI Clock Formats (CPHA = 1)**

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 13-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting

in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

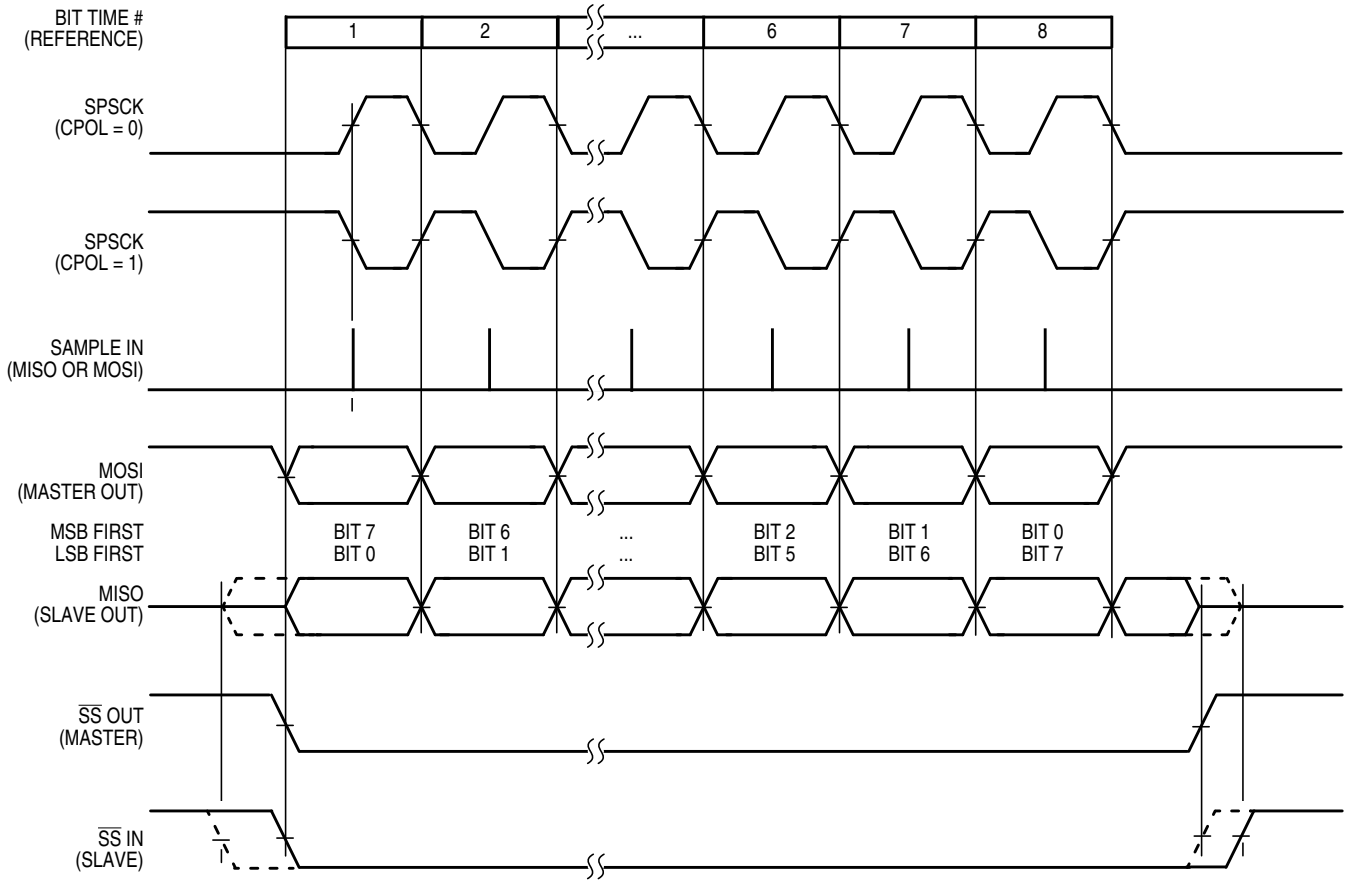



**Figure 13-11. SPI Clock Formats (CPHA = 0)**

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when $\overline{SS}$ goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{SS}$ input must go to its inactive high level between transfers.

### 13.5.2 SPI Interrupts

There are three flag bits, two interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

### 13.5.3 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the $\overline{SS}$ pin (provided the $\overline{SS}$ pin is configured as the mode fault input signal). The $\overline{SS}$ pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's $\overline{SS}$ pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPIC1). User software should verify the error condition has been corrected before changing the SPI back to master mode.

# Chapter 14
# Serial Communications Interface (S08SCIV4)

## 14.1 Introduction

All MCUs in the MC9S08DZ60 Series include SCI1 and SCI2.

**NOTE**

- MC9S08DZ60 Series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Please ignore references to stop1.
- The RxD1 pin does not contain a clamp diode to $V_{DD}$ and should not be driven above $V_{DD}$. The voltage measured on the internally pulled up RxD1 pin may be as low as $V_{DD}$ – 0.7 V. The internal gates connected to this pin are pulled all the way to $V_{DD}$.

### 14.1.1 SCI2 Configuration Information

The SCI2 module pins, TxD2 and RxD2 can be repositioned under software control using SCI2PS in SOPT1 as shown in Table 14-1. SCI2PS in SOPT1 selects which general-purpose I/O ports are associated with SCI2 operation.

**Table 14-1. SCI2 Position Options**

| SCI2PS in SOPT1 | Port Pin for TxD2 | Port Pin for RxD2 |
|---|---|---|
| 0 (default) | PTF0 | PTF1 |
| 1 | PTE6 | PTE7 |

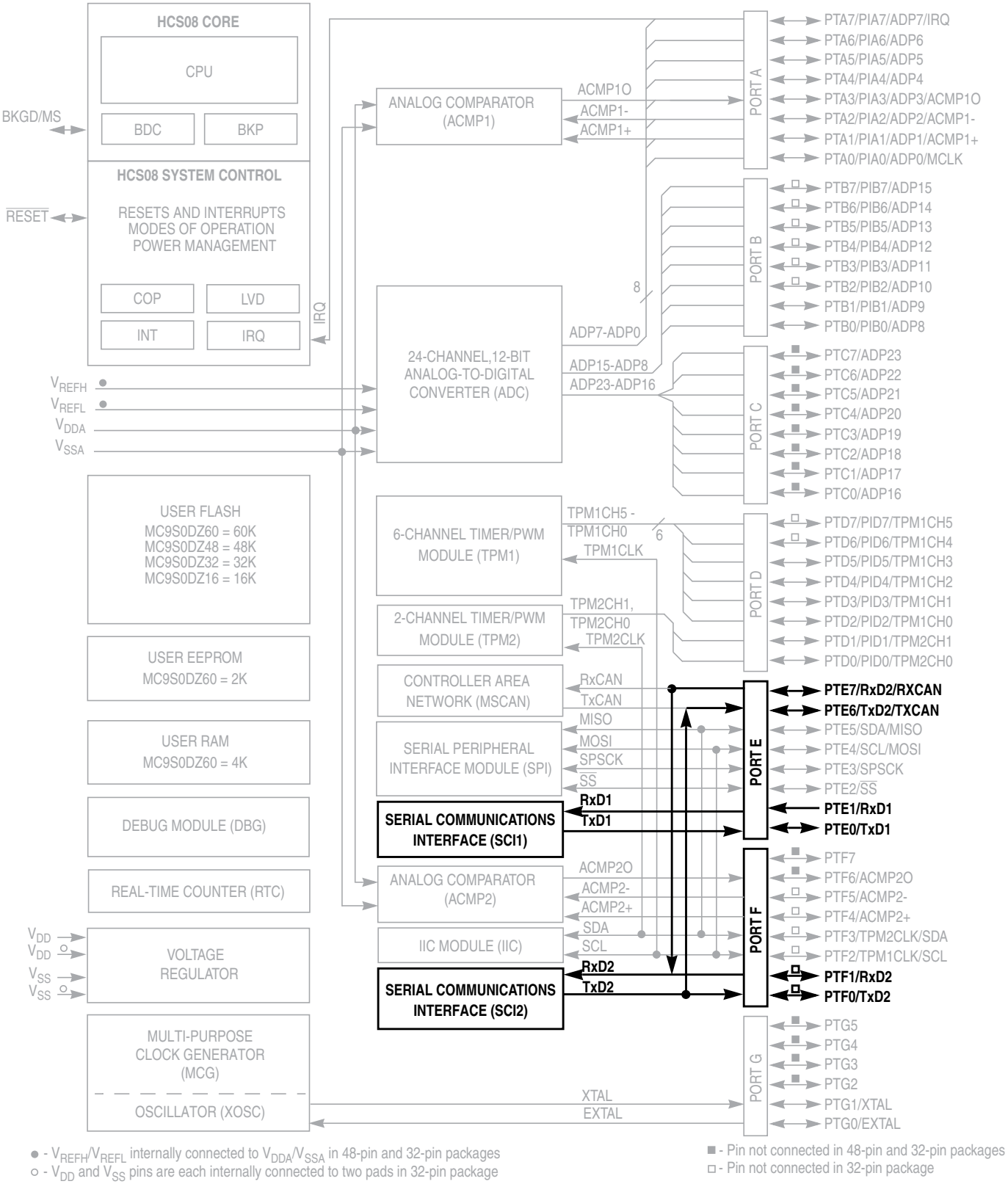



**Figure 14-1. MC9S08DZ60 Block Diagram**

### 14.1.2 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

### 14.1.3 Modes of Operation

See Section 14.3, “Functional Description,” For details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

## 14.1.4 Block Diagram

Figure 14-2 shows the transmitter portion of the SCI.

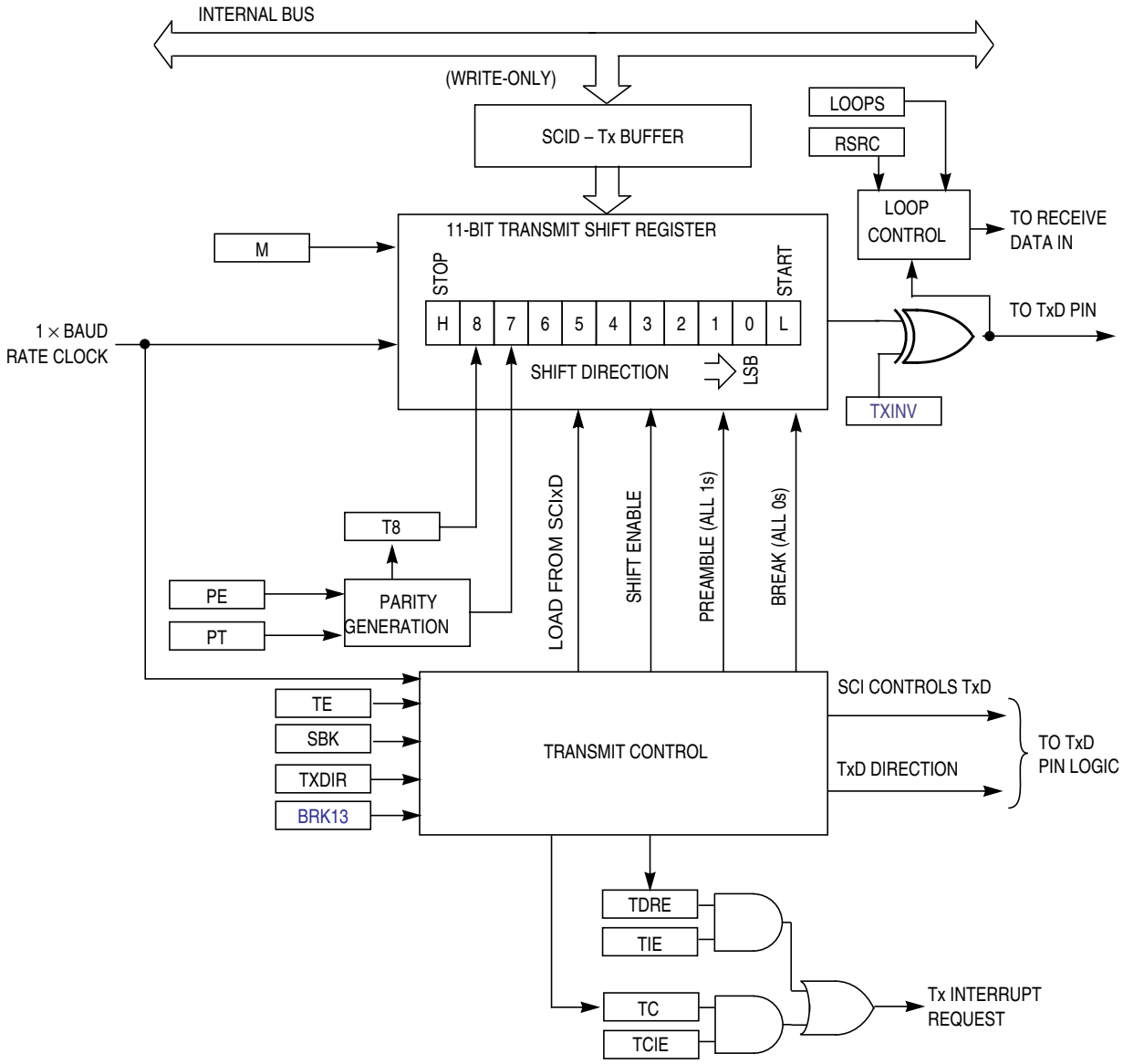



**Figure 14-2. SCI Transmitter Block Diagram**

Figure 14-3 shows the receiver portion of the SCI.

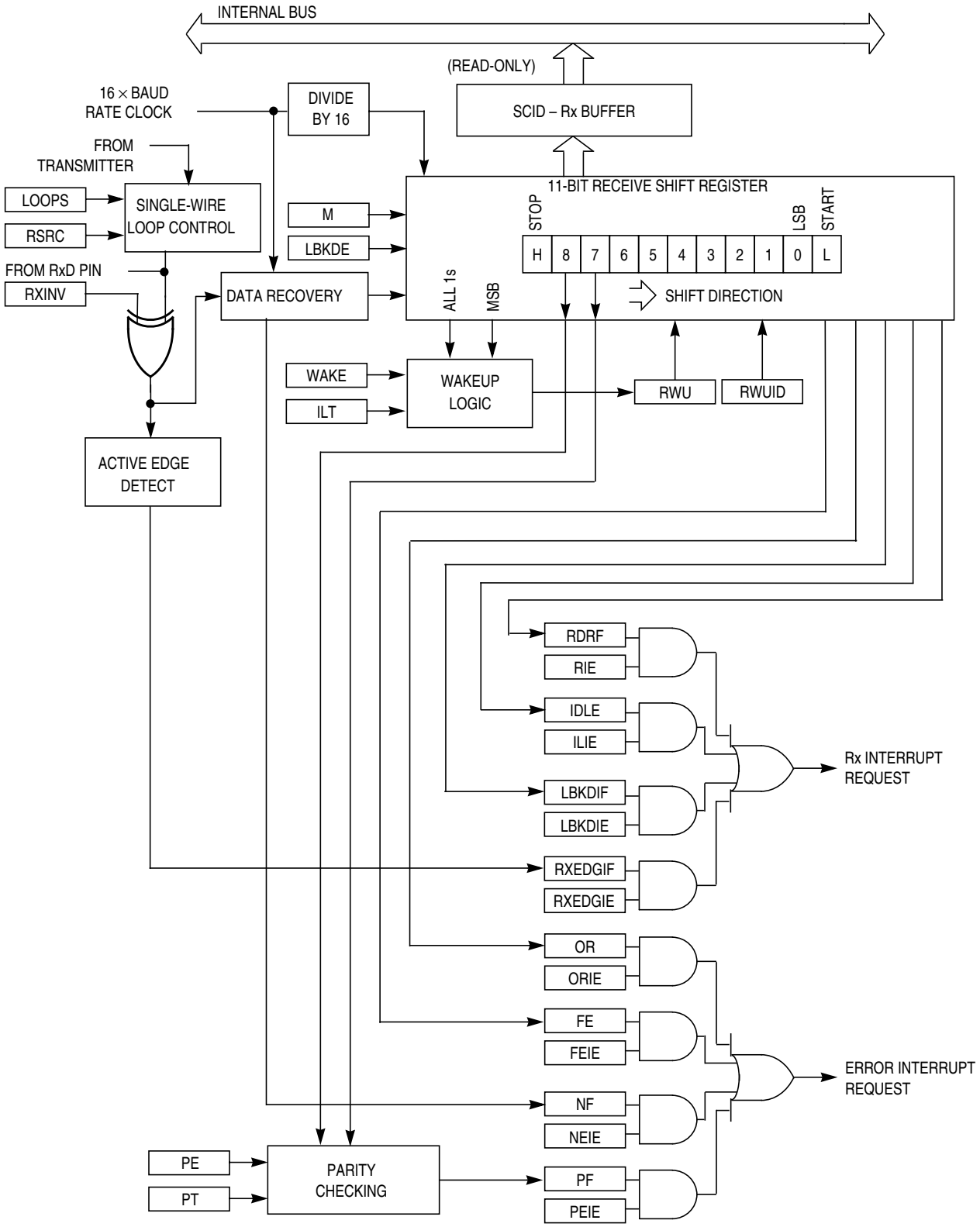



**Figure 14-3. SCI Receiver Block Diagram**

## 14.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 14.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LBKDIE | RXEDGIE | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-4. SCI Baud Rate Register (SCIxBDH)**

**Table 14-2. SCIxBDH Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LBKDIE | **LIN Break Detect Interrupt Enable (for LBKDIF)**<br>0 Hardware interrupts from LBKDIF disabled (use polling).<br>1 Hardware interrupt requested when LBKDIF flag is 1. |
| 6<br>RXEDGIE | **RxD Input Active Edge Interrupt Enable (for RXEDGIF)**<br>0 Hardware interrupts from RXEDGIF disabled (use polling).<br>1 Hardware interrupt requested when RXEDGIF flag is 1. |
| 4:0<br>SBR[12:8] | **Baud Rate Modulo Divisor** — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 14-3. |

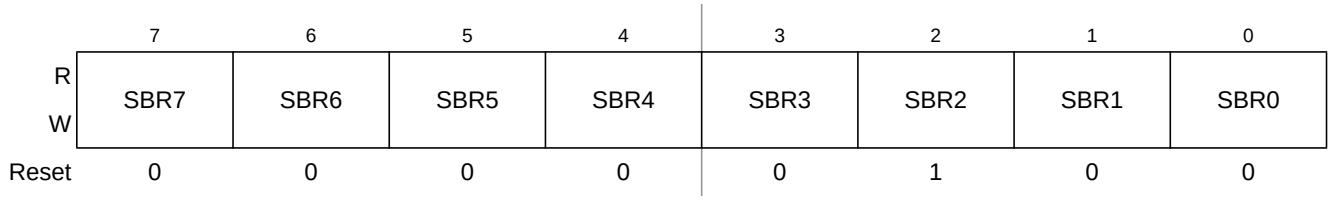



**Figure 14-5. SCI Baud Rate Register (SCIxBDL)**

**Table 14-3. SCIxBDL Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>SBR[7:0] | **Baud Rate Modulo Divisor** — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 14-2. |

## 14.2.2 SCI Control Register 1 (SCIxC1)

This read/write register is used to control various optional features of the SCI system.

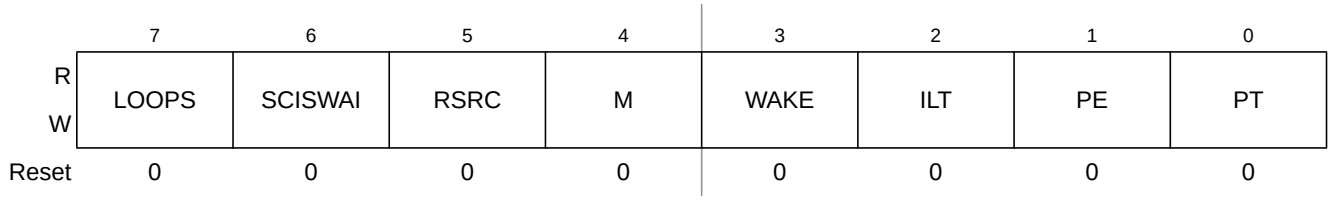



**Figure 14-6. SCI Control Register 1 (SCIxC1)**

**Table 14-4. SCIxC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LOOPS | **Loop Mode Select** — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input.<br>0 Normal operation — RxD and TxD use separate pins.<br>1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI. |
| 6<br>SCISWAI | **SCI Stops in Wait Mode**<br>0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU.<br>1 SCI clocks freeze while CPU is in wait mode. |
| 5<br>RSRC | **Receiver Source Select** — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output.<br>0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins.<br>1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input. |
| 4<br>M | **9-Bit or 8-Bit Mode Select**<br>0 Normal — start + 8 data bits (LSB first) + stop.<br>1 Receiver and transmitter use 9-bit data characters<br>start + 8 data bits (LSB first) + 9th data bit + stop. |

**Table 14-4. SCIxC1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>WAKE | **Receiver Wakeup Method Select** — Refer to Section 14.3.3.2, “Receiver Wakeup Operation” for more information.<br>0 Idle-line wakeup.<br>1 Address-mark wakeup. |
| 2<br>ILT | **Idle Line Type Select** — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 14.3.3.2.1, “Idle-Line Wakeup” for more information.<br>0 Idle character bit count starts after start bit.<br>1 Idle character bit count starts after stop bit. |
| 1<br>PE | **Parity Enable** — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit.<br>0 No hardware parity generation or checking.<br>1 Parity enabled. |
| 0<br>PT | **Parity Type** — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.<br>0 Even parity.<br>1 Odd parity. |

## 14.2.3 SCI Control Register 2 (SCIxC2)

This register can be read or written at any time.

| | 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 14-7. SCI Control Register 2 (SCIxC2)**

**Table 14-5. SCIxC2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TIE | **Transmit Interrupt Enable (for TDRE)**<br>0 Hardware interrupts from TDRE disabled (use polling).<br>1 Hardware interrupt requested when TDRE flag is 1. |
| 6<br>TCIE | **Transmission Complete Interrupt Enable (for TC)**<br>0 Hardware interrupts from TC disabled (use polling).<br>1 Hardware interrupt requested when TC flag is 1. |
| 5<br>RIE | **Receiver Interrupt Enable (for RDRF)**<br>0 Hardware interrupts from RDRF disabled (use polling).<br>1 Hardware interrupt requested when RDRF flag is 1. |
| 4<br>ILIE | **Idle Line Interrupt Enable (for IDLE)**<br>0 Hardware interrupts from IDLE disabled (use polling).<br>1 Hardware interrupt requested when IDLE flag is 1. |

**Table 14-5. SCIxC2 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>TE | **Transmitter Enable**<br>0 Transmitter off.<br>1 Transmitter on.<br>TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system.<br>When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin).<br>TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to Section 14.3.2.1, “Send Break and Queued Idle” for more details.<br>When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin. |
| 2<br>RE | **Receiver Enable** — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1.<br>0 Receiver off.<br>1 Receiver on. |
| 1<br>RWU | **Receiver Wakeup Control** — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 14.3.3.2, “Receiver Wakeup Operation” for more details.<br>0 Normal SCI receiver operation.<br>1 SCI receiver in standby waiting for wakeup condition. |
| 0<br>SBK | **Send Break** — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 14.3.2.1, “Send Break and Queued Idle” for more details.<br>0 Normal transmitter operation.<br>1 Queue break character(s) to be sent. |

## 14.2.4 SCI Status Register 1 (SCIxS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

| | 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 14-8. SCI Status Register 1 (SCIxS1)**

**Table 14-6. SCIxS1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TDRE | **Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE = 1 and then write to the SCI data register (SCIxD).<br>0 Transmit data register (buffer) full.<br>1 Transmit data register (buffer) empty. |
| 6<br>TC | **Transmission Complete Flag** — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.<br>0 Transmitter active (sending data, a preamble, or a break).<br>1 Transmitter idle (transmission activity complete).<br>TC is cleared automatically by reading SCIxS1 with TC = 1 and then doing one of the following three things:<br>• Write to the SCI data register (SCIxD) to transmit new data<br>• Queue a preamble by changing TE from 0 to 1<br>• Queue a break character by writing 1 to SBK in SCIxC2 |
| 5<br>RDRF | **Receive Data Register Full Flag** — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). To clear RDRF, read SCIxS1 with RDRF = 1 and then read the SCI data register (SCIxD).<br>0 Receive data register empty.<br>1 Receive data register full. |
| 4<br>IDLE | **Idle Line Flag** — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, read SCIxS1 with IDLE = 1 and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period.<br>0 No idle line detected.<br>1 Idle line was detected. |
| 3<br>OR | **Receiver Overrun Flag** — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with OR = 1 and then read the SCI data register (SCIxD).<br>0 No overrun.<br>1 Receive overrun (new SCI data lost). |
| 2<br>NF | **Noise Flag** — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD).<br>0 No noise detected.<br>1 Noise detected in the received character in SCIxD. |

**Table 14-6. SCIxS1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FE | **Framing Error Flag** — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIxS1 with FE = 1 and then read the SCI data register (SCIxD).<br>0 No framing error detected. This does not guarantee the framing is correct.<br>1 Framing error. |
| 0<br>PF | **Parity Error Flag** — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIxS1 and then read the SCI data register (SCIxD).<br>0 No parity error.<br>1 Parity error. |

## 14.2.5 SCI Status Register 2 (SCIxS2)

This register has one read-only status flag.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LBKDIF | RXEDGIF | 0 | RXINV | RWUID | BRK13 | LBKDE | RAF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-9. SCI Status Register 2 (SCIxS2)**

**Table 14-7. SCIxS2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LBKDIF | **LIN Break Detect Interrupt Flag** — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it.<br>0 No LIN break character has been detected.<br>1 LIN break character has been detected. |
| 6<br>RXEDGIF | **RxD Pin Active Edge Interrupt Flag** — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it.<br>0 No active edge on the receive pin has occurred.<br>1 An active edge on the receive pin has occurred. |
| 4<br>RXINV[1] | **Receive Data Inversion** — Setting this bit reverses the polarity of the received data input.<br>0 Receive data not inverted<br>1 Receive data inverted |
| 3<br>RWUID | **Receive Wake Up Idle Detect**— RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit.<br>0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character.<br>1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. |
| 2<br>BRK13 | **Break Character Generation Length** — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit.<br>0 Break character is transmitted with length of 10 bit times (11 if M = 1)<br>1 Break character is transmitted with length of 13 bit times (14 if M = 1) |

**Table 14-7. SCIxS2 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>LBKDE | **LIN Break Detection Enable**— LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting.<br>0 Break character is detected at length of 10 bit times (11 if M = 1).<br>1 Break character is detected at length of 11 bit times (12 if M = 1). |
| 0<br>RAF | **Receiver Active Flag** — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode.<br>0 SCI receiver idle waiting for a start bit.<br>1 SCI receiver active (RxD input not idle). |

[1] Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

## 14.2.6 SCI Control Register 3 (SCIxC3)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R8 | T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-10. SCI Control Register 3 (SCIxC3)**

**Table 14-8. SCIxC3 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>R8 | **Ninth Data Bit for Receiver** — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCIxD register. When reading 9-bit data, read R8 before reading SCIxD because reading SCIxD completes automatic flag clearing sequences which could allow R8 and SCIxD to be overwritten with new data. |
| 6<br>T8 | **Ninth Data Bit for Transmitter** — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written. |
| 5<br>TXDIR | **TxD Pin Direction in Single-Wire Mode** — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin.<br>0 TxD pin is an input in single-wire mode.<br>1 TxD pin is an output in single-wire mode. |

**Table 14-8. SCIxC3 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>TXINV[1] | **Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output.<br>0 Transmit data not inverted<br>1 Transmit data inverted |
| 3<br>ORIE | **Overrun Interrupt Enable** — This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0 OR interrupts disabled (use polling).<br>1 Hardware interrupt requested when OR = 1. |
| 2<br>NEIE | **Noise Error Interrupt Enable** — This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0 NF interrupts disabled (use polling).<br>1 Hardware interrupt requested when NF = 1. |
| 1<br>FEIE | **Framing Error Interrupt Enable** — This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0 FE interrupts disabled (use polling).<br>1 Hardware interrupt requested when FE = 1. |
| 0<br>PEIE | **Parity Error Interrupt Enable** — This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0 PF interrupts disabled (use polling).<br>1 Hardware interrupt requested when PF = 1. |

1 Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

## 14.2.7 SCI Data Register (SCIxD)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

| | 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 14-11. SCI Data Register (SCIxD)**

# 14.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

## 14.3.1 Baud Rate Generation

As shown in Figure 14-12, the clock source for the SCI baud rate generator is the bus-rate clock.

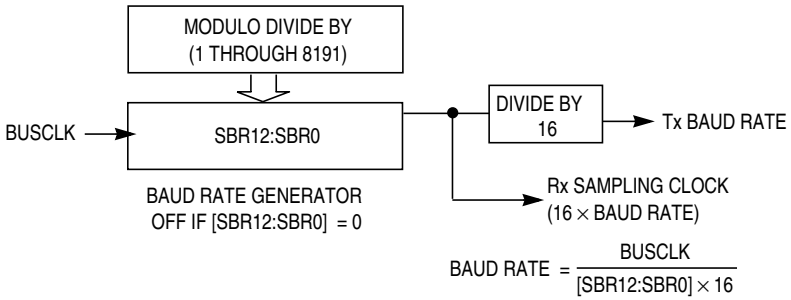



**Figure 14-12. SCI Baud Rate Generation**

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about 4.5percent for 8-bit data format and about 4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

## 14.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in Figure 14-2.

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIxC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

#### 14.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

**Table 14-9. Break Character Length**

| BRK13 | M | Break Character Length |
|---|---|---|
| 0 | 0 | 10 bit times |
| 0 | 1 | 11 bit times |
| 1 | 0 | 13 bit times |
| 1 | 1 | 14 bit times |

### 14.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 14-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 14.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status

flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to Section 14.3.4, "Interrupts and Status Flags" for more details about flag clearing.

#### 14.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

#### 14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant

message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

#### 14.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

#### 14.3.3.2.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

### 14.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while RDRF = 1 and then reading SCIxD.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while IDLE = 1 and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled (RE = 1).

## 14.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

### 14.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCIxD.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

#### 14.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2.. An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

#### 14.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

#### 14.3.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

# Chapter 15
# Real-Time Counter (S08RTCV1)

## 15.1 Introduction

The RTC module consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, three clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake up from low power modes without the need of external components.

All devices in the MC9S08DZ60 Series feature the RTC.

### 15.1.1 RTC Clock Signal Names

References to ERCLK and IRCLK in this chapter correspond to signals MCGERCLK and MCGIRCLK, respectively.

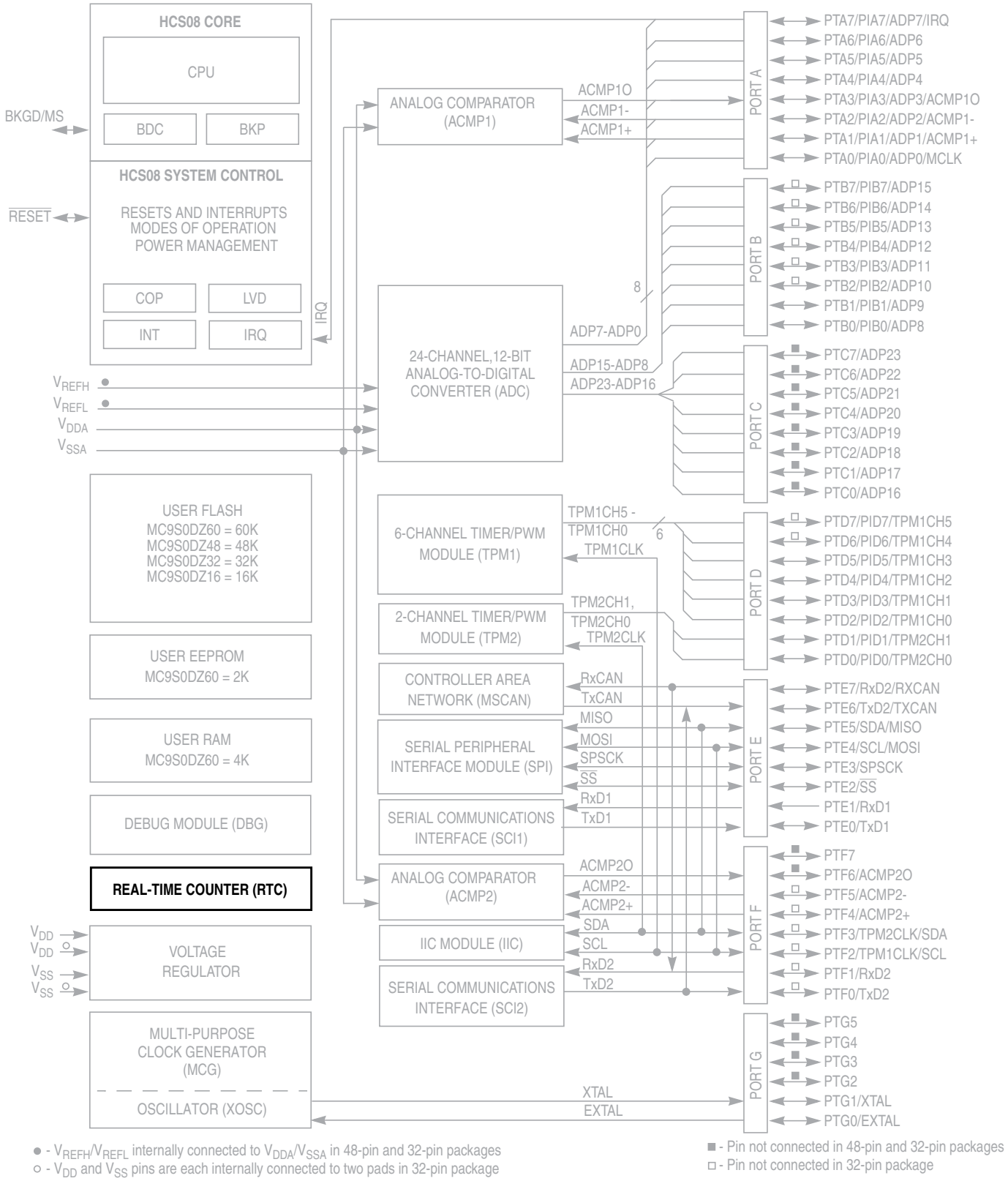



**Figure 15-1. MC9S08DZ60 Block Diagram**

### 15.1.2 Features

Features of the RTC module include:

- 8-bit up-counter
  - 8-bit modulo match limit
  - Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler with selectable binary-based and decimal-based divider values
  - 1-kHz internal low-power oscillator (LPO)
  - External clock (ERCLK)
  - 32-kHz internal clock (IRCLK)

### 15.1.3 Modes of Operation

This section defines the operation in stop, wait and background debug modes.

#### 15.1.3.1 Wait Mode

The RTC continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the RTC can 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.

#### 15.1.3.2 Stop Modes

The RTC continues to run in stop2 or stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

The LPO clock can be used in stop2 and stop3 modes. ERCLK and IRCLK clocks are only available in stop3 mode.

Power consumption is lower when all clock sources are disabled, but in that case, the real-time interrupt cannot wake up the MCU from stop modes.

#### 15.1.3.3 Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

### 15.1.4 Block Diagram

The block diagram for the RTC module is shown in Figure 15-2.

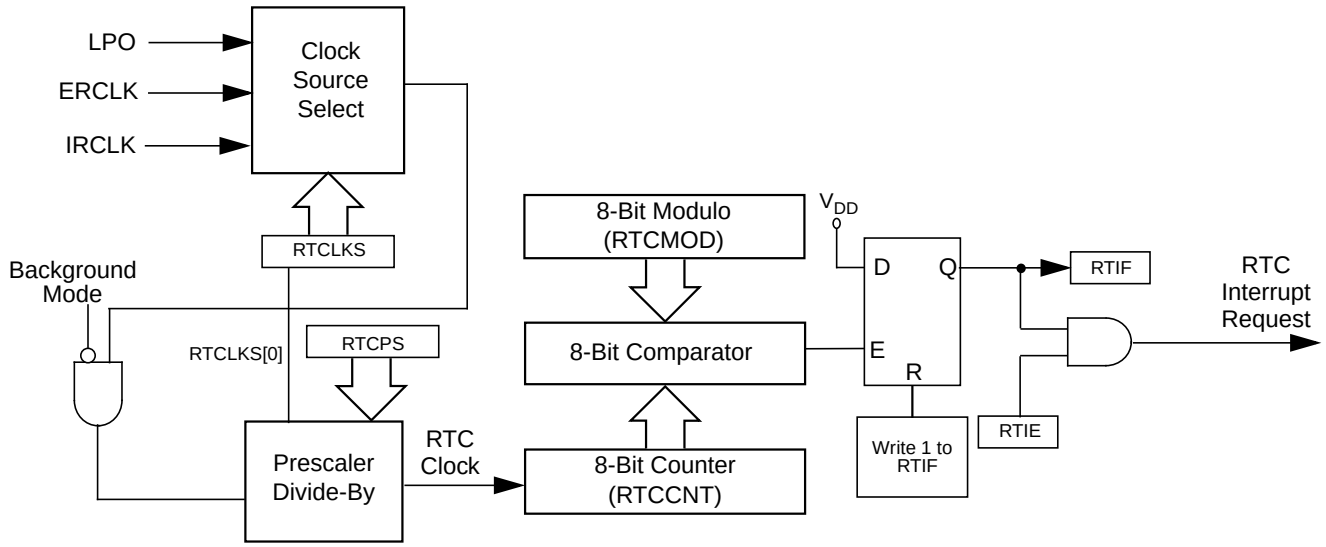



**Figure 15-2. Real-Time Counter (RTC) Block Diagram**

## 15.2 External Signal Description

The RTC does not include any off-chip signals.

## 15.3 Register Definition

The RTC includes a status and control register, an 8-bit counter register, and an 8-bit modulo register.

Refer to the direct-page register summary in the memory section of this document for the absolute address assignments for all RTC registers.This section refers to registers and control bits only by their names and relative address offsets.

Table 15-1 is a summary of RTC registers.

**Table 15-1. RTC Register Summary**

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| RTCSC | R | RTIF | RTCLKS | | RTIE | RTCPS | | | |
| | W | | | | | | | | |
| RTCCNT | R | RTCCNT | | | | | | | |
| | W | | | | | | | | |
| RTCMOD | R | RTCMOD | | | | | | | |
| | W | | | | | | | | |

### 15.3.1 RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).

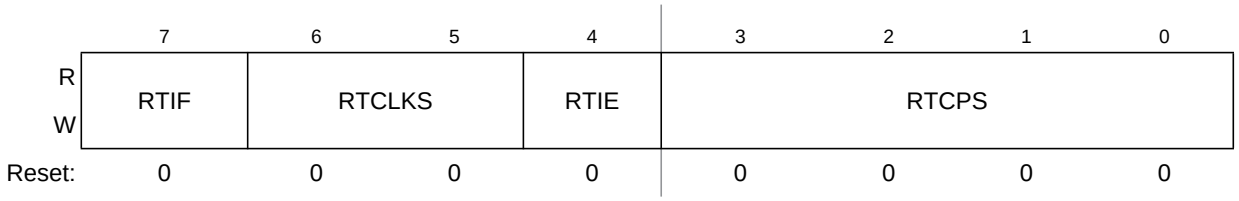



**Figure 15-3. RTC Status and Control Register (RTCSC)**

**Table 15-2. RTCSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIF | Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF.<br>0 RTC counter has not reached the value in the RTC modulo register.<br>1 RTC counter has reached the value in the RTC modulo register. |
| 6–5<br>RTCLKS | Real-Time Clock Source Select. These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. When selecting a clock source, ensure that the clock source is properly enabled (if applicable) to ensure correct operation of the RTC. Reset clears RTCLKS.<br>00 Real-time clock source is the 1-kHz low power oscillator (LPO)<br>01 Real-time clock source is the external clock (ERCLK)<br>1x Real-time clock source is the internal clock (IRCLK) |
| 4<br>RTIE | Real-Time Interrupt Enable. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE.<br>0 Real-time interrupt requests are disabled. Use software polling.<br>1 Real-time interrupt requests are enabled. |
| 3–0<br>RTCPS | Real-Time Clock Prescaler Select. These four read/write bits select binary-based or decimal-based divide-by values for the clock source. See Table 15-3. Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS. |

**Table 15-3. RTC Prescaler Divide-by values**

| RTCLKS[0] | RTCPS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **0** | Off | $2^3$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ | $2^{10}$ | 1 | 2 | $2^2$ | 10 | $2^4$ | $10^2$ | $5x10^2$ | $10^3$ |
| **1** | Off | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{13}$ | $2^{14}$ | $2^{15}$ | $2^{16}$ | $10^3$ | $2x10^3$ | $5x10^3$ | $10^4$ | $2x10^4$ | $5x10^4$ | $10^5$ | $2x10^5$ |

### 15.3.2 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

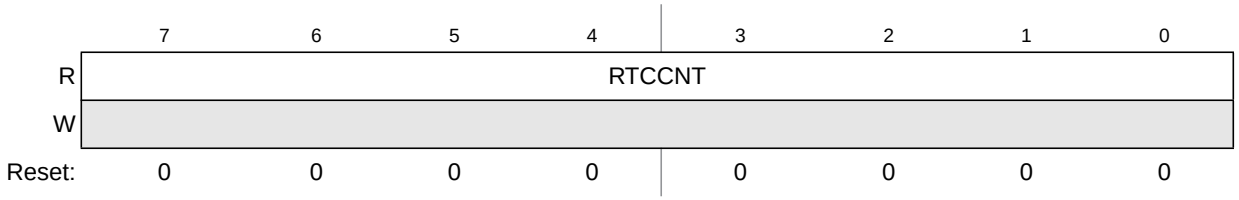



**Figure 15-4. RTC Counter Register (RTCCNT)**

**Table 15-4. RTCCNT Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>RTCCNT | RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00. |

### 15.3.3 RTC Modulo Register (RTCMOD)

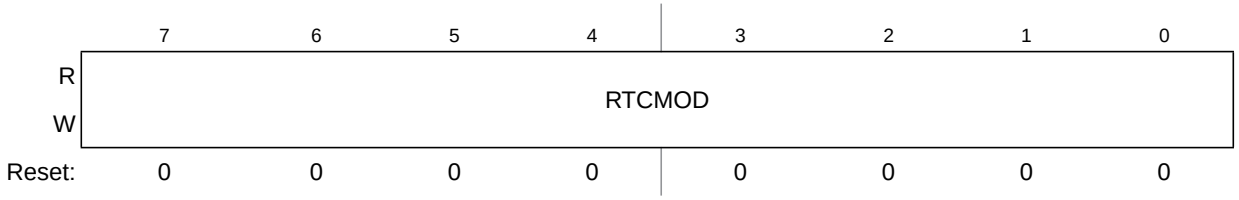



**Figure 15-5. RTC Modulo Register (RTCMOD)**

**Table 15-5. RTCMOD Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>RTCMOD | RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00. |

## 15.4 Functional Description

The RTC is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

RTCPS and the RTCLKS[0] bit select the desired divide-by value. If a different value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. Table 15-6 shows different prescaler period values.

**Table 15-6. Prescaler Period**

| RTCPS | 1-kHz Internal Clock (RTCLKS = 00) | 1-MHz External Clock (RTCLKS = 01) | 32-kHz Internal Clock (RTCLKS = 10) | 32-kHz Internal Clock (RTCLKS = 11) |
|---|---|---|---|---|
| 0000 | Off | Off | Off | Off |
| 0001 | 8 ms | 1.024 ms | 250 µs | 32 ms |
| 0010 | 32 ms | 2.048 ms | 1 ms | 64 ms |
| 0011 | 64 ms | 4.096 ms | 2 ms | 128 ms |
| 0100 | 128 ms | 8.192 ms | 4 ms | 256 ms |
| 0101 | 256 ms | 16.4 ms | 8 ms | 512 ms |
| 0110 | 512 ms | 32.8 ms | 16 ms | 1.024 s |
| 0111 | 1.024 s | 65.5 ms | 32 ms | 2.048 s |
| 1000 | 1 ms | 1 ms | 31.25 µs | 31.25 ms |
| 1001 | 2 ms | 2 ms | 62.5 µs | 62.5 ms |
| 1010 | 4 ms | 5 ms | 125 µs | 156.25 ms |
| 1011 | 10 ms | 10 ms | 312.5 µs | 312.5 ms |
| 1100 | 16 ms | 20 ms | 0.5 ms | 0.625 s |
| 1101 | 0.1 s | 50 ms | 3.125 ms | 1.5625 s |
| 1110 | 0.5 s | 0.1 s | 15.625 ms | 3.125 s |
| 1111 | 1 s | 0.2 s | 31.25 ms | 6.25 s |

The RTC modulo register (RTCMOD) allows the compare value to be set to any value from 0x00 to 0xFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x00 and continues counting. The real-time interrupt flag (RTIF) is set when a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00.

The RTC allows for an interrupt to be generated when RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

### 15.4.1 RTC Operation Example

This section shows an example of the RTC operation as the counter reaches a matching value from the modulo register.

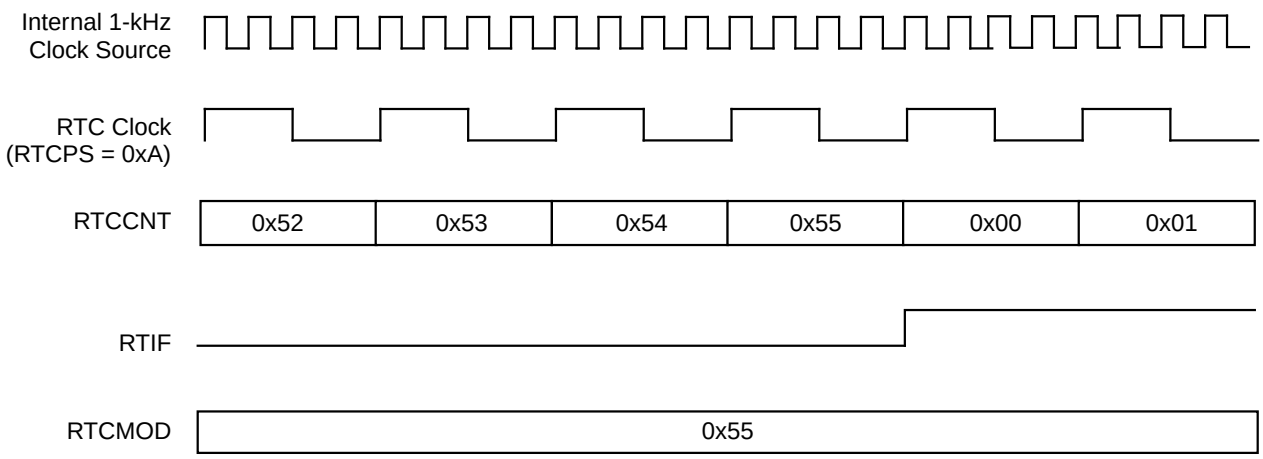



**Figure 15-6. RTC Counter Overflow Example**

In the example of Figure 15-6, the selected clock source is the 1-kHz internal oscillator clock source. The prescaler (RTCPS) is set to 0xA or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

## 15.5 Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;

/*********************************************************************
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*********************************************************************/
```

```
#pragma TRAP_PROC
void RTC_ISR(void)
{
        /* Clear the interrupt flag */
        RTCSC.byte = RTCSC.byte | 0x80;
        /* RTC interrupts every 1 Second */
        Seconds++;
        /* 60 seconds in a minute */
        if (Seconds > 59){
        Minutes++;
        Seconds = 0;
        }
        /* 60 minutes in an hour */
        if (Minutes > 59){
        Hours++;
        Minutes = 0;
        }
        /* 24 hours in a day */
        if (Hours > 23){
        Days ++;
        Hours = 0;
        }
```

# Chapter 16
# Timer Pulse-Width Modulator (S08TPMV3)

**NOTE**

This chapter refers to S08TPM version 3, which applies to the 0M74K and newer mask sets of this device. 3M05C and older mask set devices use S08TPM version 2. If your device uses mask 3M05C or older, please refer to Appendix B, “Timer Pulse-Width Modulator (TPMV2) on page 391 for information pertaining to that module.

## 16.1 Introduction

The TPM is a one-to-eight-channel timer system which supports traditional input capture, output compare, or edge-aligned PWM on each channel. A control bit allows the TPM to be configured such that all channels may be used for center-aligned PWM functions. Timing functions are based on a 16-bit counter with prescaler and modulo features to control frequency and range (period between overflows) of the time reference. This timing system is ideally suited for a wide range of control applications, and the center-aligned PWM capability extends the field of application to motor control in small appliances.

The TPM uses one input/output (I/O) pin per channel, TPMxCHn, where x is the TPM number (for example, 1 or 2) and n is the channel number (for example, 0–5). The TPM shares its I/O pins with general-purpose I/O port pins (refer to the Pins and Connections chapter for more information).

MC9S08DZ60 Series MCUs have two TPM modules. In all packages, TPM2 is 2-channel. The number of channels available on external pins in TPM1 depends on the package:

- Six channels in 64-pin and 48-pin packages
- Four channels in 32-pin packages.

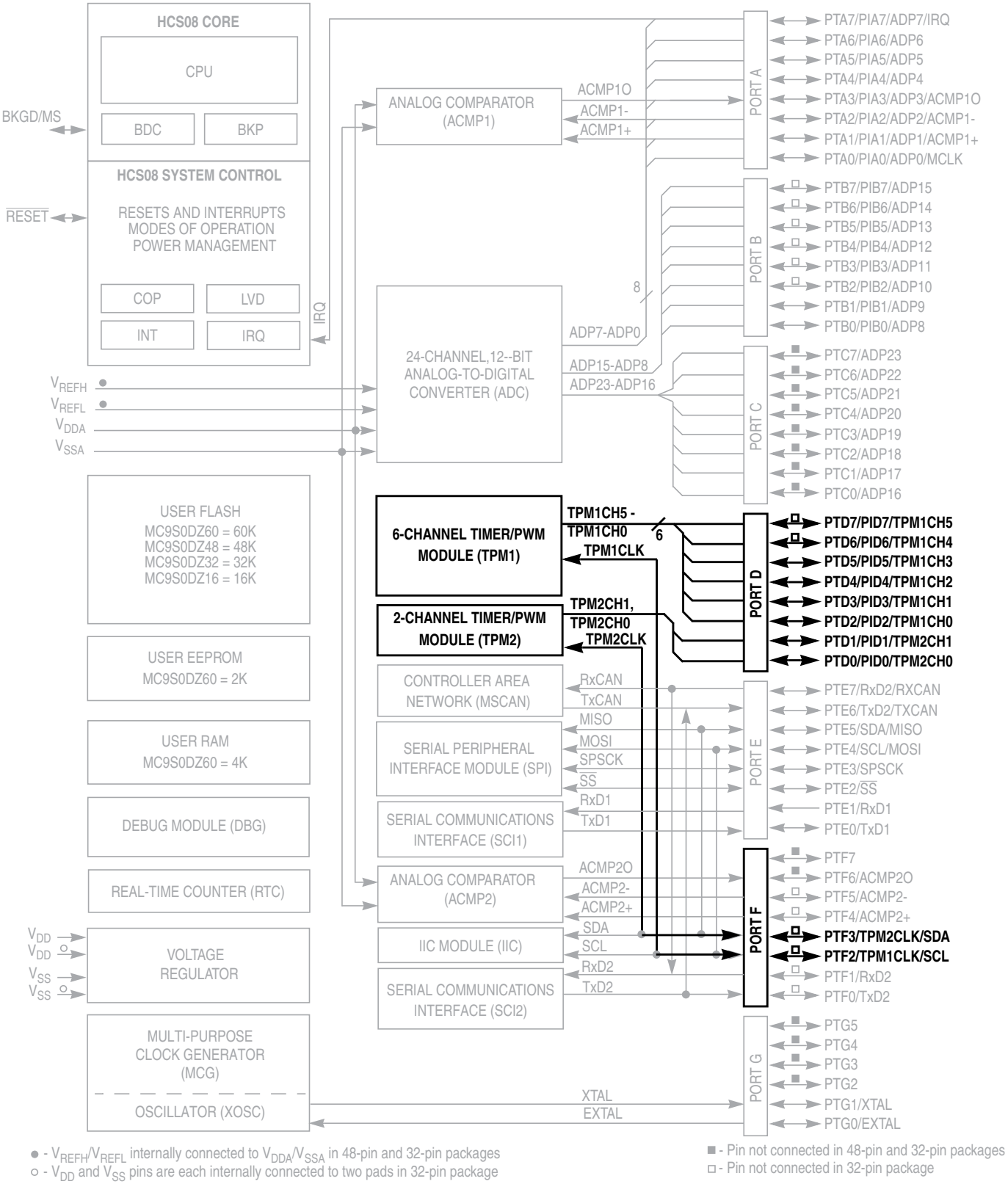



**Figure 16-1. MC9S08DZ60 Block Diagram**

### 16.1.1 Features

The TPM includes these distinctive features:

- One to eight channels:
  - Each channel may be input capture, output compare, or edge-aligned PWM
  - Rising-Edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
  - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
  - Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
  - External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

### 16.1.2 Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- Input capture mode

  When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- Output compare mode

  When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

- Edge-aligned PWM mode

  The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- Center-aligned PWM mode

  Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

### 16.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n) where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 16-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.

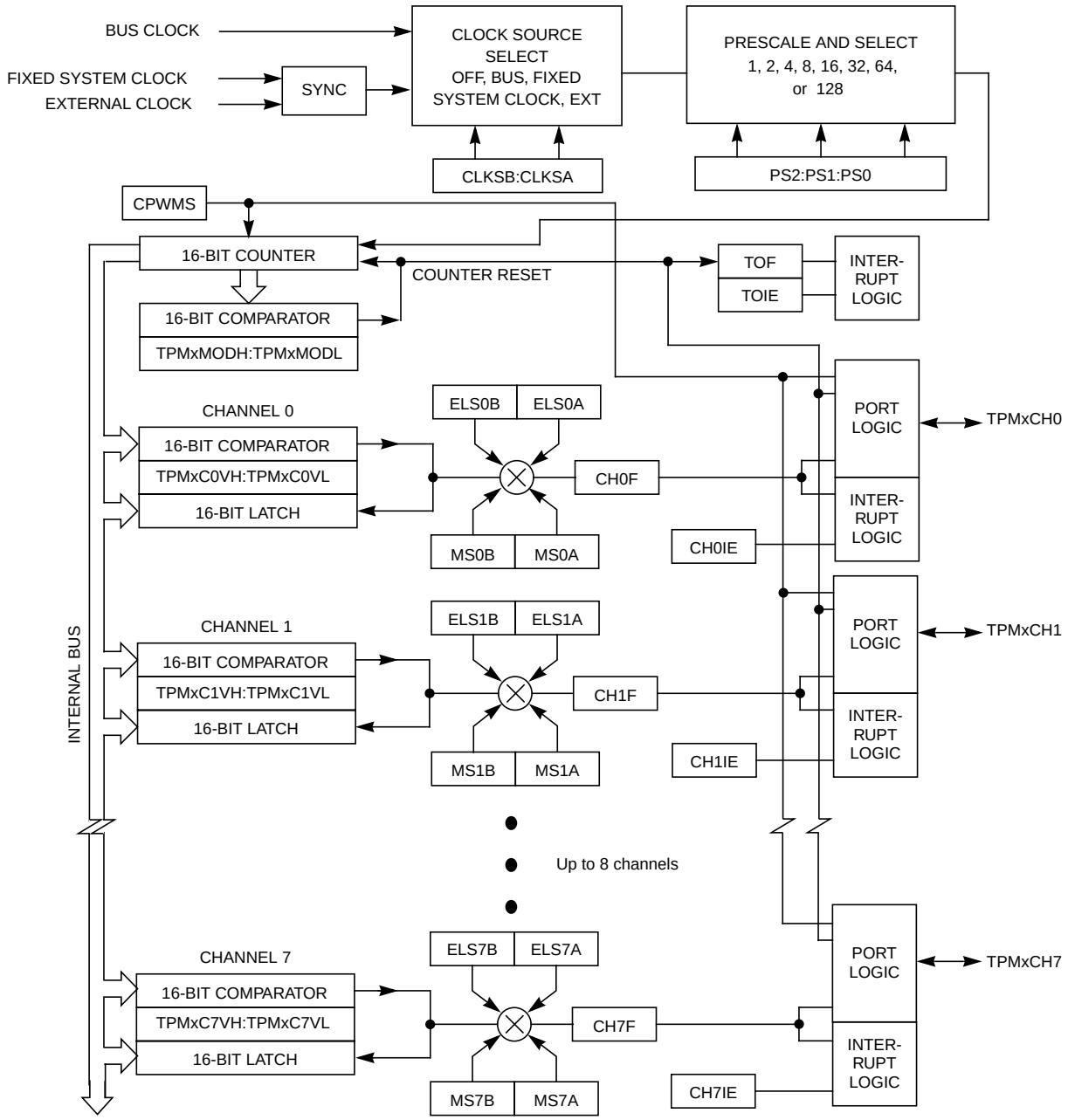



**Figure 16-2. TPM Block Diagram**

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

## 16.2 Signal Description

Table 16-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

**Table 16-1. Signal Properties**

| Name | Function |
|---|---|
| EXTCLK[1] | External clock source which may be selected to drive the TPM counter. |
| TPMxCHn[2] | I/O pin associated with TPM channel n |

1 When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

2 n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

### 16.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 16-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

#### 16.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

#### 16.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMxCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMxCHn pin is forced low at the start of each new period (TPMxCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

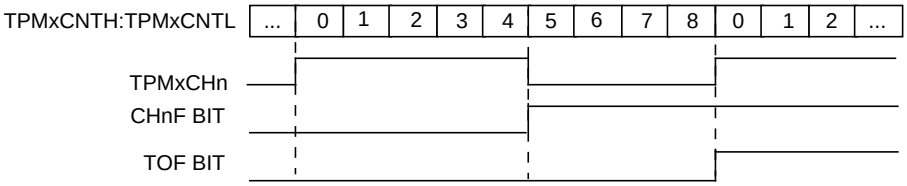



**Figure 16-3. High-True Pulse of an Edge-Aligned PWM**

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

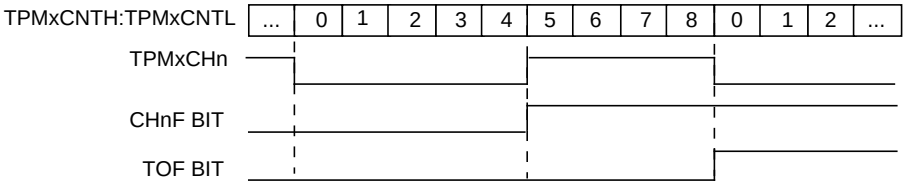



**Figure 16-4. Low-True Pulse of an Edge-Aligned PWM**

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMxCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMxCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMxCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMxCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMxCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMxCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

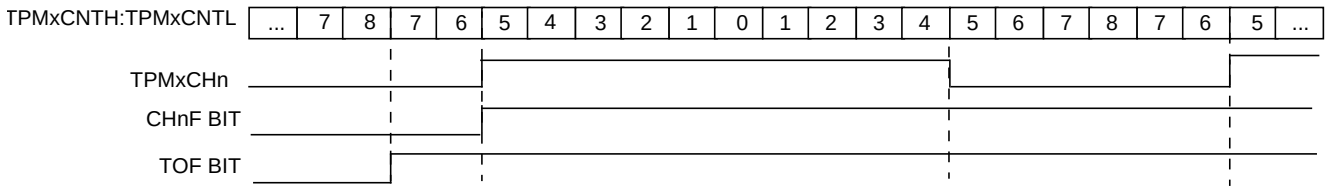



**Figure 16-5. High-True Pulse of a Center-Aligned PWM**

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

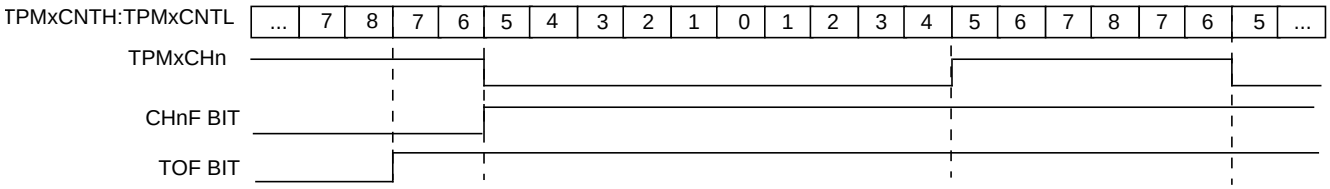



**Figure 16-6. Low-True Pulse of a Center-Aligned PWM**

## 16.3 Register Definition

This section consists of register descriptions in address order. A typical MCU system may contain multiple TPMs, and each TPM may have one to eight channels, so register names include placeholder characters to identify which TPM and which channel is being referenced. For example, TPMxCnSC refers to timer (TPM) x, channel n. TPM1C2SC would be the status and control register for channel 2 of timer 1.

### 16.3.1 TPM Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| W | 0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-7. TPM Status and Control Register (TPMxSC)**

**Table 16-2. TPMxSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect.<br>0 TPM counter has not reached modulo value or overflow<br>1 TPM counter has overflowed |
| 6<br>TOIE | Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE.<br>0 TOF interrupts inhibited (use for software polling)<br>1 TOF interrupts enabled |
| 5<br>CPWMS | Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS.<br>0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register.<br>1 All channels operate in center-aligned PWM mode. |

**Table 16-2. TPMxSC Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4–3<br>CLKS[B:A] | Clock source selects. As shown in Table 16-3, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based or FLL-based system clock. When there is no PLL or FLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL or FLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL or FLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock. |
| 2–0<br>PS[2:0] | Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 16-4. This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits. |

**Table 16-3. TPM-Clock-Source Selection**

| CLKSB:CLKSA | TPM Clock Source to Prescaler Input |
|---|---|
| 00 | No clock selected (TPM counter disable) |
| 01 | Bus rate clock |
| 10 | Fixed system clock |
| 11 | External source |

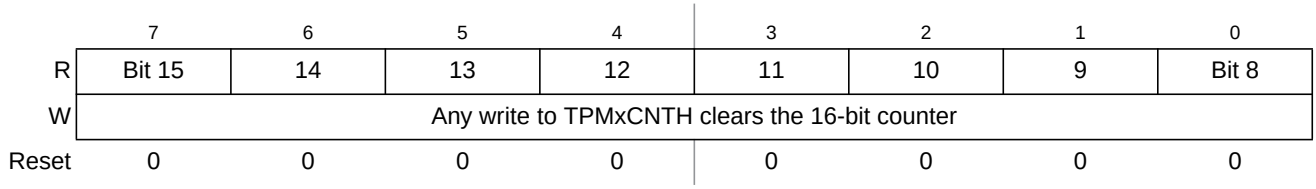
**Table 16-4. Prescale Factor Selection**

| PS2:PS1:PS0 | TPM Clock Source Divided-by |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 4 |
| 011 | 8 |
| 100 | 16 |
| 101 | 32 |
| 110 | 64 |
| 111 | 128 |

## 16.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL)

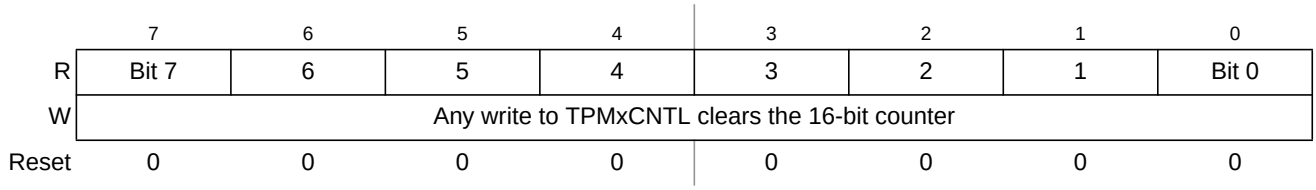
The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers. Writing any value to TPMxCNTH or TPMxCNTL also clears the TPM counter (TPMxCNTH:TPMxCNTL) and resets the coherency mechanism, regardless of the data involved in the write.



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| W | Any write to TPMxCNTH clears the 16-bit counter | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-8. TPM Counter Register High (TPMxCNTH)**



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| W | Any write to TPMxCNTL clears the 16-bit counter | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-9. TPM Counter Register Low (TPMxCNTL)**

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPMxSC, TPMxCNTH or TPMxCNTL registers resets the read coherency mechanism of the TPMxCNTH:L registers, regardless of the data involved in the write.

### 16.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMxMODH or TPMxMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMxSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-10. TPM Counter Modulo Register High (TPMxMODH)**

| | 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 16-11. TPM Counter Modulo Register Low (TPMxMODL)**

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

## 16.3.4 TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CHnF | CHnIE | MSnB | MSnA | ELSnB | ELSnA | 0 | 0 |
| W | 0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-12. TPM Channel n Status and Control Register (TPMxCnSC)**

**Table 16-5. TPMxCnSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CHnF | Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers.<br>A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF.<br>Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect.<br>0 No input capture or output compare event occurred on channel n<br>1 Input capture or output compare event on channel n |
| 6<br>CHnIE | Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE.<br>0 Channel n interrupt requests disabled (use for software polling)<br>1 Channel n interrupt requests enabled |
| 5<br>MSnB | Mode select B for TPM channel n. When CPWMS=0, MSnB=1 configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 16-6. |
| 4<br>MSnA | Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 16-6 for a summary of channel mode and setup controls.<br>**Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger. |
| 3–2<br>ELSnB<br>ELSnA | Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 16-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output.<br>Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin. |

**Table 16-6. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| X | XX | 00 | Pin not used for TPM - revert to general purpose I/O or other peripheral control | |

**Table 16-6. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| 0 | 00 | 01 | Input capture | Capture on rising edge only |
| | | 10 | | Capture on falling edge only |
| | | 11 | | Capture on rising or falling edge |
| | 01 | 01 | Output compare | Toggle output on compare |
| | | 10 | | Clear output on compare |
| | | 11 | | Set output on compare |
| | 1X | 10 | Edge-aligned PWM | High-true pulses (clear output on compare) |
| | | X1 | | Low-true pulses (set output on compare) |
| 1 | XX | 10 | Center-aligned PWM | High-true pulses (clear output on compare-up) |
| | | X1 | | Low-true pulses (set output on compare-up) |

## 16.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-13. TPM Channel Value Register High (TPMxCnVH)**

| | 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 16-14. TPM Channel Value Register Low (TPMxCnVL)**

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets

(becomes unlatched) when the TPMxCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxCnSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMxCnVH and TPMxCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKSB:CLKSA bits and the selected mode, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written.
- If (CLKSB:CLKSA not = 0:0 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKSB:CLKSA not = 0:0 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMxCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

## 16.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

## 16.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

### 16.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 16-3. After any MCU reset, CLKSB:CLKSA=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

**Table 16-7. TPM Clock Source Selection**

| CLKSB:CLKSA | TPM Clock Source to Prescaler Input |
|---|---|
| 00 | No clock selected (TPM counter disabled) |
| 01 | Bus rate clock |
| 10 | Fixed system clock |
| 11 | External source |

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL and FLL or the PLL and FLL are not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL or FLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user’s responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

### 16.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

#### 16.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

#### 16.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

### 16.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

#### 16.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

In input capture mode, the TPMxCnVH and TPMxCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

#### 16.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

### 16.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 16-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

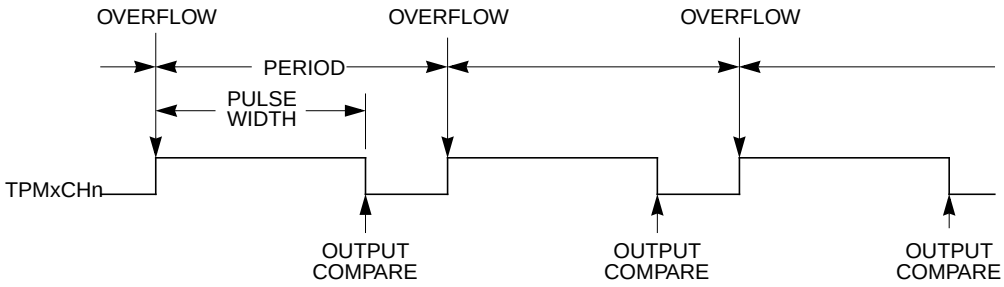



**Figure 16-15. PWM Period and Pulse Width (ELSnA=0)**

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If

the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

### 16.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

pulse width = 2 x (TPMxCnVH:TPMxCnVL)

period = 2 x (TPMxMODH:TPMxMODL); TPMxMODH:TPMxMODL=0x0001-0x7FFF

If the channel-value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPMxMODH:TPMxMODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 16-16). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

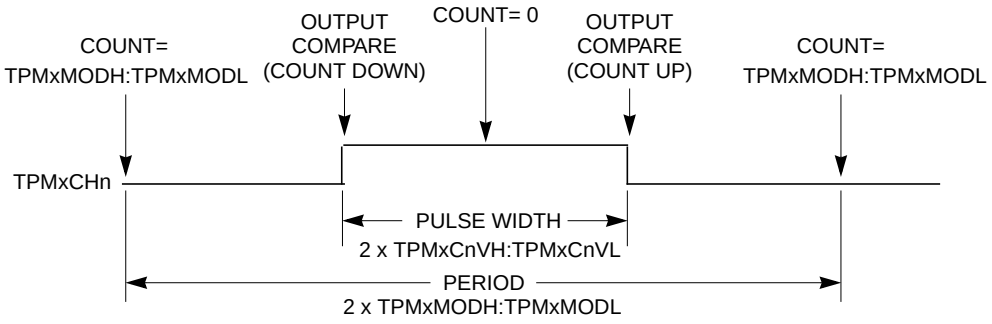



**Figure 16-16. CPWM Period and Pulse Width (ELSnA=0)**

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMxCnVH:L registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

When TPMxCNTH:TPMxCNTL=TPMxMODH:TPMxMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

## 16.5 Reset Overview

### 16.5.1 General

The TPM is reset whenever any MCU reset occurs.

### 16.5.2 Description of Reset Operation

Reset clears the TPMxSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

## 16.6 Interrupts

### 16.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel’s mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in Table 16-8 which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

**Table 16-8. Interrupt Summary**

| Interrupt | Local Enable | Source | Description |
|---|---|---|---|
| TOF | TOIE | Counter overflow | Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000) |
| CHnF | CHnIE | Channel event | An input capture or output compare event took place on channel n |

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user's guide for the interrupt module or to the chip's complete documentation for details.

## 16.6.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user's software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

### 16.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

#### 16.6.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

#### 16.6.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

### 16.6.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

#### 16.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in Section 16.6.2, "Description of Interrupt Operation."

#### 16.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described Section 16.6.2, "Description of Interrupt Operation."

#### 16.6.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described Section 16.6.2, "Description of Interrupt Operation."

## 16.7 The Differences from TPM v2 to TPM v3

1. Write to TPMxCNTH:L registers (Section 16.3.2, "TPM-Counter Registers (TPMxCNTH:TPMxCNTL)) [SE110-TPM case 7]

   Any write to TPMxCNTH or TPMxCNTL registers in TPM v3 clears the TPM counter (TPMxCNTH:L) and the prescaler counter. Instead, in the TPM v2 only the TPM counter is cleared in this case.
2. Read of TPMxCNTH:L registers (Section 16.3.2, "TPM-Counter Registers (TPMxCNTH:TPMxCNTL))
   - In TPM v3, any read of TPMxCNTH:L registers during BDM mode returns the value of the TPM counter that is frozen. In TPM v2, if only one byte of the TPMxCNTH:L registers was read before the BDM mode became active, then any read of TPMxCNTH:L registers during

BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the frozen TPM counter value.

— This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxSC, TPMxCNTH or TPMxCNTL. Instead, in these conditions the TPM v2 does not clear this read coherency mechanism.

3. Read of TPMxCnVH:L registers (Section 16.3.5, “TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)”)

— In TPM v3, any read of TPMxCnVH:L registers during BDM mode returns the value of the TPMxCnVH:L register. In TPM v2, if only one byte of the TPMxCnVH:L registers was read before the BDM mode became active, then any read of TPMxCnVH:L registers during BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the value in the TPMxCnVH:L registers.

— This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxCnSC. Instead, in this condition the TPM v2 does not clear this read coherency mechanism.

4. Write to TPMxCnVH:L registers

— Input Capture Mode (Section 16.4.2.1, “Input Capture Mode)

In this mode the TPM v3 does not allow the writes to TPMxCnVH:L registers. Instead, the TPM v2 allows these writes.

— Output Compare Mode (Section 16.4.2.2, “Output Compare Mode)

In this mode and if (CLKSB:CLKSA not = 0:0), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written. Instead, the TPM v2 always updates these registers when their second byte is written.

The following procedure can be used in the TPM v3 to verify if the TPMxCnVH:L registers were updated with the new value that was written to these registers (value in their write buffer).

...

write the new value to TPMxCnVH:L;

read TPMxCnVH and TPMxCnVL registers;

while (the read value of TPMxCnVH:L is different from the new value written to TPMxCnVH:L)

begin

read again TPMxCnVH and TPMxCnVL;

end

...

In this point, the TPMxCnVH:L registers were updated, so the program can continue and, for example, write to TPMxC0SC without cancelling the previous write to TPMxCnVH:L registers.

— Edge-Aligned PWM (Section 16.4.2.3, “Edge-Aligned PWM Mode)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the

TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from $FFFE to $FFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to $0000.

— Center-Aligned PWM (Section 16.4.2.4, “Center-Aligned PWM Mode)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from $FFFE to $FFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to (TPMxMODH:L - 1).

5. Center-Aligned PWM (Section 16.4.2.4, “Center-Aligned PWM Mode)

— TPMxCnVH:L = TPMxMODH:L [SE110-TPM case 1]

In this case, the TPM v3 produces 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

— TPMxCnVH:L = (TPMxMODH:L - 1) [SE110-TPM case 2]

In this case, the TPM v3 produces almost 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

— TPMxCnVH:L is changed from 0x0000 to a non-zero value [SE110-TPM case 3 and 5]

In this case, the TPM v3 waits for the start of a new PWM period to begin using the new duty cycle setting. Instead, the TPM v2 changes the channel output at the middle of the current PWM period (when the count reaches 0x0000).

— TPMxCnVH:L is changed from a non-zero value to 0x0000 [SE110-TPM case 4]

In this case, the TPM v3 finishes the current PWM period using the old duty cycle setting. Instead, the TPM v2 finishes the current PWM period using the new duty cycle setting.

6. Write to TPMxMODH:L registers in BDM mode (Section 16.3.3, “TPM Counter Modulo Registers (TPMxMODH:TPMxMODL))

In the TPM v3 a write to TPMxSC register in BDM mode clears the write coherency mechanism of TPMxMODH:L registers. Instead, in the TPM v2 this coherency mechanism is not cleared when there is a write to TPMxSC register.

7. Update of EPWM signal when CLKSB:CLKSA = 00

In the TPM v3 if CLKSB:CLKSA = 00, then the EPWM signal in the channel output is not update (it is frozen while CLKSB:CLKSA = 00). Instead, in the TPM v2 the EPWM signal is updated at the next rising edge of bus clock after a write to TPMxCnSC register.

The Figure 0-1 and Figure 0-2 show when the EPWM signals generated by TPM v2 and TPM v3 after the reset (CLKSB:CLKSA = 00) and if there is a write to TPMxCnSC register.

EPWM mode
TPMxMODH:TPMxMODL = 0x0007
TPMxCnVH:TPMxCnVL = 0x0005

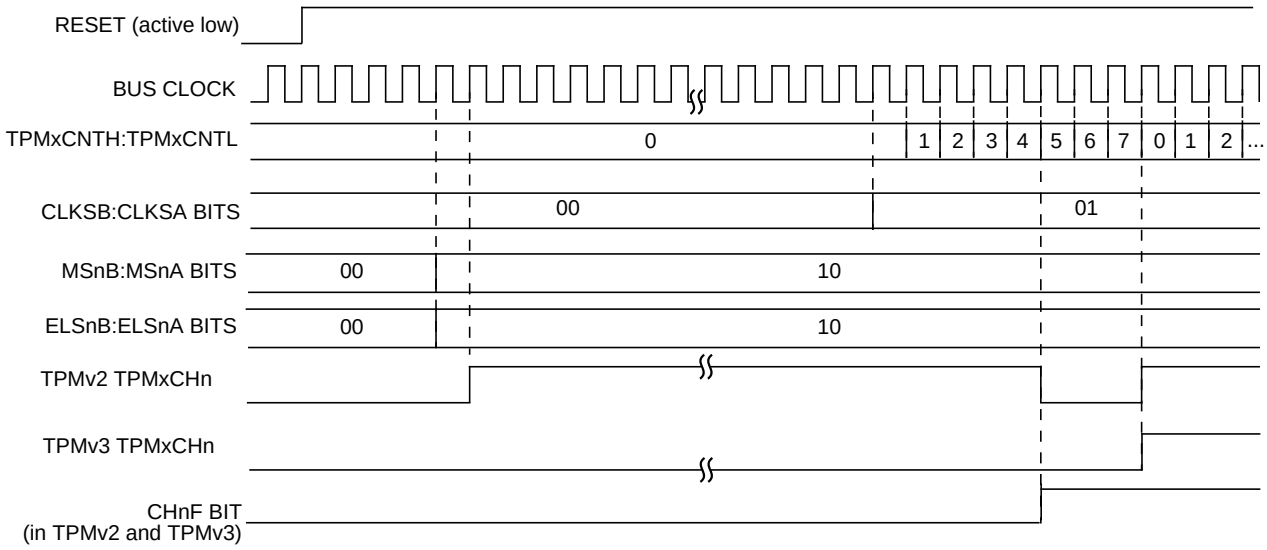



**Figure 0-1. Generation of high-true EPWM signal by TPM v2 and v3 after the reset**

EPWM mode
TPMxMODH:TPMxMODL = 0x0007
TPMxCnVH:TPMxCnVL = 0x0005

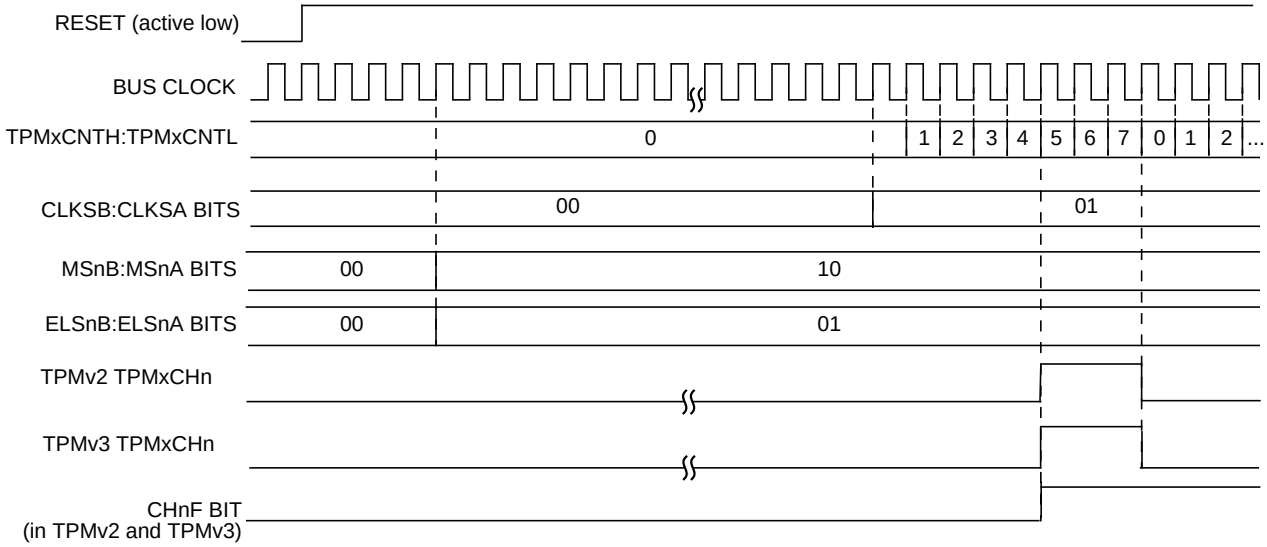



**Figure 0-2. Generation of low-true EPWM signal by TPM v2 and v3 after the reset**

The following procedure can be used in TPM v3 (when the channel pin is also a port pin) to emulate the high-true EPWM generated by TPM v2 after the reset.

...

configure the channel pin as output port pin and set the output pin;

configure the channel to generate the EPWM signal but keep ELSnB:ELSnA as 00;

configure the other registers (TPMxMODH, TPMxMODL, TPMxCnVH, TPMxCnVL, ...);

configure CLKSB:CLKSA bits (TPM v3 starts to generate the high-true EPWM signal, however TPM does not control the channel pin, so the EPWM signal is not available);

wait until the TOF is set (or use the TOF interrupt);

enable the channel output by configuring ELSnB:ELSnA bits (now EPWM signal is available);

...

# Chapter 17
# Development Support

## 17.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip Flash and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 Family, address and data bus signals are not available on external pins (not even in test modes). Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

### 17.1.1 Forcing Active Background

The method for forcing active background mode depends on the specific HCS08 derivative. For the MC9S08DZ60, you can force active background after a power-on reset by holding the BKGD pin low as the device exits the reset condition. You can also force active background by driving BKGD low immediately after a serial background command that writes a one to the BDFR bit in the SBDFR register. If no debug pod is connected to the BKGD pin, the MCU will always reset into normal operating mode.

### 17.1.2 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system include:

- Two trigger comparators: Two address + read/write (R/W) or one full address + data + R/W
- Flexible 8-word by 16-bit FIFO (first-in, first-out) buffer for capture information:
  - Change-of-flow addresses or
  - Event-only data
- Two types of breakpoints:
  - Tag breakpoints for instruction opcodes
  - Force breakpoints for any address access
- Nine trigger modes:
  - Basic: A-only, A OR B
  - Sequence: A then B
  - Full: A AND B data, A AND NOT B data
  - Event (store data): Event-only B, A then event-only B
  - Range: Inside range (A ≤ address ≤ B), outside range (address < A or address > B)

## 17.2 Background Debug Controller (BDC)

All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes $V_{DD}$. An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes $V_{DD}$ can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

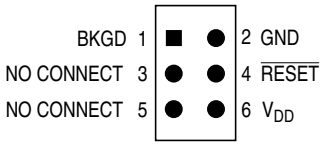



**Figure 17-1. BDM Tool Connector**

## 17.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to Section 17.2.2, "Communication Details."

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 17.2.2, "Communication Details," for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

### 17.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 17-2 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

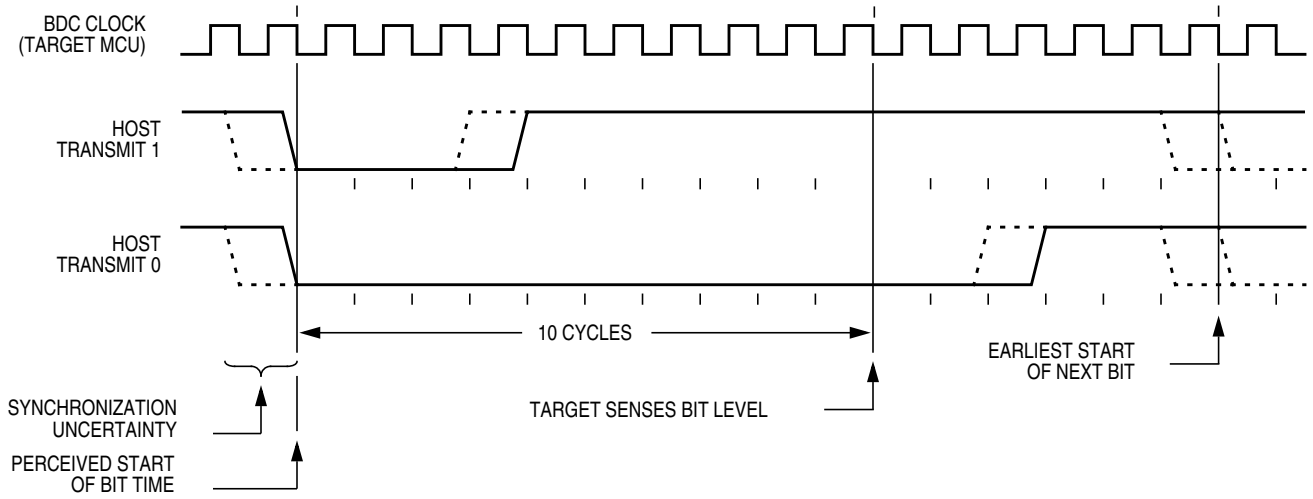



**Figure 17-2. BDC Host-to-Target Serial Bit Timing**

Figure 17-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

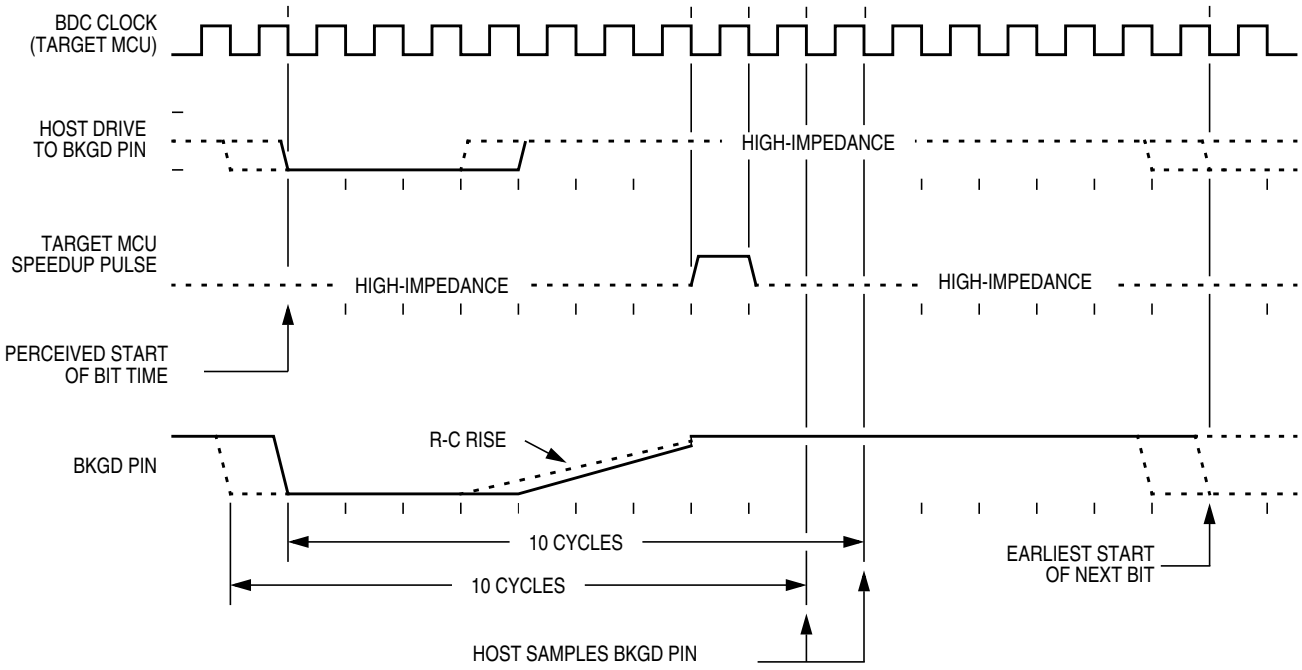



**Figure 17-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 17-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

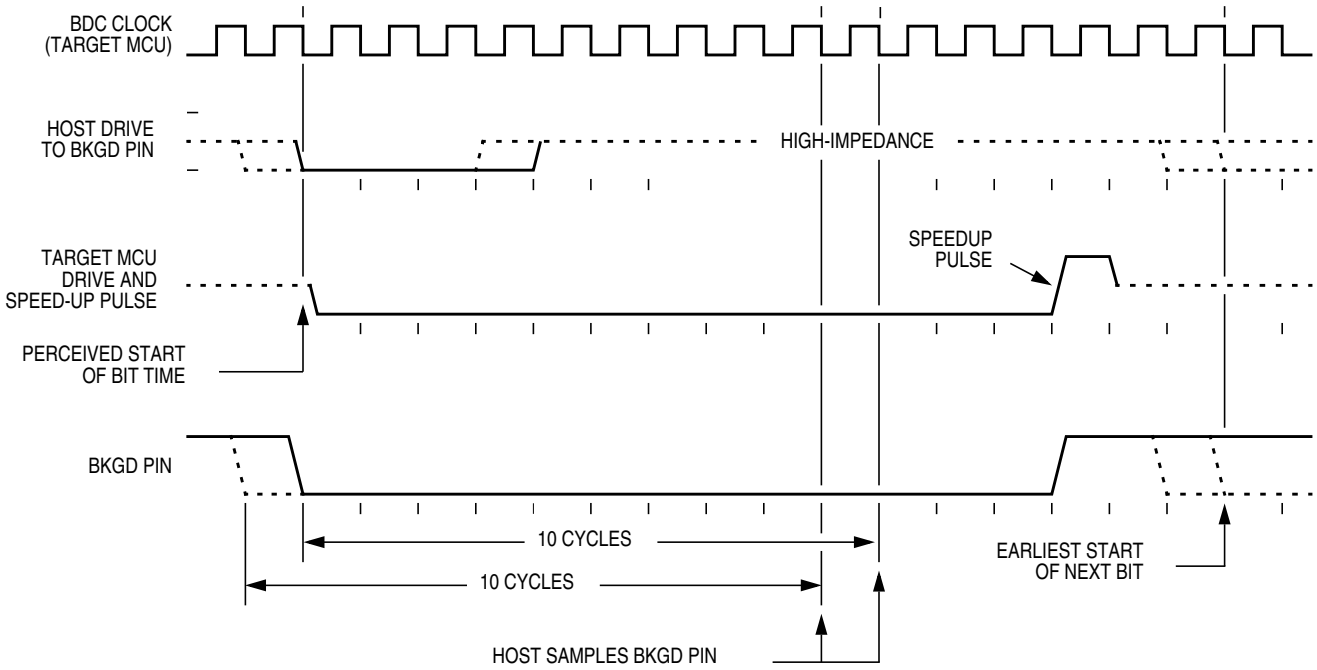



**Figure 17-4. BDM Target-to-Host Serial Bit Timing (Logic 0)**

## 17.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Active background mode commands require that the target MCU is currently in the active background mode while non-intrusive commands may be issued at any time whether the target MCU is in active background mode or running a user application program.

Table 17-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

### Coding Structure Nomenclature

This nomenclature is used in Table 17-1 to describe the coding 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 BDC clock cycles |
| AAAA | = | a 16-bit address in the host-to-target direction |
| RD | = | 8 bits of read data in the target-to-host direction |
| WD | = | 8 bits of write data in the host-to-target direction |
| RD16 | = | 16 bits of read data in the target-to-host direction |
| WD16 | = | 16 bits of write data in the host-to-target direction |
| SS | = | the contents of BDCSCR in the target-to-host direction (STATUS) |
| CC | = | 8 bits of write data for BDCSCR in the host-to-target direction (CONTROL) |
| RBKP | = | 16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register) |
| WBKP | = | 16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register) |

**Table 17-1. BDC Command Summary**

| Command Mnemonic | Active BDM/ Non-intrusive | Coding Structure | Description |
|---|---|---|---|
| SYNC | Non-intrusive | n/a[1] | Request a timed reference pulse to determine target BDC communication speed |
| ACK_ENABLE | Non-intrusive | D5/d | Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D. |
| ACK_DISABLE | Non-intrusive | D6/d | Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D. |
| BACKGROUND | Non-intrusive | 90/d | Enter active background mode if enabled (ignore if ENBDM bit equals 0) |
| READ_STATUS | Non-intrusive | E4/SS | Read BDC status from BDCSCR |
| WRITE_CONTROL | Non-intrusive | C4/CC | Write BDC controls in BDCSCR |
| READ_BYTE | Non-intrusive | E0/AAAA/d/RD | Read a byte from target memory |
| READ_BYTE_WS | Non-intrusive | E1/AAAA/d/SS/RD | Read a byte and report status |
| READ_LAST | Non-intrusive | E8/SS/RD | Re-read byte from address just read and report status |
| WRITE_BYTE | Non-intrusive | C0/AAAA/WD/d | Write a byte to target memory |
| WRITE_BYTE_WS | Non-intrusive | C1/AAAA/WD/d/SS | Write a byte and report status |
| READ_BKPT | Non-intrusive | E2/RBKP | Read BDCBKPT breakpoint register |
| WRITE_BKPT | Non-intrusive | C2/WBKP | Write BDCBKPT breakpoint register |
| GO | Active BDM | 08/d | Go to execute the user application program starting at the address currently in the PC |
| TRACE1 | Active BDM | 10/d | Trace 1 user instruction at the address in the PC, then return to active background mode |
| TAGGO | Active BDM | 18/d | Same as GO but enable external tagging (HCS08 devices have no external tagging pin) |
| READ_A | Active BDM | 68/d/RD | Read accumulator (A) |
| READ_CCR | Active BDM | 69/d/RD | Read condition code register (CCR) |
| READ_PC | Active BDM | 6B/d/RD16 | Read program counter (PC) |
| READ_HX | Active BDM | 6C/d/RD16 | Read H and X register pair (H:X) |
| READ_SP | Active BDM | 6F/d/RD16 | Read stack pointer (SP) |
| READ_NEXT | Active BDM | 70/d/RD | Increment H:X by one then read memory byte located at H:X |
| READ_NEXT_WS | Active BDM | 71/d/SS/RD | Increment H:X by one then read memory byte located at H:X. Report status and data. |
| WRITE_A | Active BDM | 48/WD/d | Write accumulator (A) |
| WRITE_CCR | Active BDM | 49/WD/d | Write condition code register (CCR) |
| WRITE_PC | Active BDM | 4B/WD16/d | Write program counter (PC) |
| WRITE_HX | Active BDM | 4C/WD16/d | Write H and X register pair (H:X) |
| WRITE_SP | Active BDM | 4F/WD16/d | Write stack pointer (SP) |
| WRITE_NEXT | Active BDM | 50/WD/d | Increment H:X by one, then write memory byte located at H:X |
| WRITE_NEXT_WS | Active BDM | 51/WD/d/SS | Increment H:X by one, then write memory byte located at H:X. Also report status. |

[1] The SYNC command is a special operation that does not have a command code.

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

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

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

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

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 communication protocol can easily tolerate speed errors of several percent.

### 17.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

## 17.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user's memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in Section 17.3.6, "Hardware Breakpoints."

### 17.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

### 17.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 17.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

### 17.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

### 17.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

### 17.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

**A-Only** — Trigger when the address matches the value in comparator A

**A OR B** — Trigger when the address matches either the value in comparator A or the value in comparator B

**A Then B** — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

**A AND B Data (Full Mode)** — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

**A AND NOT B Data (Full Mode)** — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

**Event-Only B (Store Data)** — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

**A Then Event-Only B (Store Data)** — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

**Inside Range (A ≤ Address ≤ B)** — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

**Outside Range (Address < A or Address > B)** — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

### 17.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGC register may be set to 1 to allow any of the trigger conditions described in Section 17.3.5, “Trigger Modes,” to be used to generate a hardware breakpoint request to the CPU. TAG in DBGC controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

## 17.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 17.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

### 17.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ENBDM | BDMACT | BKPTEN | FTS | CLKSW | WS | WSF | DVF |
| W | | | | | | | | |
| Normal Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset in Active BDM: | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-5. BDC Status and Control Register (BDCSCR)**

**Table 17-2. BDCSCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ENBDM | **Enable BDM (Permit Active Background Mode)** — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it.<br>0 BDM cannot be made active (non-intrusive commands still allowed)<br>1 BDM can be made active to allow active background mode commands |
| 6<br>BDMACT | **Background Mode Active Status** — This is a read-only status bit.<br>0 BDM not active (user application program running)<br>1 BDM active and waiting for serial commands |
| 5<br>BKPTEN | **BDC Breakpoint Enable** — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored.<br>0 BDC breakpoint disabled<br>1 BDC breakpoint enabled |
| 4<br>FTS | **Force/Tag Select** — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode.<br>0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction<br>1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode) |
| 3<br>CLKSW | **Select Source for BDC Communications Clock** — CLKSW defaults to 0, which selects the alternate BDC clock source.<br>0 Alternate BDC clock source<br>1 MCU bus clock |

**Table 17-2. BDCSCR Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>WS | **Wait or Stop Status** — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.<br>0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active)<br>1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode |
| 1<br>WSF | **Wait or Stop Failure Status** — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.)<br>0 Memory access did not conflict with a wait or stop instruction<br>1 Memory access command failed because the CPU entered wait or stop mode |
| 0<br>DVF | **Data Valid Failure Status** — This status bit is not used in the MC9S08DZ60 Series because it does not have any slow access memory.<br>0 Memory access did not conflict with a slow memory access<br>1 Memory access command failed because CPU was not finished with a slow memory access |

#### 17.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to Section 17.2.4, “BDC Hardware Breakpoint.”

### 17.4.2 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial background mode command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

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

= Unimplemented or Reserved

1 BDFR is writable only through serial background mode debug commands, not from user programs.

**Figure 17-6. System Background Debug Force Reset Register (SBDFR)**

**Table 17-3. SBDFR Register Field Description**

| Field | Description |
|---|---|
| 0<br>BDFR | **Background Debug Force Reset** — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program. |

## 17.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

### 17.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 17.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 17.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 17.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

#### 17.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

#### 17.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn’t necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

### 17.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DBGEN | ARM | TAG | BRKEN | RWA | RWAEN | RWB | RWBEN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-7. Debug Control Register (DBGC)**

**Table 17-4. DBGC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DBGEN | **Debug Module Enable** — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure.<br>0 DBG disabled<br>1 DBG enabled |
| 6<br>ARM | **Arm Control** — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN.<br>0 Debugger not armed<br>1 Debugger armed |
| 5<br>TAG | **Tag/Force Select** — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect.<br>0 CPU breaks requested as force type requests<br>1 CPU breaks requested as tag type requests |
| 4<br>BRKEN | **Break Enable** — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests.<br>0 CPU break requests not enabled<br>1 Triggers cause a break request to the CPU |
| 3<br>RWA | **R/W Comparison Value for Comparator A** — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A.<br>0 Comparator A can only match on a write cycle<br>1 Comparator A can only match on a read cycle |
| 2<br>RWAEN | **Enable R/W for Comparator A** — Controls whether the level of R/W is considered for a comparator A match.<br>0 R/W is not used in comparison A<br>1 R/W is used in comparison A |
| 1<br>RWB | **R/W Comparison Value for Comparator B** — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B.<br>0 Comparator B can match only on a write cycle<br>1 Comparator B can match only on a read cycle |
| 0<br>RWBEN | **Enable R/W for Comparator B** — Controls whether the level of R/W is considered for a comparator B match.<br>0 R/W is not used in comparison B<br>1 R/W is used in comparison B |

### 17.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TRGSEL | BEGIN | 0 | 0 | TRG3 | TRG2 | TRG1 | TRG0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-8. Debug Trigger Register (DBGT)**

**Table 17-5. DBGT Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TRGSEL | **Trigger Type** — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed.<br>0 Trigger on access to compare address (force)<br>1 Trigger if opcode at compare address is executed (tag) |
| 6<br>BEGIN | **Begin/End Trigger Select** — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces.<br>0 Data stored in FIFO until trigger (end trace)<br>1 Trigger initiates data storage (begin trace) |
| 3:0<br>TRG[3:0] | **Select Trigger Mode** — Selects one of nine triggering modes, as described below.<br>0000 A-only<br>0001 A OR B<br>0010 A Then B<br>0011 Event-only B (store data)<br>0100 A then event-only B (store data)<br>0101 A AND B data (full mode)<br>0110 A AND NOT B data (full mode)<br>0111 Inside range: A ≤ address ≤ B<br>1000 Outside range: address < A or address > B<br>1001 – 1111 (No trigger) |

### 17.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AF | BF | ARMF | 0 | CNT3 | CNT2 | CNT1 | CNT0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-9. Debug Status Register (DBGS)**

**Table 17-6. DBGS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AF | **Trigger Match A Flag** — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming.<br>0 Comparator A has not matched<br>1 Comparator A match |
| 6<br>BF | **Trigger Match B Flag** — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming.<br>0 Comparator B has not matched<br>1 Comparator B match |
| 5<br>ARMF | **Arm Flag** — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGC.<br>0 Debugger not armed<br>1 Debugger armed |
| 3:0<br>CNT[3:0] | **FIFO Valid Count** — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO.<br>0000 Number of valid words in FIFO = No valid data<br>0001 Number of valid words in FIFO = 1<br>0010 Number of valid words in FIFO = 2<br>0011 Number of valid words in FIFO = 3<br>0100 Number of valid words in FIFO = 4<br>0101 Number of valid words in FIFO = 5<br>0110 Number of valid words in FIFO = 6<br>0111 Number of valid words in FIFO = 7<br>1000 Number of valid words in FIFO = 8 |

# Appendix A
# Electrical Characteristics

## A.1 Introduction

This section contains the most accurate electrical and timing information for the MC9S08DZ60 Series of microcontrollers available at the time of publication.

## A.2 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:

**Table A-1. Parameter Classifications**

| | |
|---|---|
| **P** | Those parameters are guaranteed during production testing on each individual device. |
| **C** | Those parameters are achieved by the 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. |

**NOTE**

The classification is shown in the column labeled “C” in the parameter tables where appropriate.

## A.3 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only, and functional operation at the maxima is not guaranteed. Stress beyond the limits specified in Table A-2 may affect device reliability or cause permanent damage to the device. For functional operating conditions, refer to the remaining tables in this section.

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 (for instance, either $V_{SS}$ or $V_{DD}$).

**Table A-2. Absolute Maximum Ratings**

| Num | Rating | Symbol | Value | Unit |
|---|---|---|---|---|
| 1 | Supply voltage | $V_{DD}$ | –0.3 to + 5.8 | V |
| 2 | Input voltage | $V_{In}$ | – 0.3 to $V_{DD}$ + 0.3 | V |
| 3 | Instantaneous maximum current Single pin limit (applies to all port pins)[1, 2, 3] | $I_D$ | ± 25 | mA |
| 4 | Maximum current into $V_{DD}$ | $I_{DD}$ | 120 | mA |
| 5 | Storage temperature | $T_{stg}$ | –55 to +150 | °C |

1 Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive ($V_{DD}$) and negative ($V_{SS}$) clamp voltages, then use the larger of the two resistance values.

2 All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.

3 Power supply must maintain regulation within operating $V_{DD}$ range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than $I_{DD}$, the injection current may flow out of $V_{DD}$ and could result in external power supply going out of regulation. Ensure external $V_{DD}$ load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if the clock rate is very low which would reduce overall power consumption.

## A.4 Thermal Characteristics

This section provides information about operating temperature range, power dissipation, and package thermal resistance. Power dissipation on I/O pins is usually small compared to the power dissipation in on-chip logic and it is user-determined rather than being controlled by the MCU design. In order to take $P_{I/O}$ into account in power calculations, determine the difference between actual pin voltage and $V_{SS}$ or $V_{DD}$ and multiply by the pin current for each I/O pin. Except in cases of unusually high pin current (heavy loads), the difference between pin voltage and $V_{SS}$ or $V_{DD}$ will be very small.

**Table A-3. Thermal Characteristics**

| Num | C | Rating | Symbol | Value | Unit | Temp. Code |
|---|---|---|---|---|---|---|
| 1 | D | Operating temperature range (packaged) | $T_A$ | –40 to 125<br>–40 to 105<br>–40 to 85 | °C | M<br>V<br>C |
| 2 | T | Maximum Junction Temperature[1] | $T_J$ | 135 | °C | — |
| 3 | D | Thermal resistance[2] | | | | |
| | | Single-layer board | | | | |
| | | 64-pin LQFP | $\theta_{JA}$ | 69 | °C/W | |
| | | 48-pin LQFP | $\theta_{JA}$ | 75 | °C/W | |
| | | 32-pin LQFP | $\theta_{JA}$ | 80 | °C/W | |
| | | Four-Layer board | | | | |
| | | 64-pin LQFP | $\theta_{JA}$ | 51 | °C/W | |
| | | 48-pin LQFP | $\theta_{JA}$ | 51 | °C/W | |
| | | 32-pin LQFP | $\theta_{JA}$ | 52 | °C/W | |

1 Junction temperature is a function of die size, on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and board thermal resistance.

2 Junction to Ambient Natural Convection

The average chip-junction temperature ($T_J$) in °C can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \qquad \textit{Eqn. A-1}$$

where:

$T_A$ = Ambient temperature, °C

$\theta_{JA}$ = Package thermal resistance, junction-to-ambient, °C/W

$P_D = P_{int} + P_{I/O}$

$P_{int} = I_{DD} \times V_{DD}$, Watts — chip internal power

$P_{I/O}$ = Power dissipation on input and output pins — user determined

For most applications, $P_{I/O} << P_{int}$ and can be neglected. An approximate relationship between $P_D$ and $T_J$ (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \qquad \textit{Eqn. A-2}$$

Solving equations 1 and 2 for K gives:

$$K = P_D \times (T_A + 273°C) + \theta_{JA} \times (P_D)^2 \qquad \textit{Eqn. A-3}$$

where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring $P_D$ (at equilibrium) for a known $T_A$. Using this value of K, the values of $P_D$ and $T_J$ can be obtained by solving equations 1 and 2 iteratively for any value of $T_A$.

## A.5 ESD Protection and Latch-Up Immunity

Although damage from electrostatic discharge (ESD) is much less common on these devices than on early CMOS circuits, normal handling precautions should be used to avoid exposure to static discharge. Qualification tests are performed to ensure that these devices can withstand exposure to reasonable levels of static without suffering any permanent damage.

All ESD testing is in conformity with 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 Charge Device Model (CDM).

A device is 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-4. ESD and Latch-up Test Conditions**

| Model | Description | Symbol | Value | Unit |
|---|---|---|---|---|
| Human Body | Series Resistance | R1 | 1500 | Ω |
| | Storage Capacitance | C | 100 | pF |
| | Number of Pulse per pin | – | 3 | |
| Latch-up | Minimum input voltage limit | | –2.5 | V |
| | Maximum input voltage limit | | 7.5 | V |

**Table A-5. ESD and Latch-Up Protection Characteristics**

| Num | Rating | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Human Body Model (HBM) | $V_{HBM}$ | +/- 2000 | – | V |
| 2 | Charge Device Model (CDM) | $V_{CDM}$ | +/- 500 | – | V |
| 3 | Latch-up Current at $T_A$ = 125°C | $I_{LAT}$ | +/- 100 | – | mA |

## A.6 DC Characteristics

This section includes information about power supply requirements, I/O pin characteristics, and power supply current in various operating modes.

**Table A-6. DC Characteristics**

| Num | C | Characteristic | Symbol | Condition | Min | Typ[1] | Max | Unit |
|---|---|---|---|---|---|---|---|---|
| 1 | — | Operating Voltage | $V_{DD}$ | | 2.7 | — | 5.5 | V |
| 2 | P | Output high voltage — All I/O pins, low-drive strength | $V_{OH}$ | 5 V, $I_{Load}$ = –2 mA | $V_{DD}$ – 1.5 | — | — | V |
| | C | | | 3 V, $I_{Load}$ = –0.6 mA | $V_{DD}$ – 1.5 | — | — | |
| | C | | | 5 V, $I_{Load}$ = –0.4 mA | $V_{DD}$ – 0.8 | — | — | |
| | C | | | 3 V, $I_{Load}$ = –0.24 mA | $V_{DD}$ – 0.8 | — | — | |
| | P | Output high voltage — All I/O pins, high-drive strength | | 5 V, $I_{Load}$ = –10 mA | $V_{DD}$ – 1.5 | — | — | |
| | C | | | 3 V, $I_{Load}$ = –3 mA | $V_{DD}$ – 1.5 | — | — | |
| | C | | | 5 V, $I_{Load}$ = –2 mA | $V_{DD}$ – 0.8 | — | — | |
| | C | | | 3 V, $I_{Load}$ = –0.4 mA | $V_{DD}$ – 0.8 | — | — | |
| 3 | C | Output high current — Max total $I_{OH}$ for all ports | $I_{OHT}$ | 5 V | 0 | — | -100 | mA |
| | | | | 3 V | 0 | — | -60 | |
| 4 | P | Output low voltage — All I/O pins, low-drive strength | $V_{OL}$ | 5 V, $I_{Load}$ = 2 mA | — | — | 1.5 | V |
| | C | | | 3 V, $I_{Load}$ = 0.6 mA | — | — | 1.5 | |
| | C | | | 5 V, $I_{Load}$ = 0.4 mA | — | — | 0.8 | |
| | C | | | 3 V, $I_{Load}$ = 0.24 mA | — | — | 0.8 | |
| | P | Output low voltage — All I/O pins, high-drive strength | | 5 V, $I_{Load}$ = 10 mA | — | — | 1.5 | |
| | C | | | 3 V, $I_{Load}$ = 3 mA | — | — | 1.5 | |
| | C | | | 5 V, $I_{Load}$ = 2 mA | — | — | 0.8 | |
| | C | | | 3 V, $I_{Load}$ = 0.4 mA | — | — | 0.8 | |
| 5 | C | Output low current — Max total $I_{OL}$ for all ports | $I_{OLT}$ | 5 V | 0 | — | 100 | mA |
| | | | | 3 V | 0 | — | 60 | |
| 6 | C | Input high voltage; all digital inputs | $V_{IH}$ | 5V | 0.65 x $V_{DD}$ | — | — | V |
| 7 | C | Input low voltage; all digital inputs | $V_{IL}$ | 5V | — | — | 0.35 x $V_{DD}$ | |
| 8 | C | Input hysteresis | $V_{hys}$ | | 0.06 x $V_{DD}$ | | | mV |
| 9 | P | Input leakage current (Per pin) all input only pins | $\lvert I_{In} \rvert$ | $V_{In}$ = $V_{DD}$ or $V_{SS}$ | — | 0.1 | 1 | µA |
| 10 | P | Hi-Z (off-state) leakage current (per pin) all input/output | $\lvert I_{OZ} \rvert$ | $V_{In}$ = $V_{DD}$ or $V_{SS}$ | — | 0.1 | 1 | µA |
| 11 | P | Pullup resistors (or Pulldown[2] resistors when enabled) | $R_{PU}$, $R_{PD}$ | 5 V | 20 | 45 | 65 | kΩ |
| | C | | | 3 V | 20 | 45 | 65 | |
| 12 | T | Input Capacitance, all pins | $C_{In}$ | | — | — | 8 | pF |
| 13 | D | RAM retention voltage | $V_{RAM}$ | | — | 0.6 | 1.0 | V |

**Table A-6. DC Characteristics (continued)**

| Num | C | Characteristic | Symbol | Condition | Min | Typ[1] | Max | Unit |
|---|---|---|---|---|---|---|---|---|
| 14 | D | POR re-arm voltage[3] | $V_{POR}$ | | 0.9 | 1.4 | 2.0 | V |
| 15 | D | POR re-arm time[4] | $t_{POR}$ | | 10 | — | — | µs |
| 16 | P | Low-voltage detection threshold — high range<br>$V_{DD}$ falling<br>$V_{DD}$ rising | $V_{LVD1}$ | | <br>3.9<br>4.0 | <br>4.0<br>4.1 | <br>4.1<br>4.2 | V |
| 17 | P | Low-voltage detection threshold — low range<br>$V_{DD}$ falling<br>$V_{DD}$ rising | $V_{LVD0}$ | | <br>2.48<br>2.54 | <br>2.56<br>2.62 | <br>2.64<br>2.70 | V |
| 18 | C | Low-voltage warning threshold — high range 1<br>$V_{DD}$ falling<br>$V_{DD}$ rising | $V_{LVW3}$ | | <br>4.5<br>4.6 | <br>4.6<br>4.7 | <br>4.7<br>4.8 | V |
| 19 | P | Low-voltage warning threshold — high range 0<br>$V_{DD}$ falling<br>$V_{DD}$ rising | $V_{LVW2}$ | | <br>4.2<br>4.3 | <br>4.3<br>4.4 | <br>4.4<br>4.5 | V |
| 20 | P | Low-voltage warning threshold low range 1<br>$V_{DD}$ falling<br>$V_{DD}$ rising | $V_{LVW1}$ | | <br>2.84<br>2.90 | <br>2.92<br>2.98 | <br>3.00<br>3.06 | V |
| 21 | C | Low-voltage warning threshold — low range 0<br>$V_{DD}$ falling<br>$V_{DD}$ rising | $V_{LVW0}$ | | <br>2.66<br>2.72 | <br>2.74<br>2.80 | <br>2.82<br>2.88 | V |
| 22 | T | Low-voltage inhibit reset/recover hysteresis | $V_{hys}$ | 5 V<br>3 V | —<br>— | 100<br>60 | —<br>— | mV |
| 23 | D | dc injection current [5, 6, 7, 8]<br>Single pin limit<br>Total MCU limit, includes sum of all stressed pins | $I_{IC}$ | $V_{IN} > V_{DD}$<br>$V_{IN} < V_{SS}$<br>$V_{IN} > V_{DD}$<br>$V_{IN} < V_{SS}$ | 0<br>0<br>0<br>0 | —<br>—<br>—<br>— | 2<br>–0.2<br>25<br>–5 | mA |
| 24 | C | Bandgap Voltage Reference<br>Factory trimmed at<br>$V_{DD}$ = 5.0 V, Temp = 25°C | $V_{BG}$ | | 1.19 | 1.20 | 1.21 | V |

1 Typical values are measured at 25°C. Characterized, not tested

2 When a pin interrupt is configured to detect rising edges, pulldown resistors are used in place of pullup resistors.

3 Maximum is highest voltage that POR is guaranteed.

4 Simulated, not tested

5 Power supply must maintain regulation within operating $V_{DD}$ range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than $I_{DD}$, the injection current may flow out of $V_{DD}$ and could result in external power supply going out of regulation. Ensure external $V_{DD}$ load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if clock rate is very low which (would reduce overall power consumption).

6 All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.

[7] Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.

[8] PTE1 does not have a clamp diode to $V_{DD}$. Do not drive PTE1 above $V_{DD}$.

## A.7 Supply Current Characteristics

**Table A-7. Supply Current Characteristics**

| Num | C | Parameter | Symbol | $V_{DD}$ (V) | Typical[1] | Max[2] | Unit |
|---|---|---|---|---|---|---|---|
| 1 | C | Run supply current[3] measured at (CPU clock = 2 MHz, $f_{Bus}$ = 1 MHz) | $RI_{DD}$ | 5 | 3 | 7.5 | mA |
| | C | | | 3 | 2.8 | 7.4 | |
| 2 | P | Run supply current[3] measured at (CPU clock = 16 MHz, $f_{Bus}$ = 8 MHz) | $RI_{DD}$ | 5 | 7.7 | 11.4 | mA |
| | C | | | 3 | 7.4 | 11.2 | |
| 3 | P | Run supply current[3] measured at (CPU clock = 40 MHz, $f_{Bus}$ = 20 MHz) | $RI_{DD}$ | 5 | 15 | 24 | mA |
| | C | | | 3 | 14 | 23 | |
| 4 | P[4] | Stop3 mode supply current –40 °C (C, V, & M suffix) | $S3I_{DD}$ | 5 | 0.9 | — | µA |
| | P[4] | 25 °C (All parts) | | | 1.0 | — | |
| | P | 105 °C (V suffix only) | | | 26 | 39 | |
| | P | 125 °C (M suffix only) | | | 62 | 90 | |
| | C | –40 °C (C, V, & M suffix) | | 3 | 0.8 | — | |
| | C | 25 °C (All parts) | | | 0.9 | — | |
| | C | 105 °C (V suffix only) | | | 21 | 32 | |
| | C | 125 °C (M suffix only) | | | 52 | 80 | |
| 5 | P[4] | Stop2 mode supply current –40 °C (C, V, & M suffix) | $S2I_{DD}$ | 5 | 0.8 | — | µA |
| | P[4] | 25 °C (All parts) | | | 0.9 | — | |
| | P | 105 °C (V suffix only) | | | 25 | 37 | |
| | P | 125 °C (M suffix only) | | | 46 | 70 | |
| | C | –40 °C (C, V, & M suffix) | | 3 | 0.7 | — | |
| | C | 25 °C (All parts) | | | 0.8 | — | |
| | C | 105 °C (V suffix only) | | | 20 | 30 | |
| | C | 125 °C (M suffix only) | | | 40 | 60 | |

**Table A-7. Supply Current Characteristics (continued)**

| Num | C | Parameter | Symbol | $V_{DD}$ (V) | Typical[1] | Max[2] | Unit |
|---|---|---|---|---|---|---|---|
| 6 | C | RTC adder to stop2 or stop3[5], 25°C | | 5 | 300 | — | nA |
| | | | | 3 | 300 | — | nA |
| 7 | C | LVD adder to stop3 (LVDE = LVDSE = 1) | | 5 | 110 | — | µA |
| | | | | 3 | 90 | — | µA |
| 8 | C | Adder to stop3 for oscillator enabled[6] (IRCLKEN = 1 and IREFSTEN = 1 or ERCLKEN = 1 and EREFSTEN = 1) | | 5 | 5 | — | µA |
| | | | | 3 | 5 | — | µA |

1 Typicals are measured at 25°C, unless otherwise noted.

2 Maximum values in this column apply for the full operating temperature range of the device unless otherwise noted.

3 All modules except ADC active, MCG configured for FBE, and does not include any dc loads on port pins

4 Stop currents are tested in production for 25°C on all parts. Tests at other temperatures depend upon the part number suffix and maturity of the product. Freescale may eliminate a test insertion at a particular temperature from the production test flow once sufficient data has been collected and is approved.

5 Most customers are expected to find that auto-wakeup from stop2 or stop3 can be used instead of the higher current wait mode.

6 Values given under the following conditions: low range operation (RANGE = 0), low power mode (HGO = 0).

## A.8 Analog Comparator (ACMP) Electricals

**Table A-8. Analog Comparator Electrical Specifications**

| Num | C | Rating | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | — | Supply voltage | $V_{DD}$ | 2.7 | — | 5.5 | V |
| 2 | D | Supply current (active) | $I_{DDAC}$ | — | 20 | 35 | µA |
| 3 | D | Analog input voltage | $V_{AIN}$ | $V_{SS}$ – 0.3 | — | $V_{DD}$ | V |
| 4 | D | Analog input offset voltage | $V_{AIO}$ | | 20 | 40 | mV |
| 5 | D | Analog Comparator hysteresis | $V_H$ | 3.0 | 6.0 | 20.0 | mV |
| 6 | D | Analog input leakage current | $I_{ALKG}$ | -- | -- | 1.0 | µA |
| 7 | D | Analog Comparator initialization delay | $t_{AINIT}$ | — | — | 1.0 | µs |

## A.9 ADC Characteristics

**Table A-9. 12-bit ADC Operating Conditions**

| Characteristic | Conditions | Symb | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| Supply voltage | Absolute | $V_{DDAD}$ | 2.7 | — | 5.5 | V | |
| | Delta to $V_{DD}$ ($V_{DD}$-$V_{DDAD}$)[2] | $\Delta V_{DDAD}$ | -100 | 0 | +100 | mV | |
| Ground voltage | Delta to $V_{SS}$ ($V_{SS}$-$V_{SSAD}$)[2] | $\Delta V_{SSAD}$ | -100 | 0 | +100 | mV | |

**Table A-9. 12-bit ADC Operating Conditions (continued)**

| Characteristic | Conditions | Symb | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| Ref Voltage High | | $V_{REFH}$ | 2.7 | $V_{DDAD}$ | $V_{DDAD}$ | V | Applicable in only 64-pin packages {$V_{REFH} < V_{DDAD}$ characterized but not production test} |
| Ref Voltage Low | | $V_{REFL}$ | $V_{SSAD}$ | $V_{SSAD}$ | $V_{SSAD}$ | V | Not Applicable in 64-pin packages (only 32- and 48-pin packages) |
| Input Voltage | | $V_{ADIN}$ | $V_{REFL}$ | — | $V_{REFH}$ | V | |
| Input Capacitance | | $C_{ADIN}$ | — | 4.5 | 5.5 | pF | |
| Input Resistance | | $R_{ADIN}$ | — | 3 | 5 | kΩ | |
| Analog Source Resistance | 12 bit mode<br>$f_{ADCK}$ > 4MHz<br>$f_{ADCK}$ < 4MHz | $R_{AS}$ | <br>—<br>— | <br>—<br>— | <br>2<br>5 | kΩ | External to MCU |
| | 10 bit mode<br>$f_{ADCK}$ > 4MHz<br>$f_{ADCK}$ < 4MHz | | <br>—<br>— | <br>—<br>— | <br>5<br>10 | | |
| | 8 bit mode (all valid $f_{ADCK}$) | | — | — | 10 | | |
| ADC Conversion Clock Freq. | High Speed (ADLPC=0) | $f_{ADCK}$ | 0.4 | — | 8.0 | MHz | |
| | Low Power (ADLPC=1) | | 0.4 | — | 4.0 | | |

1 Typical values assume $V_{DDAD}$ = 5.0V, Temp = 25°C, $f_{ADCK}$=1.0MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

2 DC potential difference.

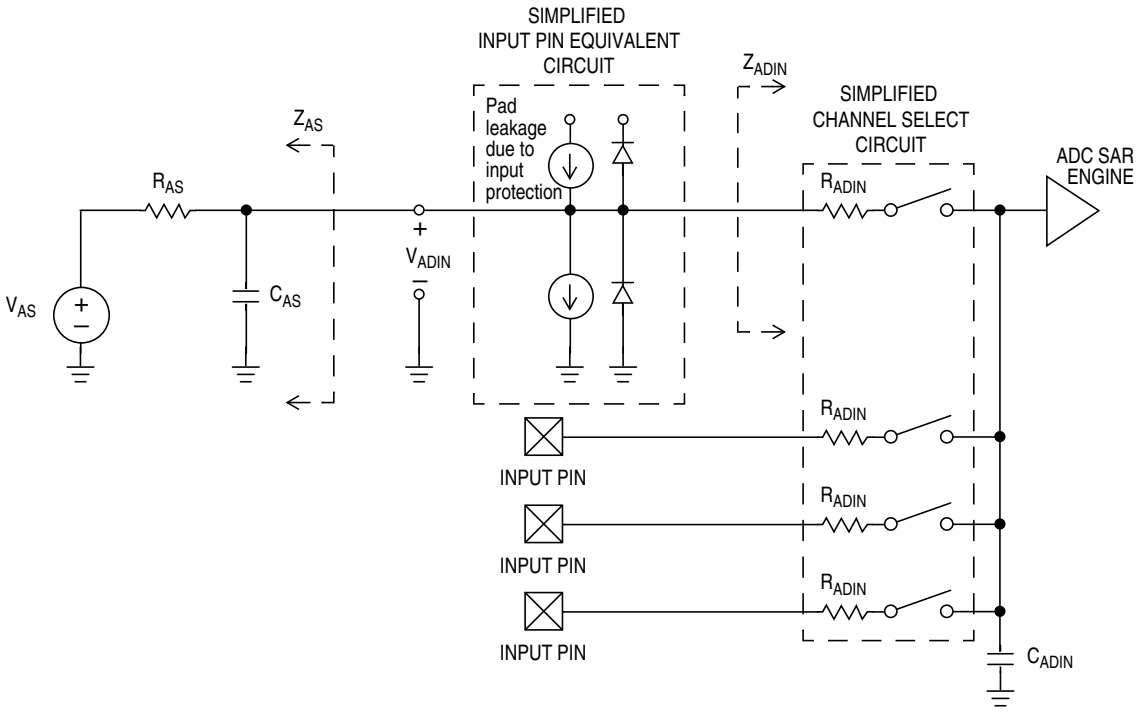



**Figure A-1. ADC Input Impedance Equivalency Diagram**

**Table A-10. 12-bit ADC Characteristics ($V_{REFH}$ = $V_{DDAD}$, $V_{REFL}$ = $V_{SSAD}$)**

| Characteristic | Conditions | C | Symb | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|---|
| Supply Current | ADLPC=1<br>ADLSMP=1<br>ADCO=1 | T | $I_{DD}$ + $I_{DDAD}$ | — | 133 | — | µA | ADC current only |
| Supply Current | ADLPC=1<br>ADLSMP=0<br>ADCO=1 | T | $I_{DD}$ + $I_{DDAD}$ | — | 218 | — | µA | ADC current only |
| Supply Current | ADLPC=0<br>ADLSMP=1<br>ADCO=1 | T | $I_{DD}$ + $I_{DDAD}$ | — | 327 | — | µA | ADC current only |
| Supply Current | ADLPC=0<br>ADLSMP=0<br>ADCO=1 | D | $I_{DD}$ + $I_{DDAD}$ | — | 0.582 | 1 | mA | ADC current only |
| Supply Current | Stop, Reset, Module Off | | $I_{DD}$ + $I_{DDAD}$ | — | 0.011 | 1 | µA | ADC current only |
| ADC Asynchronous Clock Source | High Speed (ADLPC=0) | P | $f_{ADACK}$ | 2 | 3.3 | 5 | MHz | $t_{ADACK}$ = $1/f_{ADACK}$ |
| | Low Power (ADLPC=1) | | | 1.25 | 2 | 3.3 | | |

**Table A-10. 12-bit ADC Characteristics ($V_{REFH}$ = $V_{DDAD}$, $V_{REFL}$ = $V_{SSAD}$) (continued)**

| Characteristic | Conditions | C | Symb | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|---|
| Conversion Time (Including sample time) | Short Sample (ADLSMP=0) | D | $t_{ADC}$ | — | 20 | — | ADCK cycles | See Table 10-13 for conversion time variances |
| | Long Sample (ADLSMP=1) | | | — | 40 | — | | |
| Sample Time | Short Sample (ADLSMP=0) | D | $t_{ADS}$ | — | 3.5 | — | ADCK cycles | |
| | Long Sample (ADLSMP=1) | | | — | 23.5 | — | | |
| Total Unadjusted Error | 12 bit mode | T | $E_{TUE}$ | — | ±3.0 | ±10 | LSB[2] | Includes quantization |
| | 10 bit mode | P | | — | ±1 | ±2.5 | | |
| | 8 bit mode | T | | — | ±0.5 | ±1.0 | | |
| Differential Non-Linearity | 12 bit mode | T | DNL | — | ±1.75 | ±4.0 | LSB[2] | |
| | 10 bit mode[3] | P | | — | ±0.5 | ±1.0 | | |
| | 8 bit mode[3] | T | | — | ±0.3 | ±0.5 | | |
| Integral Non-Linearity | 12 bit mode | T | INL | — | ±1.5 | ±4.0 | LSB[2] | |
| | 10 bit mode | T | | — | ±0.5 | ±1.0 | | |
| | 8 bit mode | T | | — | ±0.3 | ±0.5 | | |
| Zero-Scale Error | 12 bit mode | T | $E_{ZS}$ | — | ±1.5 | ±6.0 | LSB[2] | $V_{ADIN}$ = $V_{SSAD}$ |
| | 10 bit mode | P | | — | ±0.5 | ±1.5 | | |
| | 8 bit mode | T | | — | ±0.5 | ±0.5 | | |
| Full-Scale Error | 12 bit mode | T | $E_{FS}$ | — | ±1 | ±4.0 | LSB[2] | $V_{ADIN}$ = $V_{DDAD}$ |
| | 10 bit mode | T | | — | ±0.5 | ±1 | | |
| | 8 bit mode | T | | — | ±0.5 | ±0.5 | | |
| Quantization Error | 12 bit mode | D | $E_Q$ | — | -1 to 0 | -1 to 0 | LSB[2] | |
| | 10 bit mode | | | — | — | ±0.5 | | |
| | 8 bit mode | | | — | — | ±0.5 | | |
| Input Leakage Error | 12 bit mode | D | $E_{IL}$ | — | ±1 | ±10.0 | LSB[2] | Pad leakage[4] * $R_{AS}$ |
| | 10 bit mode | | | — | ±0.2 | ±2.5 | | |
| | 8 bit mode | | | — | ±0.1 | ±1 | | |
| Temp Sensor Slope | -40°C– 25°C | D | m | — | 3.266 | — | mV/°C | |
| | 25°C– 125°C | | | — | 3.638 | — | | |
| Temp Sensor Voltage | 25°C | D | $V_{TEMP25}$ | — | 1.396 | — | V | |

1 Typical values assume $V_{DDAD}$ = 5.0V, Temp = 25°C, $f_{ADCK}$=1.0MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

2 1 LSB = ($V_{REFH}$ - $V_{REFL}$)/$2^N$

3 Monotonicity and No-Missing-Codes guaranteed in 10 bit and 8 bit modes

4 Based on input pad leakage current. Refer to pad electricals.

## A.10 External Oscillator (XOSC) Characteristics

**Table A-11. Oscillator Electrical Specifications (Temperature Range = –40 to 125°C Ambient)**

| Num | C | Rating | Symbol | Min | Typ[1] | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | C | Oscillator crystal or resonator (EREFS = 1, ERCLKEN = 1 | | | | | |
| | | Low range (RANGE = 0) | $f_{lo}$ | 32 | — | 38.4 | kHz |
| | | High range (RANGE = 1) FEE or FBE mode [2] | $f_{hi-fll}$ | 1 | — | 5 | MHz |
| | | High range (RANGE = 1) PEE or PBE mode[3] | $f_{hi-pll}$ | 1 | — | 16 | MHz |
| | | High range (RANGE = 1, HGO = 1) BLPE mode | $f_{hi-hgo}$ | 1 | — | 16 | MHz |
| | | High range (RANGE = 1, HGO = 0) BLPE mode | $f_{hi-lp}$ | 1 | — | 8 | MHz |
| 2 | — | Load capacitors | $C_1$ $C_2$ | See crystal or resonator manufacturer's recommendation. | | | |
| 3 | — | Feedback resistor | | | | | |
| | | Low range (32 kHz to 100 kHz) | $R_F$ | — | 10 | — | MΩ |
| | | High range (1 MHz to 16 MHz) | | — | 1 | — | MΩ |
| 4 | — | Series resistor | | | | | |
| | | Low range, low gain (RANGE = 0, HGO = 0) | $R_S$ | — | 0 | — | kΩ |
| | | Low range, high gain (RANGE = 0, HGO = 1) | | — | 100 | — | |
| | | High range, low gain (RANGE = 1, HGO = 0) | | — | 0 | — | |
| | | High range, high gain (RANGE = 1, HGO = 1) ≥ 8 MHz | | — | 0 | 0 | |
| | | 4 MHz | | — | 0 | 10 | |
| | | 1 MHz | | — | 0 | 20 | |
| 5 | T | Crystal start-up time [4] | | | | | |
| | | Low range, low gain (RANGE = 0, HGO = 0) | $t_{CSTL-LP}$ | — | 200 | — | ms |
| | | Low range, high gain (RANGE = 0, HGO = 1) | $t_{CSTL-HGO}$ | — | 400 | — | |
| | | High range, low gain (RANGE = 1, HGO = 0)[5] | $t_{CSTH-LP}$ | — | 5 | — | |
| | | High range, high gain (RANGE = 1, HGO = 1)[4] | $t_{CSTH-HGO}$ | — | 15 | — | |
| 6 | T | Square wave input clock frequency (EREFS = 0, ERCLKEN = 1) | | | | | |
| | | FEE or FBE mode [2] | $f_{extal}$ | 0.03125 | — | 5 | MHz |
| | | PEE or PBE mode[3] | | 1 | — | 16 | |
| | | BLPE mode | | 0 | — | 40 | |

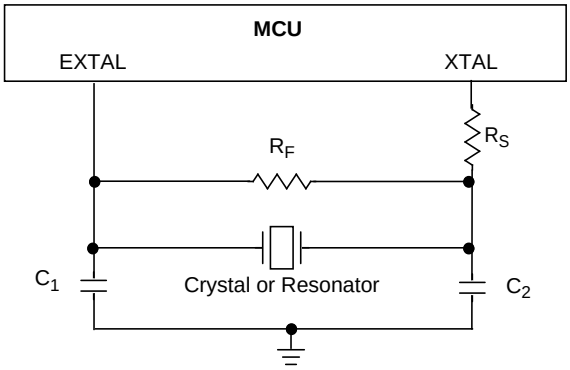
1 Typical data was characterized at 3.0 V, 25°C or is recommended value.

2 When MCG is configured for FEE or FBE mode, the input clock source must be divisible using RDIV to within the range of 31.25 kHz to 39.0625 kHz.

3 When MCG is configured for PEE or PBE mode, input clock source must be divisible using RDIV to within the range of 1 MHz to 2 MHz.

4 This parameter is characterized and not tested on each device. Proper PC board layout procedures must be followed to achieve specifications. This data will vary based upon the crystal manufacturer and board design. The crystal should be characterized by the crystal manufacturer.

5 4 MHz crystal.




## A.11 MCG Specifications

**Table A-12. MCG Frequency Specifications (Temperature Range = –40 to 125°C Ambient)**

| Num | C | Rating | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | P | Internal reference frequency - factory trimmed at $V_{DD}$ = 5 V and temperature = 25 °C | $f_{int_ft}$ | — | 31.25 | — | kHz |
| 2 | P | Average internal reference frequency - untrimmed [1] | $f_{int_ut}$ | 25 | 32.7 | 41.66 | kHz |
| 3 | P | Average internal reference frequency - user trimmed | $f_{int_t}$ | 31.25 | — | 39.0625 | kHz |
| 4 | D | Internal reference startup time | $t_{irefst}$ | — | 60 | 100 | us |
| 5 | — | DCO output frequency range - untrimmed [1] value provided for reference: $f_{dco_ut}$ = 1024 X $f_{int_ut}$ | $f_{dco_ut}$ | 25.6 | 33.48 | 42.66 | MHz |
| 6 | P | DCO output frequency range - trimmed | $f_{dco_t}$ | 32 | — | 40 | MHz |
| 7 | C | Resolution of trimmed DCO output frequency at fixed voltage and temperature (using FTRIM) | $\Delta f_{dco_res_t}$ | — | ± 0.1 | ± 0.2 | $\%f_{dco}$ |
| 8 | C | Resolution of trimmed DCO output frequency at fixed voltage and temperature (not using FTRIM) | $\Delta f_{dco_res_t}$ | — | ± 0.2 | ± 0.4 | $\%f_{dco}$ |
| 9 | P | Total deviation of trimmed DCO output frequency over voltage and temperature | $\Delta f_{dco_t}$ | — | + 0.5<br>-1.0 | ± 2 | $\%f_{dco}$ |
| 10 | C | Total deviation of trimmed DCO output frequency over fixed voltage and temperature range of 0 - 70 °C | $\Delta f_{dco_t}$ | — | ± 0.5 | ± 1 | $\%f_{dco}$ |
| 11 | C | FLL acquisition time [2] | $t_{fll_acquire}$ | — | — | 1 | ms |
| 12 | D | PLL acquisition time [3] | $t_{pll_acquire}$ | — | — | 1 | ms |
| 13 | C | Long term Jitter of DCO output clock (averaged over 2ms interval) [4] | $C_{Jitter}$ | — | 0.02 | 0.2 | $\%f_{dco}$ |
| 14 | D | VCO operating frequency | $f_{vco}$ | 7.0 | — | 55.0 | MHz |
| 15 | D | PLL reference frequency range | $f_{pll_ref}$ | 1.0 | — | 2.0 | MHz |
| 16 | T | RMS frequency variation of a single clock cycle measured 2 ms after reference edge.[5] | $f_{pll_cycjit_2ms}$ | — | 0.590[4] | — | $\%f_{pll}$ |
| 17 | T | Maximum frequency variation averaged over 2 ms window. | $f_{pll_maxjit_2ms}$ | — | 0.001 | — | $\%f_{pll}$ |

**Table A-12. MCG Frequency Specifications (Temperature Range = –40 to 125°C Ambient) (continued)**

| Num | C | Rating | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|---|---|
| 18 | T | RMS frequency variation of a single clock cycle measured 625 ns after reference edge.[6] | $f_{pll_cycjit_625ns}$ | — | 0.566[4] | — | $\%f_{pll}$ |
| 19 | T | Maximum frequency variation averaged over 625 ns window. | $f_{pll_maxjit_625ns}$ | — | 0.113 | — | $\%f_{pll}$ |
| 20 | D | Lock entry frequency tolerance [7] | $D_{lock}$ | ± 1.49 | — | ± 2.98 | % |
| 21 | D | Lock exit frequency tolerance [8] | $D_{unl}$ | ± 4.47 | — | ± 5.97 | % |
| 22 | D | Lock time - FLL | $t_{fll_lock}$ | — | — | $t_{fll_acquire}$+ 1075(1/$f_{int_t}$) | s |
| 23 | D | Lock time - PLL | $t_{pll_lock}$ | — | — | $t_{pll_acquire}$+ 1075(1/$f_{pll_ref}$) | s |
| 24 | D | Loss of external clock minimum frequency - RANGE = 0 | $f_{loc_low}$ | (3/5) x $f_{int}$ | — | — | kHz |
| 25 | D | Loss of external clock minimum frequency - RANGE = 1 | $f_{loc_high}$ | (16/5) x $f_{int}$ | — | — | kHz |

1 TRIM register at default value (0x80) and FTRIM control bit at default value (0x0).

2 This specification applies to any time the FLL reference source or reference divider is changed, trim value changed or changing from FLL disabled (BLPE, BLPI) to FLL enabled (FEI, FEE, FBE, FBI). If a crystal/resonator is being used as the reference, this specification assumes it is already running.

3 This specification applies to any time the PLL VCO divider or reference divider is changed, or changing from PLL disabled (BLPE, BLPI) to PLL enabled (PBE, PEE). If a crystal/resonator is being used as the reference, this specification assumes it is already running.

4 Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum $f_{BUS}$. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the FLL circuitry via $V_{DD}$ and $V_{SS}$ and variation in crystal oscillator frequency increase the $C_{Jitter}$ percentage for a given interval. Jitter measurements are based upon a 40MHz MCGOUT clock frequency.

5 In some specifications, this value is described as “long term accuracy of PLL output clock (averaged over 2 ms)” with symbol “$f_{pll_jitter_2ms}$.” The parameter is unchanged, but the description has been changed for clarification purposes.

6 In some specifications, this value is described as “Jitter of PLL output clock measured over 625 ns” with symbol “$f_{pll_jitter_625ns}$.” The parameter is unchanged, but the description has been changed for clarification purposes.

7 Below $D_{lock}$ minimum, the MCG is guaranteed to enter lock. Above $D_{lock}$ maximum, the MCG will not enter lock. But if the MCG is already in lock, then the MCG may stay in lock.

8 Below $D_{unl}$ minimum, the MCG will not exit lock if already in lock. Above $D_{unl}$ maximum, the MCG is guaranteed to exit lock.

## A.12 AC Characteristics

This section describes ac timing characteristics for each peripheral system.

### A.12.1 Control Timing

**Table A-13. Control Timing**

| Num | C | Rating | Symbol | Min | Typical[1] | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D/P | Bus frequency ($t_{cyc}$ = 1/$f_{Bus}$) | $f_{Bus}$ | dc | — | 20 | MHz |
| 2 | T | Internal low-power oscillator period | $t_{LPO}$ | — | 1500 | — | µs |
| 3 | D | External reset pulse width[2] | $t_{extrst}$ | 1.5 x $t_{cyc}$ | | — | ns |
| 4 | D | Reset low drive[3] | $t_{rstdrv}$ | 34 x $t_{cyc}$ | | — | ns |
| 5 | D | Active background debug mode latch setup time | $t_{MSSU}$ | 25 | | — | ns |
| 6 | D | Active background debug mode latch hold time | $t_{MSH}$ | 25 | | — | ns |
| 7 | D | IRQ/PIAx/ PIBx/PIDx pulse width<br>Asynchronous path[2]<br>Synchronous path[3] | $t_{ILIH}$, $t_{IHIL}$ | <br>100<br>1.5 $t_{cyc}$ | — | — | ns |
| 8 | T | Port rise and fall time —<br>Low output drive (PTxDS = 0) (load = 50 pF)[4]<br>Slew rate control disabled (PTxSE = 0)<br>Slew rate control enabled (PTxSE = 1) | $t_{Rise}$, $t_{Fall}$ | <br><br>—<br>— | <br><br>40<br>75 | | ns |
| | | Port rise and fall time —<br>High output drive (PTxDS = 1) (load = 50 pF)[4]<br>Slew rate control disabled (PTxSE = 0)<br>Slew rate control enabled (PTxSE = 1) | $t_{Rise}$, $t_{Fall}$ | <br><br>—<br>— | <br><br>11<br>35 | | ns |

1 Typical data was characterized at 5.0 V, 25°C unless otherwise stated.

2 This is the shortest pulse that is guaranteed to be recognized as a reset pin request. Shorter pulses are not guaranteed to override reset requests from internal sources.

3 When any reset is initiated, internal circuitry drives the $\overline{\text{RESET}}$ pin low for about 34 cycles of $t_{cyc}$. After POR reset, the bus clock frequency changes to the untrimmed DCO frequency (freset = ($f_{dco_ut}$)/4) because TRIM is reset to 0x80 and FTRIM is reset to 0; and there is an extra divide-by-two because BDIV is reset to 0:1. After other resets, trim stays at the pre-reset value.

4 Timing is shown with respect to 20% $V_{DD}$ and 80% $V_{DD}$ levels. Temperature range –40°C to 125°C.

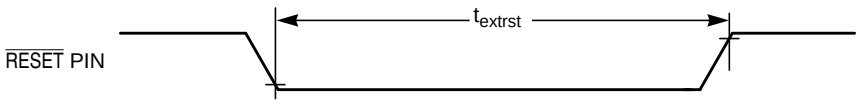



**Figure A-2. Reset Timing**

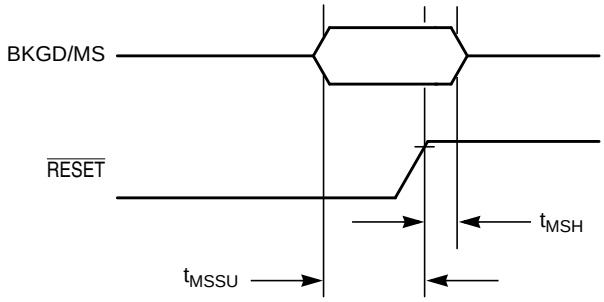



**Figure A-3. Active Background Debug Mode Latch Timing**

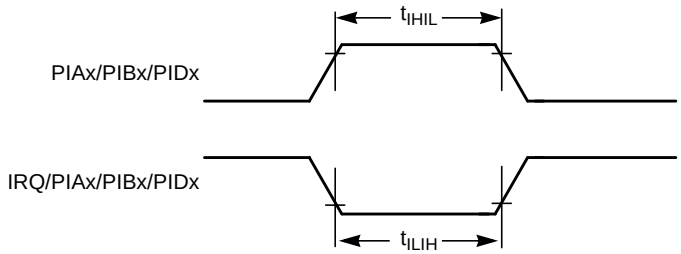



**Figure A-4. Pin Interrupt Timing**

## A.12.2 Timer/PWM

Synchronizer circuits determine the shortest input pulses that can be recognized or the fastest clock that can be used as the optional external source to the timer counter. These synchronizers operate from the current bus rate clock.

**Table A-14. TPM Input Timing**

| Num | C | Rating | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | — | External clock frequency | $f_{TCLK}$ | dc | $f_{Bus}/4$ | MHz |
| 2 | — | External clock period | $t_{TCLK}$ | 4 | — | $t_{cyc}$ |
| 3 | D | External clock high time | $t_{clkh}$ | 1.5 | — | $t_{cyc}$ |
| 4 | D | External clock low time | $t_{clkl}$ | 1.5 | — | $t_{cyc}$ |
| 5 | D | Input capture pulse width | $t_{ICPW}$ | 1.5 | — | $t_{cyc}$ |

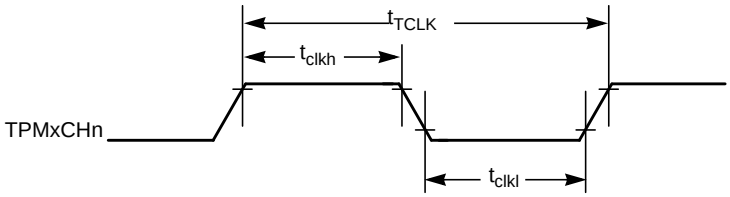



**Figure A-5. Timer External Clock**

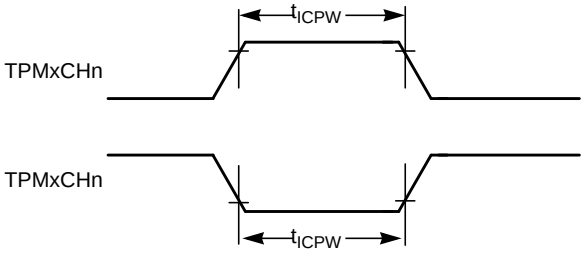



**Figure A-6. Timer Input Capture Pulse**

## A.12.3 MSCAN

**Table A-15. MSCAN Wake-up Pulse Characteristics**

| Num | C | Rating | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | D | MSCAN Wake-up dominant pulse filtered | $t_{WUP}$ | — | — | 2 | µs |
| 2 | D | MSCAN Wake-up dominant pulse pass | $t_{WUP}$ | 5 | — | — | µs |

## A.12.4 SPI

Table A-16 and Figure A-7 through Figure A-10 describe the timing requirements for the SPI system.

**Table A-16. SPI Electrical Characteristic**

| Num[1] | C | Rating[2] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | D | Cycle time<br>Master<br>Slave | <br>$t_{SCK}$<br>$t_{SCK}$ | <br>2<br>4 | <br>2048<br>— | <br>$t_{cyc}$<br>$t_{cyc}$ |
| 2 | D | Enable lead time<br>Master<br>Slave | <br>$t_{Lead}$<br>$t_{Lead}$ | <br>—<br>1/2 | <br>1/2<br>— | <br>$t_{SCK}$<br>$t_{SCK}$ |
| 3 | D | Enable lag time<br>Master<br>Slave | <br>$t_{Lag}$<br>$t_{Lag}$ | <br>—<br>1/2 | <br>1/2<br>— | <br>$t_{SCK}$<br>$t_{SCK}$ |
| 4 | D | Clock (SPSCK) high time<br>Master and Slave | $t_{SCKH}$ | (1/2 $t_{SCK}$)– 25 | — | ns |
| 5 | D | Clock (SPSCK) low time<br>Master and Slave | $t_{SCKL}$ | (1/2 $t_{SCK}$) – 25 | — | ns |
| 6 | D | Data setup time (inputs)<br>Master<br>Slave | <br>$t_{SI(M)}$<br>$t_{SI(S)}$ | <br>30<br>30 | <br>—<br>— | <br>ns<br>ns |
| 7 | D | Data hold time (inputs)<br>Master<br>Slave | <br>$t_{HI(M)}$<br>$t_{HI(S)}$ | <br>30<br>30 | <br>—<br>— | <br>ns<br>ns |
| 8 | D | Access time, slave[3] | $t_A$ | 0 | 40 | ns |
| 9 | D | Disable time, slave[4] | $t_{dis}$ | — | 40 | ns |
| 10 | D | Data setup time (outputs)<br>Master<br>Slave | <br>$t_{SO}$<br>$t_{SO}$ | <br>25<br>25 | <br>—<br>— | <br>ns<br>ns |
| 11 | D | Data hold time (outputs)<br>Master<br>Slave | <br>$t_{HO}$<br>$t_{HO}$ | <br>–10<br>–10 | <br>—<br>— | <br>ns<br>ns |
| 12 | D | Operating frequency[5]<br>Master<br>Slave | <br>$f_{op}$<br>$f_{op}$ | <br>$f_{Bus}$/2048<br>dc | <br>5<br>$f_{Bus}$/4 | <br>MHz |

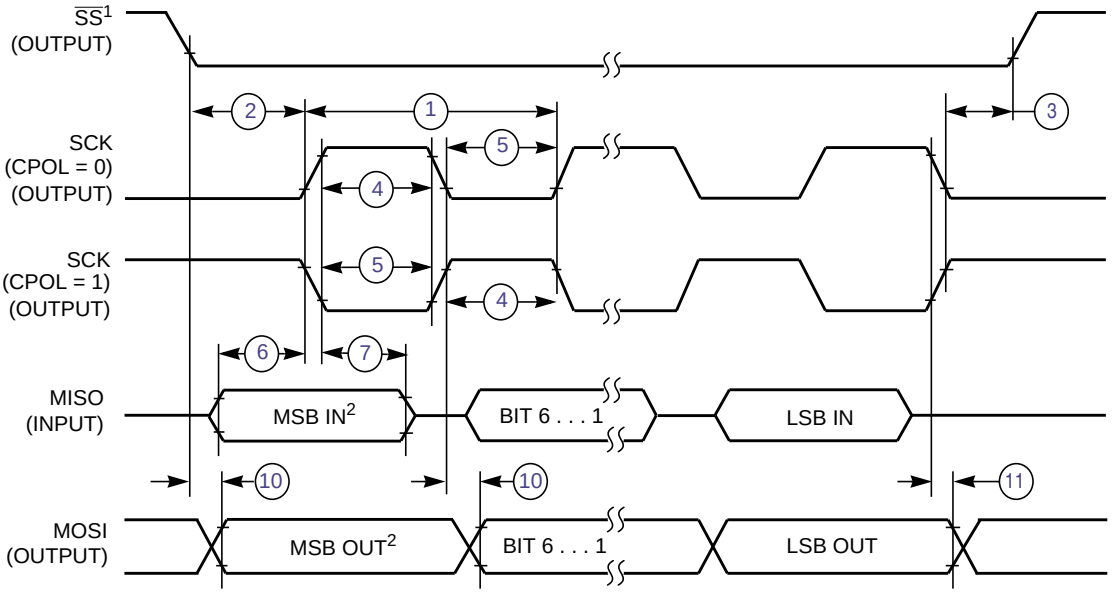
1 Refer to Figure A-7 through Figure A-10.

2 All timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$, unless noted; 100 pF load on all SPI pins. All timing assumes slew rate control disabled and high drive strength enabled for SPI output pins.

3 Time to data active from high-impedance state.

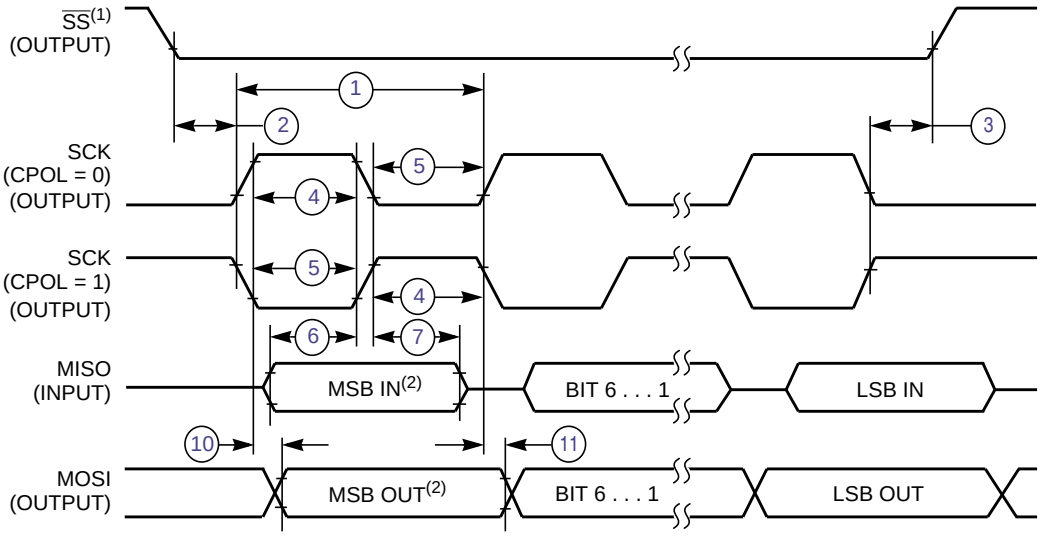
4 Hold time to high-impedance state.

5 Maximum baud rate must be limited to 5 MHz due to pad input characteristics.



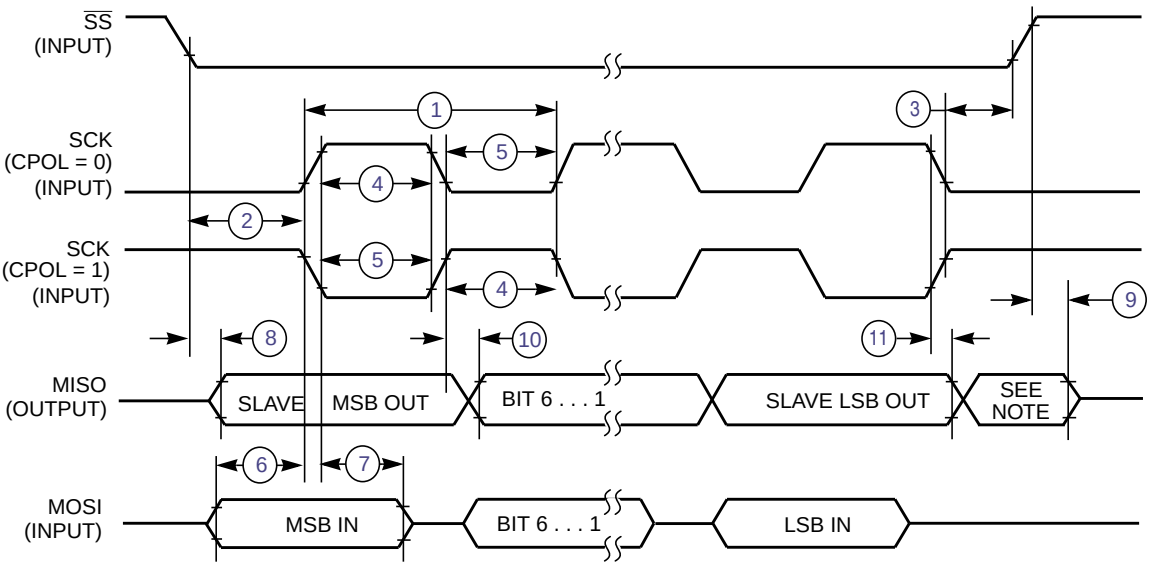

NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure A-7. SPI Master Timing (CPHA = 0)**



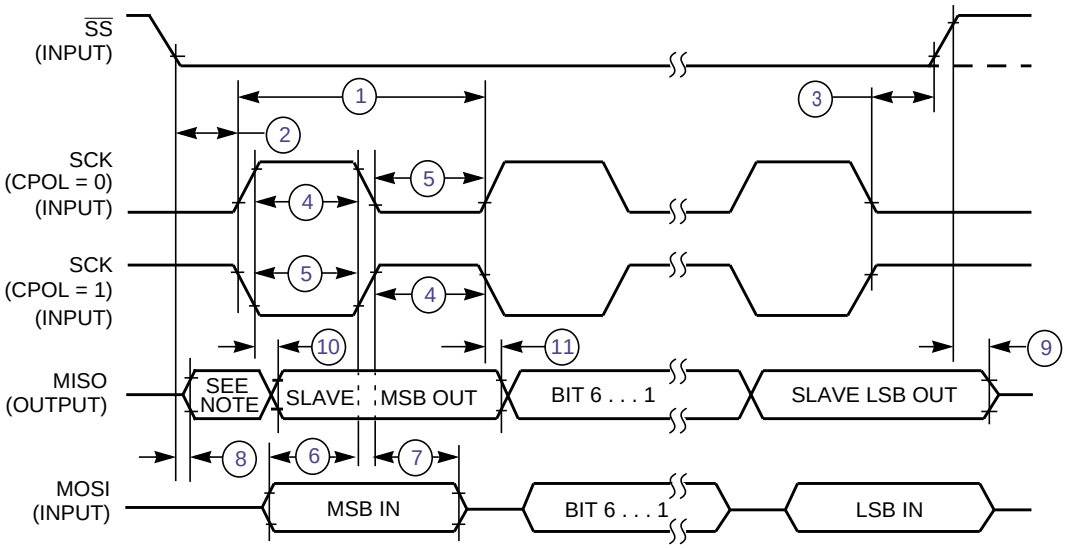

NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure A-8. SPI Master Timing (CPHA = 1)**




NOTE:
1. Not defined but normally MSB of character just received

**Figure A-9. SPI Slave Timing (CPHA = 0)**




NOTE:
1. Not defined but normally LSB of character just received

**Figure A-10. SPI Slave Timing (CPHA = 1)**

## A.13 Flash and EEPROM

This section provides details about program/erase times and program-erase endurance for the Flash and EEPROM memory.

Program and erase operations do not require any special power sources other than the normal $V_{DD}$ supply. For more detailed information about program/erase operations, see Chapter 4, "Memory."

**Table A-17. Flash and EEPROM Characteristics**

| Num | C | Rating | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|---|---|
| 1 | — | Supply voltage for program/erase | $V_{prog/erase}$ | 2.7 | | 5.5 | V |
| 2 | — | Supply voltage for read operation<br>$0 < f_{Bus} < 8$ MHz<br>$0 < f_{Bus} < 20$ MHz | $V_{Read}$ | 2.7 | | 5.5 | V |
| 3 | — | Internal FCLK frequency[1] | $f_{FCLK}$ | 150 | | 200 | kHz |
| 4 | — | Internal FCLK period (1/FCLK) | $t_{Fcyc}$ | 5 | | 6.67 | µs |
| 5 | — | Byte program time (random location)(2) | $t_{prog}$ | 9 | | | $t_{Fcyc}$ |
| 6 | — | Byte program time (burst mode)(2) | $t_{Burst}$ | 4 | | | $t_{Fcyc}$ |
| 7 | — | Page erase time[2] | $t_{Page}$ | 4000 | | | $t_{Fcyc}$ |
| 8 | — | Mass erase time(2) | $t_{Mass}$ | 20,000 | | | $t_{Fcyc}$ |
| 9 | C | Flash Program/erase endurance[3]<br>$T_L$ to $T_H$ = –40°C to + 125°C<br>T = 25°C | $n_{FLPE}$ | 10,000<br>— | —<br>100,000 | —<br>— | cycles |
| 10 | C | EEPROM Program/erase endurance[3]<br>$T_L$ to $T_H$ = –40°C to + 0°C<br>$T_L$ to $T_H$ = 0°C to + 125°C<br>T = 25°C | $n_{EEPE}$ | 10,000<br>50,000<br>— | —<br>—<br>100,000 | —<br>—<br>— | cycles |
| 11 | C | Data retention[4] | $t_{D_ret}$ | 15 | 100 | — | years |

1 The frequency of this clock is controlled by a software setting.

2 These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.

3 **Typical endurance** for Flash and EEPROM is based on the intrinsic bit cell performance. For additional information on how Freescale Semiconductor defines typical endurance, please refer to Engineering Bulletin EB619, *Typical Endurance for Nonvolatile Memory.*

4 **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 Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618, *Typical Data Retention for Nonvolatile Memory.*

## A.14 EMC Performance

Electromagnetic compatibility (EMC) performance is highly dependant on the environment in which the MCU resides. Board design and layout, circuit topology choices, location and characteristics of external components as well as MCU software operation all play a significant role in EMC performance. The system designer should consult Freescale applications notes such as AN2321, AN1050, AN1263, AN2764, and AN1259 for advice and guidance specifically targeted at optimizing EMC performance.

### A.14.1 Radiated Emissions

Microcontroller radiated RF emissions are measured from 150 kHz to 1 GHz using the TEM/GTEM Cell method in accordance with the IEC 61967-2 and SAE J1752/3 standards. The measurement is performed with the microcontroller installed on a custom EMC evaluation board while running specialized EMC test software. The radiated emissions from the microcontroller are measured in a TEM cell in two package orientations (North and East). For more detailed information concerning the evaluation results, conditions and setup, please refer to the EMC Evaluation Report for this device.

The maximum radiated RF emissions of the tested configuration in all orientations are less than or equal to the reported emissions levels.

**Table A-18. Radiated Emissions for 3M05C Mask Set**

| Parameter | Symbol | Conditions | Frequency | $f_{osc}/f_{CPU}$ | Level[1] (Max) | Unit |
|---|---|---|---|---|---|---|
| Radiated emissions, electric field — Conditions - | $V_{RE_TEM}$ | $V_{DD}$ = 5<br>$T_A$ = +25°C<br>64 LQFP | 0.15 – 50 MHz | 16 MHz Crystal<br>20 MHz Bus | 18 | dBµV |
| | | | 50 – 150 MHz | | 18 | |
| | | | 150 – 500 MHz | | 13 | |
| | | | 500 – 1000 MHz | | 7 | |
| | | | IEC Level | | L | — |
| | | | SAE Level | | 2 | — |

1 Data based on qualification test results.

# Appendix B
# Timer Pulse-Width Modulator (TPMV2)

**NOTE**

This chapter refers to S08TPM version 2, which applies to the 3M05C and older mask sets of this device. )M74K and newer mask set devices use S08TPM version 3. If your device uses mask 0M74K or newer, please refer to Chapter 16, “Timer Pulse-Width Modulator (S08TPMV3) for information pertaining to that module.

The TPM uses one input/output (I/O) pin per channel, TPMxCHn where x is the TPM number (for example, 1 or 2) and n is the channel number (for example, 0–4). The TPM shares its I/O pins with general-purpose I/O port pins (refer to the Pins and Connections chapter for more information).

## B.0.1 Features

The TPM has the following features:

- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock sources independently selectable per TPM (multiple TPMs device)
- Selectable clock sources (device dependent): bus clock, fixed system clock, external pin
- Clock prescaler taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Timer system enable
- One interrupt per channel plus a terminal count interrupt for each TPM module (multiple TPMs device)
- Channel features:
  - Each channel may be input capture, output compare, or buffered edge-aligned PWM
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs

## B.0.2 Block Diagram

Figure B-1 shows the structure of a TPM. Some MCUs include more than one TPM, with various numbers of channels.

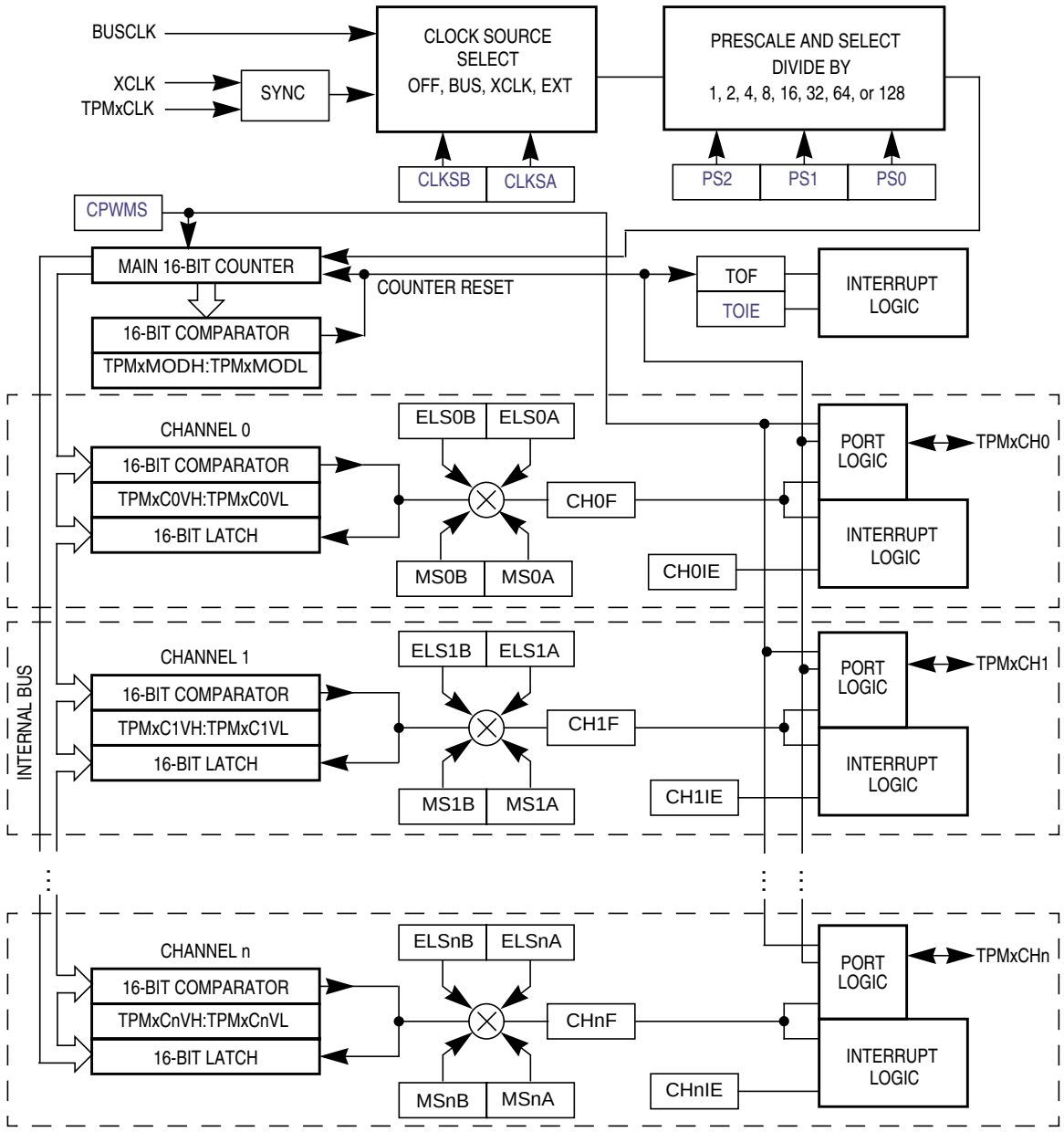



**Figure B-1. TPM Block Diagram**

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter. (The values 0x0000 or 0xFFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the TPMxCNT counter resets the counter regardless of the data value written.

All TPM channels are programmable independently as input capture, output compare, or buffered edge-aligned PWM channels.

## B.1 External Signal Description

When any pin associated with the timer is configured as a timer input, a passive pullup can be enabled. After reset, the TPM modules are disabled and all pins default to general-purpose inputs with the passive pullups disabled.

### B.1.1 External TPM Clock Sources

When control bits CLKSB:CLKSA in the timer status and control register are set to 1:1, the prescaler and consequently the 16-bit counter for TPMx are driven by an external clock source, TPMxCLK, connected to an I/O pin. A synchronizer is needed between the external clock and the rest of the TPM. This synchronizer is clocked by the bus clock so the frequency of the external source must be less than one-half the frequency of the bus rate clock. The upper frequency limit for this external clock source is specified to be one-fourth the bus frequency to conservatively accommodate duty cycle and phase-locked loop (PLL) or frequency-locked loop (FLL) frequency jitter effects.

On some devices the external clock input is shared with one of the TPM channels. When a TPM channel is shared as the external clock input, the associated TPM channel cannot use the pin. (The channel can still be used in output compare mode as a software timer.) Also, if one of the TPM channels is used as the external clock input, the corresponding ELSnB:ELSnA control bits must be set to 0:0 so the channel is not trying to use the same pin.

### B.1.2 TPMxCHn — TPMx Channel n I/O Pins

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the configuration of the channel. In some cases, no pin function is needed so the pin reverts to being controlled by general-purpose I/O controls. When a timer has control of a port pin, the port data and data direction registers do not affect the related pin(s). See the Pins and Connections chapter for additional information about shared pin functions.

## B.2 Register Definition

The TPM includes:

- An 8-bit status and control register (TPMxSC)
- A 16-bit counter (TPMxCNTH:TPMxCNTL)
- A 16-bit modulo register (TPMxMODH:TPMxMODL)

Each timer channel has:

- An 8-bit status and control register (TPMxCnSC)
- A 16-bit channel value register (TPMxCnVH:TPMxCnVL)

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all TPM registers. This section refers to registers and control bits only by their names. A

Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

## B.2.1 Timer Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.

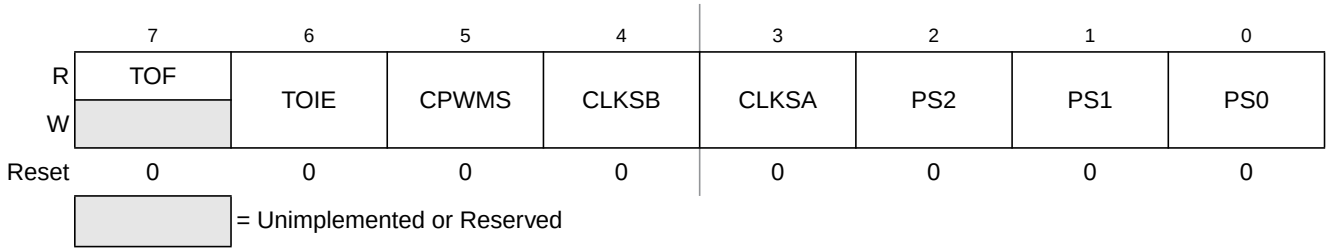



**Figure B-2. Timer Status and Control Register (TPMxSC)**

**Table B-1. TPMxSC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — This flag is set when the TPM counter changes to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value. Clear TOF by reading the TPM status and control register when TOF is set and then writing a 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. Reset clears TOF. Writing a 1 to TOF has no effect.<br>0 TPM counter has not reached modulo value or overflow<br>1 TPM counter has overflowed |
| 6<br>TOIE | **Timer Overflow Interrupt Enable** — This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals 1. Reset clears TOIE.<br>0 TOF interrupts inhibited (use software polling)<br>1 TOF interrupts enabled |
| 5<br>CPWMS | **Center-Aligned PWM Select** — This read/write bit selects CPWM operating mode. Reset clears this bit so the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up-/down-counting mode for CPWM functions. Reset clears CPWMS.<br>0 All TPMx channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register<br>1 All TPMx channels operate in center-aligned PWM mode |
| 4:3<br>CLKS[B:A] | **Clock Source Select** — As shown in Table B-2, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The external source and the XCLK are synchronized to the bus clock by an on-chip synchronization circuit. |
| 2:0<br>PS[2:0] | **Prescale Divisor Select** — This 3-bit field selects one of eight divisors for the TPM clock input as shown in Table B-3. This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system. |

**Table B-2. TPM Clock Source Selection**

| CLKSB:CLKSA | TPM Clock Source to Prescaler Input |
|---|---|
| 0:0 | No clock selected (TPMx disabled) |
| 0:1 | Bus rate clock (BUSCLK) |
| 1:0 | Fixed system clock (XCLK) |
| 1:1 | External source (TPMxCLK)[1,2] |

1 The maximum frequency that is allowed as an external clock is one-fourth of the bus frequency.

2 If the external clock input is shared with channel n and is selected as the TPM clock source, the corresponding ELSnB:ELSnA control bits should be set to 0:0 so channel n does not try to use the same pin for a conflicting function.

**Table B-3. Prescale Divisor Selection**

| PS2:PS1:PS0 | TPM Clock Source Divided-By |
|---|---|
| 0:0:0 | 1 |
| 0:0:1 | 2 |
| 0:1:0 | 4 |
| 0:1:1 | 8 |
| 1:0:0 | 16 |
| 1:0:1 | 32 |
| 1:1:0 | 64 |
| 1:1:1 | 128 |

## B.2.2 Timer Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This allows coherent 16-bit reads in either order. The coherency mechanism is automatically restarted by an MCU reset, a write of any value to TPMxCNTH or TPMxCNTL, or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| W | Any write to TPMxCNTH clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure B-3. Timer Counter Register High (TPMxCNTH)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| W | Any write to TPMxCNTL clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure B-4. Timer Counter Register Low (TPMxCNTL)**

When background mode is active, the timer counter and the coherency mechanism are frozen such that the buffer latches remain in the state they were in when the background mode became active even if one or both bytes of the counter are read while background mode is active.

## B.2.3 Timer Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits TOF and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000, which results in a free-running timer counter (modulo disabled).

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure B-5. Timer Counter Modulo Register High (TPMxMODH)**

| | 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 B-6. Timer Counter Modulo Register Low (TPMxMODL)**

It is good practice to wait for an overflow interrupt so both bytes of the modulo register can be written well before a new overflow. An alternative approach is to reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

## B.2.4 Timer Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel interrupt status flag and control bits that are used to configure the interrupt enable, channel configuration, and pin function.

**Figure B-7. Timer Channel n Status and Control Register (TPMxCnSC)**

**Table B-4. TPMxCnSC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CHnF | **Channel n Flag** — When channel n is configured for input capture, this flag bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. This flag is seldom used with center-aligned PWMs because it is set every time the counter matches the channel value register, which correspond to both edges of the active duty cycle period.<br>A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while CHnF is set and then writing a 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF would remain set after the clear sequence was completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost by clearing a previous CHnF. Reset clears CHnF. Writing a 1 to CHnF has no effect.<br>0 No input capture or output compare event occurred on channel n<br>1 Input capture or output compare event occurred on channel n |
| 6<br>CHnIE | **Channel n Interrupt Enable** — This read/write bit enables interrupts from channel n. Reset clears CHnIE.<br>0 Channel n interrupt requests disabled (use software polling)<br>1 Channel n interrupt requests enabled |
| 5<br>MSnB | **Mode Select B for TPM Channel n** — When CPWMS = 0, MSnB = 1 configures TPM channel n for edge-aligned PWM mode. For a summary of channel mode and setup controls, refer to Table B-5. |
| 4<br>MSnA | **Mode Select A for TPM Channel n** — When CPWMS = 0 and MSnB = 0, MSnA configures TPM channel n for input capture mode or output compare mode. Refer to Table B-5 for a summary of channel mode and setup controls. |
| 3:2<br>ELSn[B:A] | **Edge/Level Select Bits** — Depending on the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table B-5, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output.<br>Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general-purpose I/O pin unrelated to any timer channel functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general-purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin. |

**Table B-5. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| X | XX | 00 | Pin not used for TPM channel; use as an external clock for the TPM or revert to general-purpose I/O | |
| 0 | 00 | 01 | Input capture | Capture on rising edge only |
| | | 10 | | Capture on falling edge only |
| | | 11 | | Capture on rising or falling edge |
| | 01 | 00 | Output compare | Software compare only |
| | | 01 | | Toggle output on compare |
| | | 10 | | Clear output on compare |
| | | 11 | | Set output on compare |
| | 1X | 10 | Edge-aligned PWM | High-true pulses (clear output on compare) |
| | | X1 | | Low-true pulses (set output on compare) |
| 1 | XX | 10 | Center-aligned PWM | High-true pulses (clear output on compare-up) |
| | | X1 | | Low-true pulses (set output on compare-up) |

If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts or using the status flags to avoid any unexpected behavior.

## B.2.5 Timer Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel value registers are cleared by reset.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure B-8. Timer Channel Value Register High (TPMxCnVH)**

| | 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 B-9. Timer Channel Value Register Low (TPMxCnVL)**

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the timer channel value registers. This latching mechanism may be manually reset by writing to the TPMxCnSC register.

This latching mechanism allows coherent 16-bit writes in either order, which is friendly to various compiler implementations.

## B.3 Functional Description

All TPM functions are associated with a main 16-bit counter that allows flexible selection of the clock source and prescale divisor. A 16-bit modulo register also is associated with the main 16-bit counter in the TPM. Each TPM channel is optionally associated with an MCU pin and a maskable interrupt function.

The TPM has center-aligned PWM capabilities controlled by the CPWMS control bit in TPMxSC. When CPWMS is set to 1, timer counter TPMxCNT changes to an up-/down-counter and all channels in the associated TPM act as center-aligned PWM channels. When CPWMS = 0, each channel can independently be configured to operate in input capture, output compare, or buffered edge-aligned PWM mode.

The following sections describe the main 16-bit counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend on the operating mode, these topics are covered in the associated mode sections.

### B.3.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, up-counting vs. up-/down-counting, end-of-count overflow, and manual counter reset.

After any MCU reset, CLKSB:CLKSA = 0:0 so no clock source is selected and the TPM is inactive. Normally, CLKSB:CLKSA would be set to 0:1 so the bus clock drives the timer counter. The clock source for the TPM can be selected to be off, the bus clock (BUSCLK), the fixed system clock (XCLK), or an external input. The maximum frequency allowed for the external clock option is one-fourth the bus rate. Refer to Section B.2.1, “Timer Status and Control Register (TPMxSC)” and Table B-2 for more information about clock source selection.

When the microcontroller is in active background mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all TPM clocks are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally.

The main 16-bit counter has two counting modes. When center-aligned PWM is selected (CPWMS = 1), the counter operates in up-/down-counting mode. Otherwise, the counter operates as a simple up-counter.

As an up-counter, the main 16-bit counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts upward from 0x0000 through its terminal count and then counts downward to 0x0000 where it returns to up-counting. Both 0x0000 and the terminal count value (value in TPMxMODH:TPMxMODL) are normal length counts (one timer clock period long).

An interrupt flag and enable are associated with the main 16-bit counter. The timer overflow flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE = 0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE = 1) where a static hardware interrupt is automatically generated whenever the TOF flag is 1.

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the main 16-bit counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the main 16-bit counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

Because the HCS08 MCU is an 8-bit architecture, a coherency mechanism is built into the timer counter for read operations. Whenever either byte of the counter is read (TPMxCNTH or TPMxCNTL), both bytes are captured into a buffer so when the other byte is read, the value will represent the other byte of the count at the time the first byte was read. The counter continues to count normally, but no new value can be read from either byte until both bytes of the old count have been read.

The main timer counter can be reset manually at any time by writing any value to either byte of the timer count TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only one byte of the counter was read before resetting the count.

## B.3.2 Channel Mode Selection

Provided CPWMS = 0 (center-aligned PWM operation is not specified), the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and buffered edge-aligned PWM.

### B.3.2.1 Input Capture Mode

With the input capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TPM latches the contents of the TPM counter into the channel value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either byte of the 16-bit capture register is read, both bytes are latched into a buffer to support coherent 16-bit accesses regardless of order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

### B.3.2.2 Output Compare Mode

With the output compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel value registers of an output compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel value registers only after both 8-bit bytes of a 16-bit register have been written. This coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

### B.3.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS = 0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the setting in the modulus register (TPMxMODH:TPMxMODL). The duty cycle is determined by the setting in the timer channel value register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As Figure B-10 shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. The time between the modulus overflow and the output compare is the pulse width. If ELSnA = 0, the counter overflow forces the PWM signal high and the output compare forces the PWM signal low. If ELSnA = 1, the counter overflow forces the PWM signal low and the output compare forces the PWM signal high.

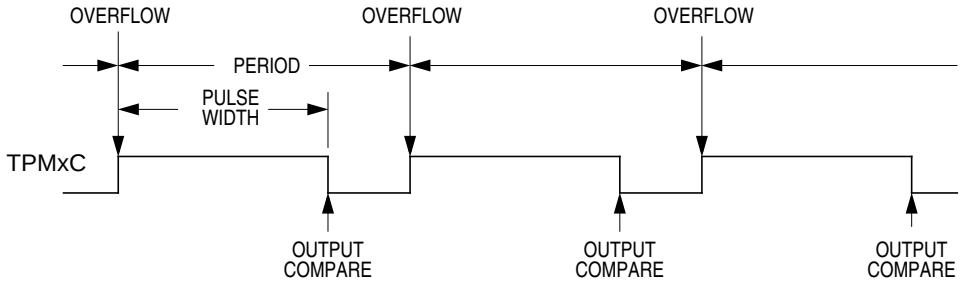



**Figure B-10. PWM Period and Pulse Width (ELSnA = 0)**

When the channel value register is set to 0x0000, the duty cycle is 0 percent. By setting the timer channel value register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting, 100% duty cycle can be achieved. This implies that the modulus setting must be less than 0xFFFF to get 100% duty cycle.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to either register, TPMxCnVH or TPMxCnVL, write to buffer registers. In edge-PWM mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and

the value in the TPMxCNTH:TPMxCNTL counter is 0x0000. (The new duty cycle does not take effect until the next full period.)

### B.3.3 Center-Aligned PWM Mode

This type of PWM output uses the up-/down-counting mode of the timer counter (CPWMS = 1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPMxCnVH:TPMxCnVL}) \qquad \textit{Eqn. 17-1}$$

$$\text{period} = 2 \times (\text{TPMxMODH:TPMxMODL}); \text{ for TPMxMODH:TPMxMODL} = 0x0001–0x7FFF \qquad \textit{Eqn. 17-2}$$

If the channel value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if generation of 100% duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

TPMxMODH:TPMxMODL = 0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS = 0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS = 1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

Figure B-11 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If ELSnA = 0, the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

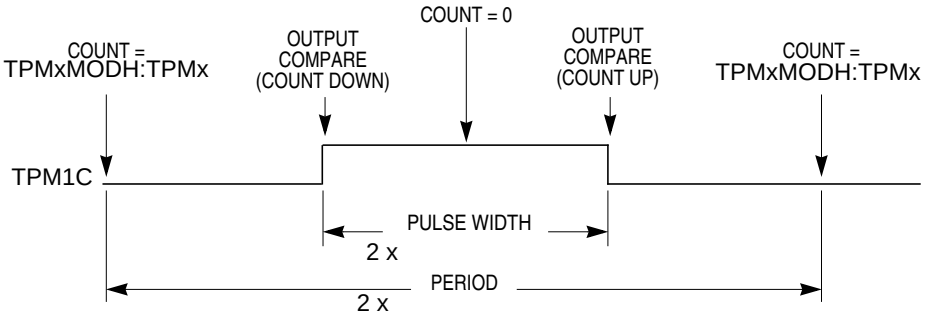



**Figure B-11. CPWM Period and Pulse Width (ELSnA = 0)**

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers, TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers. Values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the timer counter overflows (reverses direction from up-counting to down-counting at the end of the terminal count in the modulus register). This TPMxCNT overflow requirement only applies to PWM channels, not output compares.

Optionally, when TPMxCNTH:TPMxCNTL = TPMxMODH:TPMxMODL, the TPM can generate a TOF interrupt at the end of this count. The user can choose to reload any number of the PWM buffers, and they will all update simultaneously at the start of a new period.

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

## B.4 TPM Interrupts

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on the mode of operation for each channel. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register. See the Resets, Interrupts, and System Configuration chapter for absolute interrupt vector addresses, priority, and local interrupt mask control bits.

For each interrupt source in the TPM, a flag bit is set on recognition of the interrupt condition such as timer overflow, channel input capture, or output compare events. This flag may be read (polled) by software to verify that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will be generated whenever the associated interrupt flag equals 1. It is the responsibility of user software to perform a sequence of steps to clear the interrupt flag before returning from the interrupt service routine.

### B.4.1 Clearing Timer Interrupt Flags

TPM interrupt flags are cleared by a 2-step process that includes a read of the flag bit while it is set (1) followed by a write of 0 to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

### B.4.2 Timer Overflow Interrupt Description

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction

at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

### B.4.3 Channel Event Interrupt Description

The meaning of channel interrupts depends on the current mode of the channel (input capture, output compare, edge-aligned PWM, or center-aligned PWM).

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select rising edges, falling edges, any edge, or no edge (off) as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the 2-step sequence described in Section B.4.1, “Clearing Timer Interrupt Flags.”

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the 2-step sequence described in Section B.4.1, “Clearing Timer Interrupt Flags.”

### B.4.4 PWM End-of-Duty-Cycle Events

For channels that are configured for PWM operation, there are two possibilities:

- When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period.
- When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle, which are the times when the timer counter matches the channel value register.

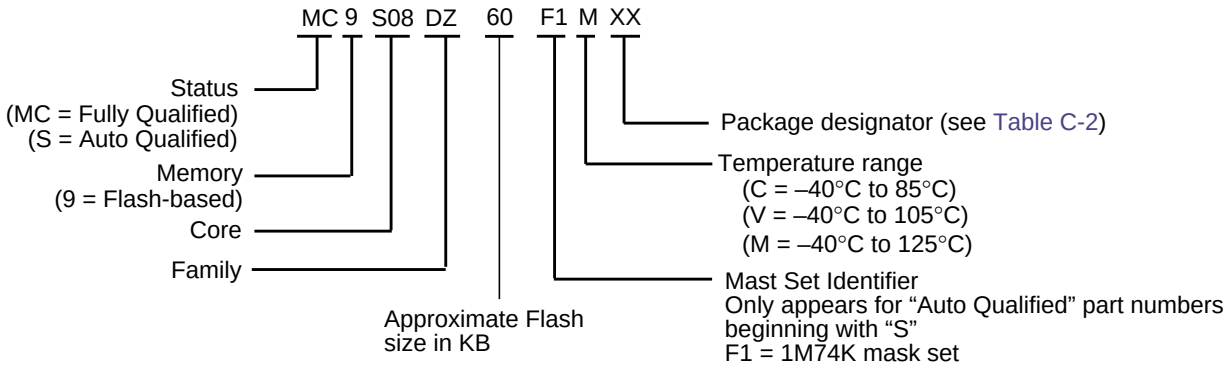
The flag is cleared by the 2-step sequence described in Section B.4.1, “Clearing Timer Interrupt Flags.”

# Appendix C
# Ordering Information and Mechanical Drawings

## C.1 Ordering Information

This section contains ordering information for MC9S08DZ60 Series devices.

Example of the device numbering system:




### C.1.1 MC9S08DZ60 Series Devices

**Table C-1. Devices in the MC9S08DZ60 Series**

| Device Number | Memory | | | Available Packages[1] |
|---|---|---|---|---|
| | FLASH | RAM | EEPROM | |
| MC9S08**DZ60** | 60,032 | 4096 | 2048 | 64-LQFP, 48-LQFP, 32-LQFP |
| MC9S08**DZ48** | 49,152 | 3072 | 1536 | |
| MC9S08**DZ32** | 33,792 | 2048 | 1024 | |
| MC9S08**DZ16** | 16,896 | 1024 | 512 | 48-LQFP, 32-LQFP |

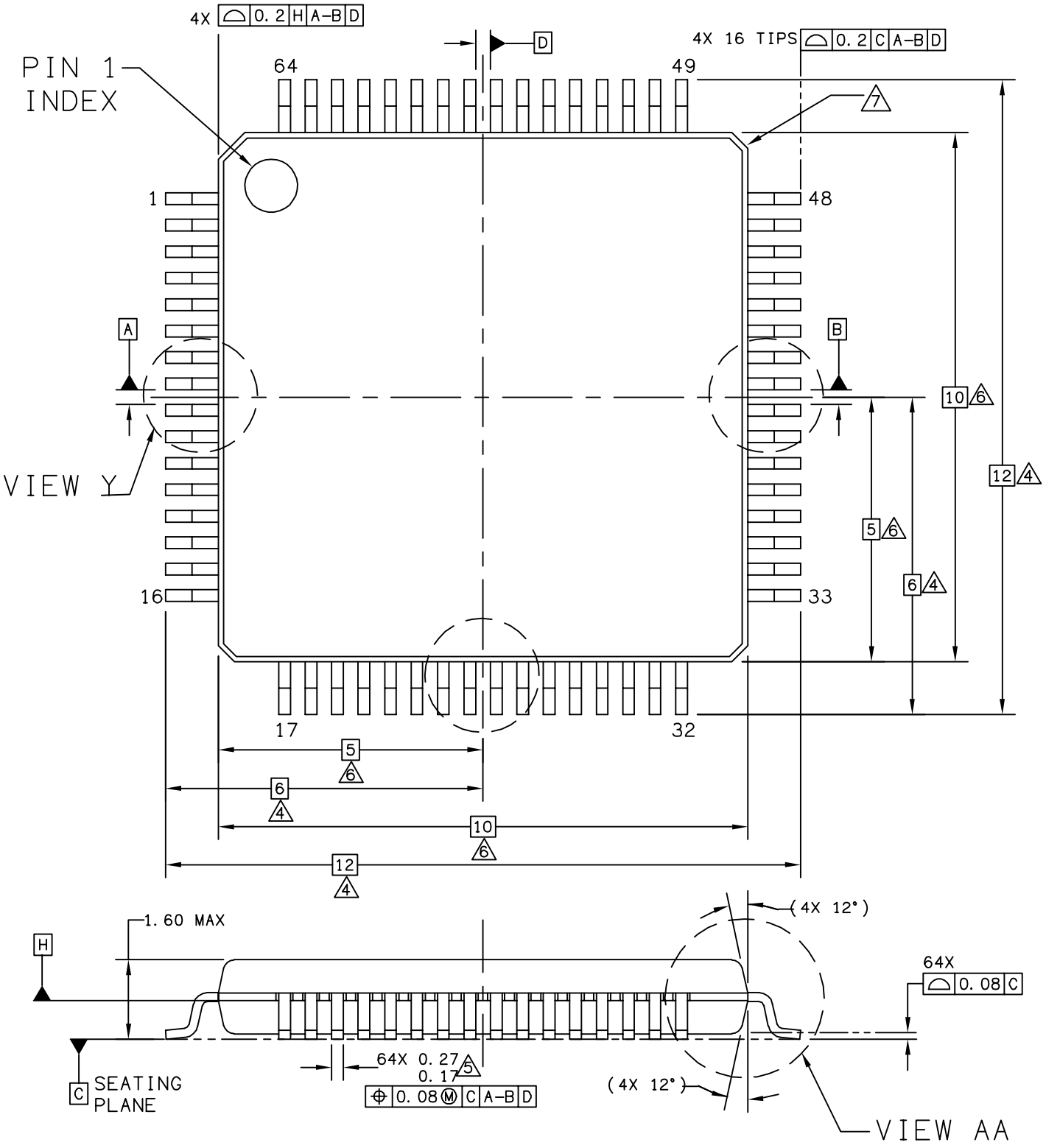
1 See Table C-2 for package information.

## C.2 Mechanical Drawings

The following pages are mechanical drawings for the packages described in the following table:

**Table C-2. Package Descriptions**

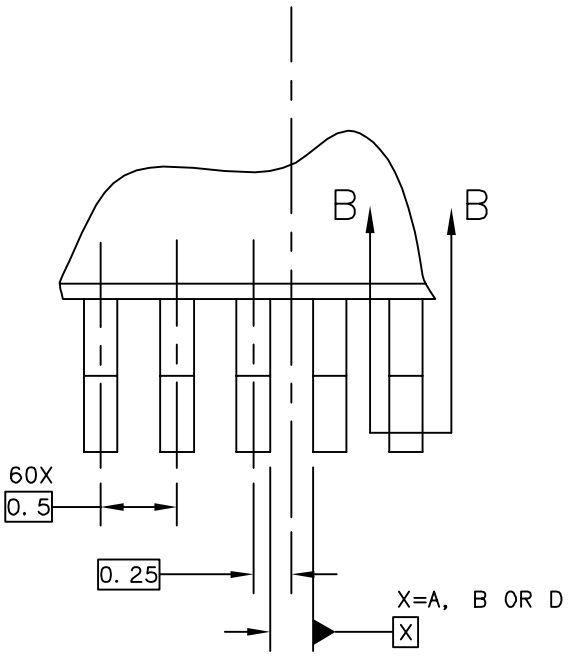
| Pin Count | Type | Abbreviation | Designator | Document No. |
|---|---|---|---|---|
| 64 | Low Quad Flat Package | LQFP | LH | 98ASS23234W |
| 48 | Low Quad Flat Package | LQFP | LF | 98ASH00962A |
| 32 | Low Quad Flat Package | LQFP | LC | 98ASH70029A |



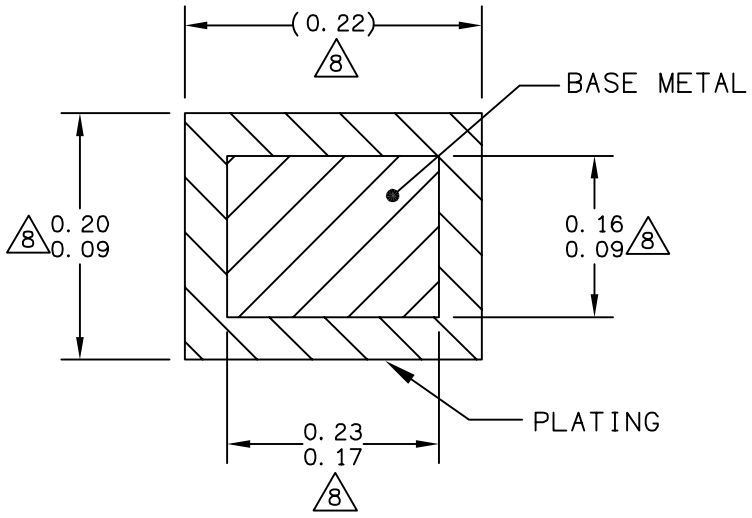

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

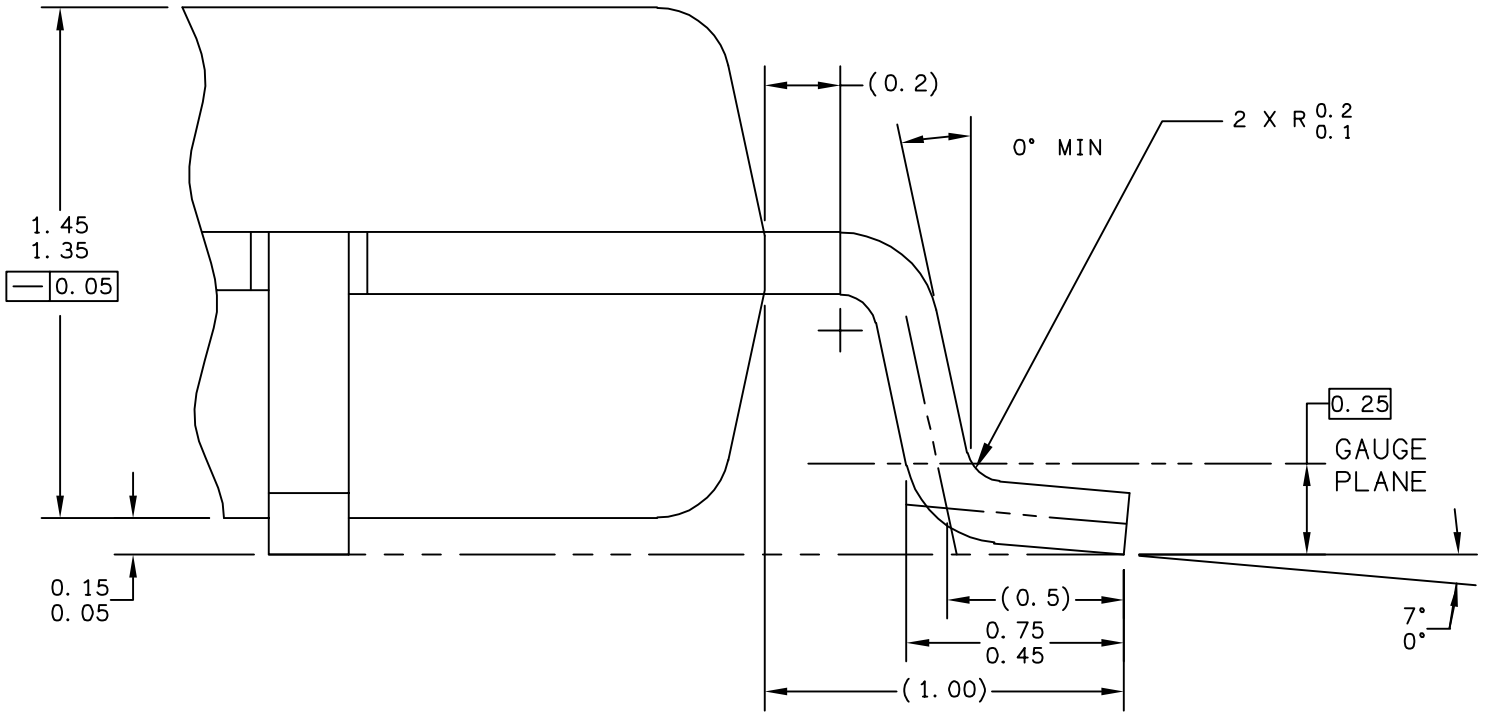
| TITLE: 64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, CASE OUTLINE | DOCUMENT NO: 98ASS23234W | REV: E |
|---|---|---|
| | CASE NUMBER: 840F-02 | 11 AUG 2006 |
| | STANDARD: JEDEC MS-026 BCD | |




VIEW Y




SECTION B–B




VIEW AA

| | | |
|---|---|---|
| © FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED. | **MECHANICAL OUTLINE** | PRINT VERSION NOT TO SCALE |
| TITLE: 64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, CASE OUTLINE | DOCUMENT NO: 98ASS23234W | REV: E |
| | CASE NUMBER: 840F-02 | 11 AUG 2006 |
| | STANDARD: JEDEC MS-026 BCD | |

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994.

3. DATUMS A, B AND D TO BE DETERMINED AT DATUM PLANE H.

4. DIMENSIONS TO BE DETERMINED AT SEATING PLANE C.

5. THIS DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE UPPER LIMIT BY MORE THAN 0.08 mm AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD SHALL NOT BE LESS THAN 0.07 mm.

6. THIS DIMENSION DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. THIS DIMENSION IS MAXIMUM PLASTIC BODY SIZE DIMENSION INCLUDING MOLD MISMATCH.

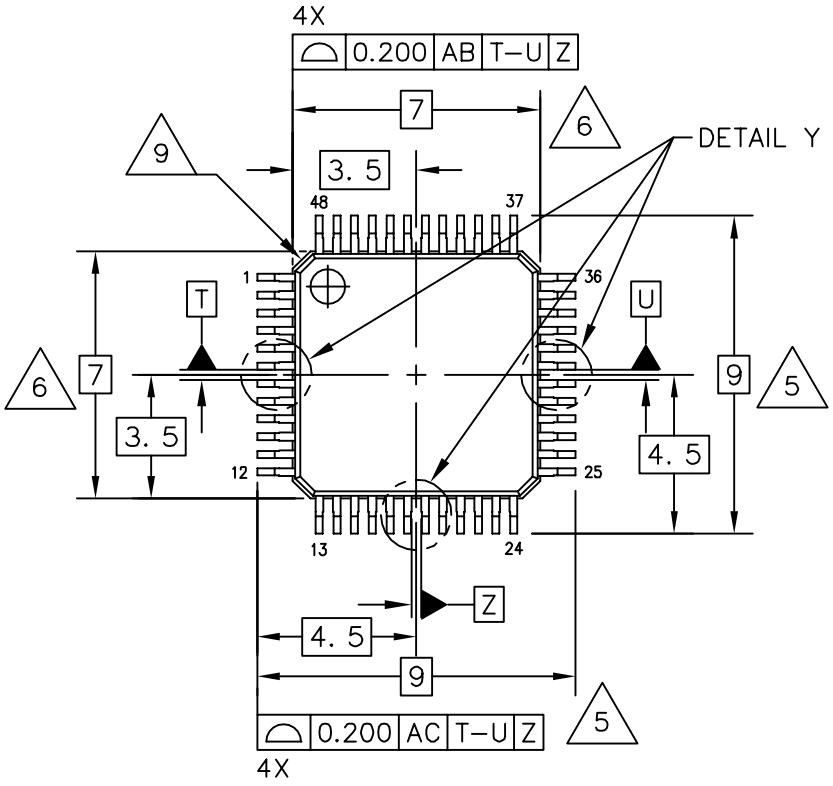
7. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

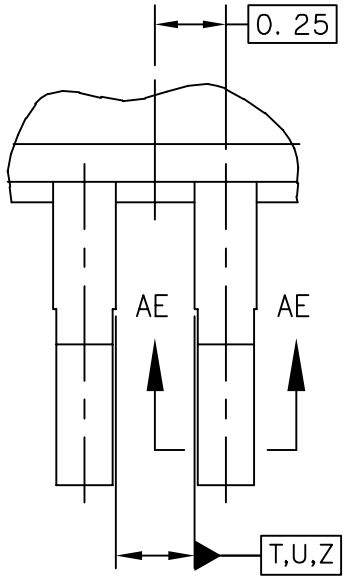
8. THESE DIMENSIONS APPLY TO THE FLAT SECTION OF THE LEAD BETWEEN 0.1 mm AND 0.25 mm FROM THE LEAD TIP.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

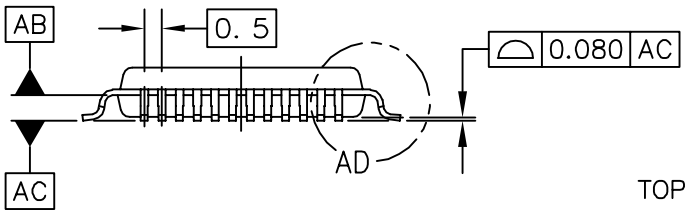
| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

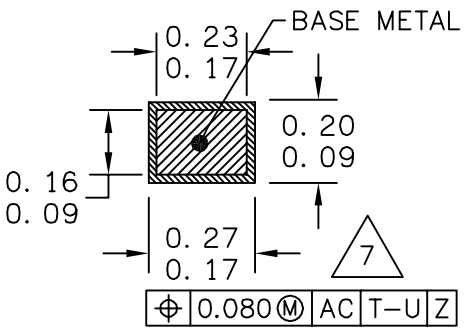
| TITLE: 64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, CASE OUTLINE | DOCUMENT NO: 98ASS23234W | REV: E |
|---|---|---|
| | CASE NUMBER: 840F-02 | 11 AUG 2006 |
| | STANDARD: JEDEC MS-026 BCD | |



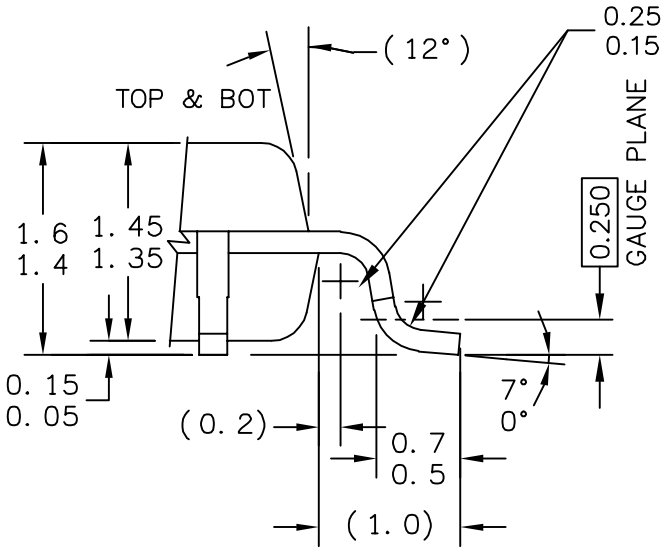




DETAIL Y




SECTION AE–AE




DETAIL AD

| | | |
|---|---|---|
| © FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED. | **MECHANICAL OUTLINE** | PRINT VERSION NOT TO SCALE |
| TITLE: LQFP, 48 LEAD, 0.50 PITCH (7.0 X 7.0 X 1.4) | DOCUMENT NO: 98ASH00962A | REV: G |
| | CASE NUMBER: 932-03 | 14 APR 2005 |
| | STANDARD: JEDEC MS-026-BBC | |

NOTES:

1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M–1994.

2. CONTROLLING DIMENSION: MILLIMETER.

3. DATUM PLANE AB IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.

4. DATUMS T, U, AND Z TO BE DETERMINED AT DATUM PLANE AB.

5. DIMENSIONS TO BE DETERMINED AT SEATING PLANE AC.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.250 PER SIDE. DIMENSIONS DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE AB.

7. THIS DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.350.

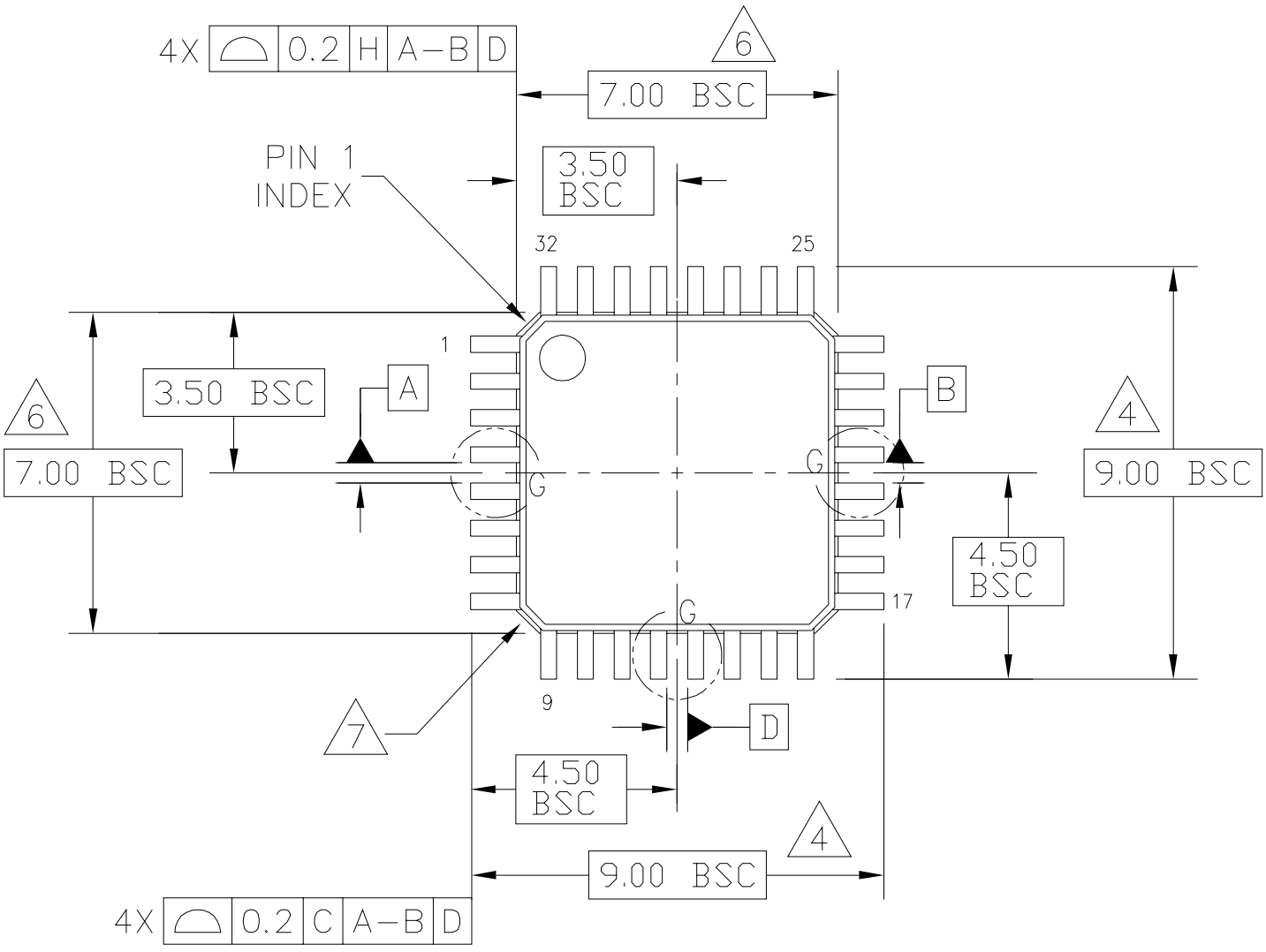
8. MINIMUM SOLDER PLATE THICKNESS SHALL BE 0.0076.

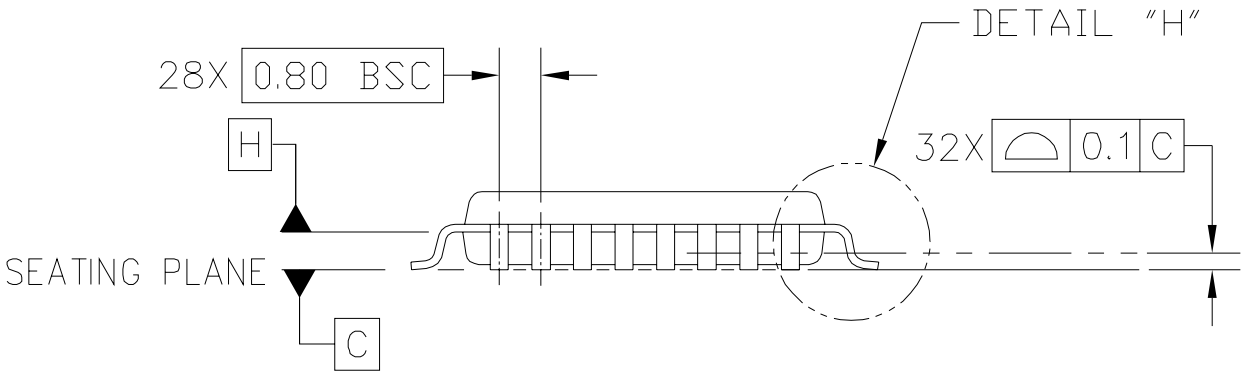
9. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH00962A | REV: G |
|---|---|---|
| LQFP, 48 LEAD, 0.50 PITCH (7.0 X 7.0 X 1.4) | CASE NUMBER: 932-03 | 14 APR 2005 |
| | STANDARD: JEDEC MS-026-BBC | |

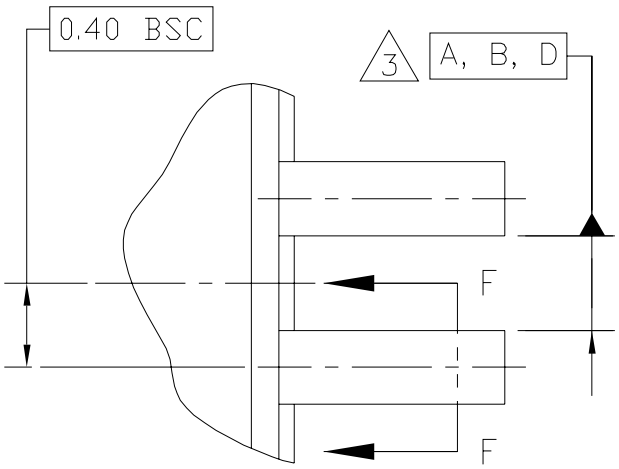


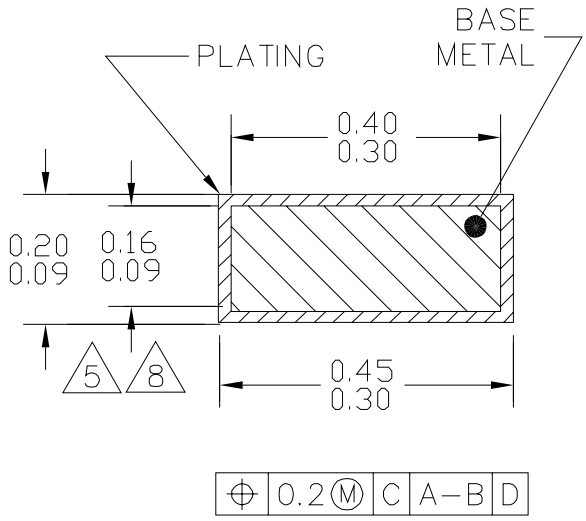

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

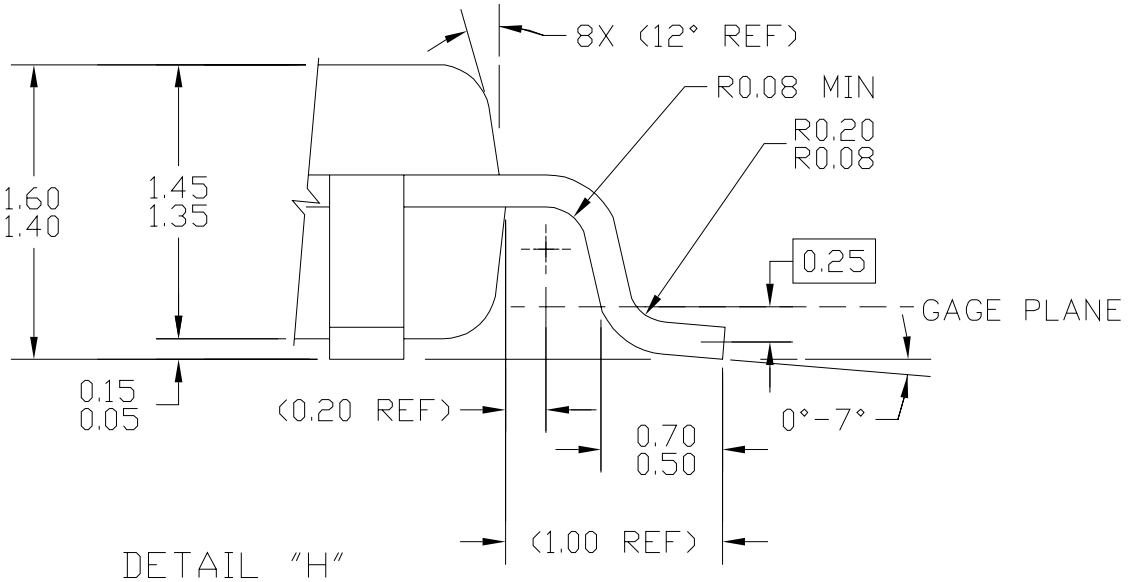
| TITLE: | DOCUMENT NO: 98ASH70029A | REV: D |
|---|---|---|
| LOW PROFILE QUAD FLAT PACK (LQFP)<br>32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | CASE NUMBER: 873A-03 | 19 MAY 2005 |
| | STANDARD: JEDEC MS-026 BBA | |




DETAIL G




SECTION F–F
ROTATED 90°CW
32 PLACES




DETAIL "H"

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH70029A | REV: D |
|---|---|---|
| LOW PROFILE QUAD FLAT PACK (LQFP)<br>32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | CASE NUMBER: 873A-03 | 19 MAY 2005 |
| | STANDARD: JEDEC MS-026 BBA | |

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5–1994.

3. DATUMS A, B, AND D TO BE DETERMINED AT DATUM PLANE H.

4. DIMENSIONS TO BE DETERMINED AT SEATING PLANE DATUM C.

5. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM DIMENSION BY MORE THAN 0.08 MM. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION: 0.07 MM.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE. DIMENSIONS ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.

7. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

8. THESE DIMENSIONS APPLY TO THE FLAT SECTION OF THE LEAD BETWEEN 0.1 MM AND 0.25 MM FROM THE LEAD TIP.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH70029A | REV: D |
|---|---|---|
| LOW PROFILE QUAD FLAT PACK (LQFP)<br>32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | CASE NUMBER: 873A-03 | 19 MAY 2005 |
| | STANDARD: JEDEC MS-026 BBA | |

## *How to Reach Us:*


**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 1-303-675-2140
Fax: 1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com



Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2007-2008. All rights reserved.



MC9S08DZ60
Rev. 4, 6/2008


freescale™
semiconductor

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View S9S08DZ16F1CLC on WIN SOURCE
- NXP / Nexperia Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management