# MC9S08AC16
# MC9S08AC8
# MC9S08AW16A
# MC9S08AW8A

Data Sheet

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

MC9S08AC16
Rev. 9
8/2011

freescale.com

freescale™
semiconductor

# MC9S08AC16 Series Features

### MC9S08AC16 Series Devices

- Consumer & Industrial
  - MC9S08AC16
  - MC9S08AC8
- Automotive
  - MC9S08AW16A
  - MC9S08AW8A

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

- 40-MHz HCS08 CPU (central processor unit)
- 20-MHz internal bus frequency
- HC08 instruction set with added BGND instruction
- Background debugging system
- Breakpoint capability toallow single breakpoint setting during in-circuit debugging (plus two more breakpoints in on-chip debug module)
- Debug module containing two comparators and nine trigger modes. Eight deep FIFO for storing change-of-flow addresses and event-only data. Debug module supports both tag and force breakpoints.
- Support for up to 32 interrupt/reset sources

### Memory Options

- Up to 16 KB of on-chip in-circuit programmable FLASH memory with block protection and security options
- Up to 1 KB of on-chip RAM

### Clock Source Options

- Clock source options include crystal, resonator, external clock, or internally generated clock with precision NVM trimming

### System Protection

- Optional computer operating properly (COP) reset with option to run from independent internal clock source or bus clock
- Low-voltage detection with reset or interrupt
- Illegal opcode detection with reset
- Illegal address detection with reset

### Power-Saving Modes

- Wait plus two stops

### Peripherals

- **ADC** — 8-channel, 10-bit analog-to-digital converter with automatic compare function
- **SCI** — Two serial communications interface modules with optional 13-bit break
- **SPI** — Serial peripheral interface module
- **IIC** — Inter-integrated circuit bus module to operate at up to 100 kbps with maximum bus loading; capable of higher baud rates with reduced loading
- **Timers** — Three 16-bit timer/pulse-width modulator (TPM) modules — Two 2-channel and one 4-channel; each has selectable input capture, output compare, and edge-aligned PWM capability on each channel. Each timer module may be configured for buffered, centered PWM (CPWM) on all channels
- **KBI** — 7-pin keyboard interrupt module

### Input/Output

- Up to 38 general-purpose input/output (I/O) pins
- Software selectable pullups on ports when used as inputs
- Software selectable slew rate control on ports when used as outputs
- Software selectable drive strength on ports when used as outputs
- Master reset pin and power-on reset (POR)
- Internal pullup on $\overline{\text{RESET}}$, IRQ, and BKGD/MS pins to reduce customer system cost

### Package Options

- 48-pin quad flat no-lead package (QFN)
- 44-pin low-profile quad flat package (LQFP)
- 42-pin shrink dual-in-line package (SDIP)
- 32-pin low-profile quad flat package (LQFP)

# MC9S08AC16 Series Data Sheet

Covers MC9S08AC16

MC9S08AC8

MC9S08AW16A

MC9S08AW8A

MC9S08AC16
Rev. 9
8/2011

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. For your convenience, the page number designators have been linked to the appropriate location.

| Revision Number | Revision Date | Description of Changes |
|---|---|---|
| 0 | 12/2007 | Initial Release. |
| 1 | 12/2007 | Updated the package designators for the 32 LQFP and 44 LQFP to be LC and LD respectively. |
| 2 | 2/2008 | Corrected the SPI block module to be V3. |
| 3 | 3/2008 | AC market launch.Verified that the ADC Temp Sensor values were correct. |
| 4 | 5/2008 | Incorporated general release edits and updates, revised the Stop2 and Stop3 max values, added the RoHS logo, and updated the back cover addresses. |
| 5 | 6/2008 | Corrected the note in the TPM introduction. |
| 6 | 7/2008 | Changed all instances of S9S08AWxxA to MC9S08AWxxA except in Appendix B. Added 42SDIP package option. |
| 7 | 5/2009 | Corrected SPI registers in Table 4-2. Added $V_{BG}$ in Table A-6.<br>Corrected title of Table 6-3, Figure 6-13, Figure 6-14, Table 6-5 and Figure 6-19.<br>Added errata for the following sections:<br>• Throughout (remove stop1 instances)<br>• Table 4-1<br>• Table 4-2<br>• Section 9.2, “Keyboard Pin Sharing”<br>• Section 9.3, “Features<br>• Table A-6<br>• Table A-7<br>• Figure A-12 |
| 8 | 11/20/2009 | Updated the whole document for MC9S08AW16A/MC9S08AW8A to support the third TPM module.<br>Updated the TPM 1 channel to 4 for the 32-pin packages in the Table 1-1.<br>Updated the bit 2 of IRQSC register in the Table 4-2.<br>Updated the Temp Sensor Voltage in the Table A-9. |
| 9 | 8/12/2011 | Corrected the address of SPI1D to 0x0055 in the Table 4-2.<br>Updated the $RI_{DD}$ in the Table A-7.<br>Updated the $t_{RTI}$ in the Table A-12 for MC9S08ACxx. |

# List of Chapters

# Contents

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

## Chapter 1
## Introduction



## Chapter 2
## Pins and Connections



## Chapter 3
## Modes of Operation



## Chapter 4
## Memory

## Chapter 5
## Resets, Interrupts, and System Configuration

## Chapter 6
## Parallel Input/Output

## Chapter 7
## Central Processor Unit (S08CPUV2)



## Chapter 8
## Internal Clock Generator (S08ICGV4)

## Chapter 9
## Keyboard Interrupt (S08KBIV1)



## Chapter 10
## Timer/PWM (S08TPMV3)

## Chapter 11
## Serial Communications Interface (S08SCIV4)

## Chapter 12
## Serial Peripheral Interface (S08SPIV3)



## Chapter 13
## Inter-Integrated Circuit (S08IICV2)

## Chapter 14
## Analog-to-Digital Converter (S08ADC10V1)

## Chapter 15
## Development Support



## Appendix A
## Electrical Characteristics and Timing Specifications

## Appendix B
## Ordering Information and Mechanical Drawings

# Chapter 1
# Introduction

## 1.1 Overview

The MC9S08AC16 Series devices are members of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types. Refer to Table 1-1 for memory sizes and package types.

**NOTE**

- The **MC9S08AC16** and **MC9S08AC8** devices are qualified for, and are intended to be used in, *consumer and industrial* applications.
- The **MC9S08AW16A** and **MC9S08AW8A** devices are qualified for, and are intended to be used in, *automotive* applications.

Table 1-1 summarizes the feature set available in the MCUs.

**Table 1-1. Features by MCU and Package**

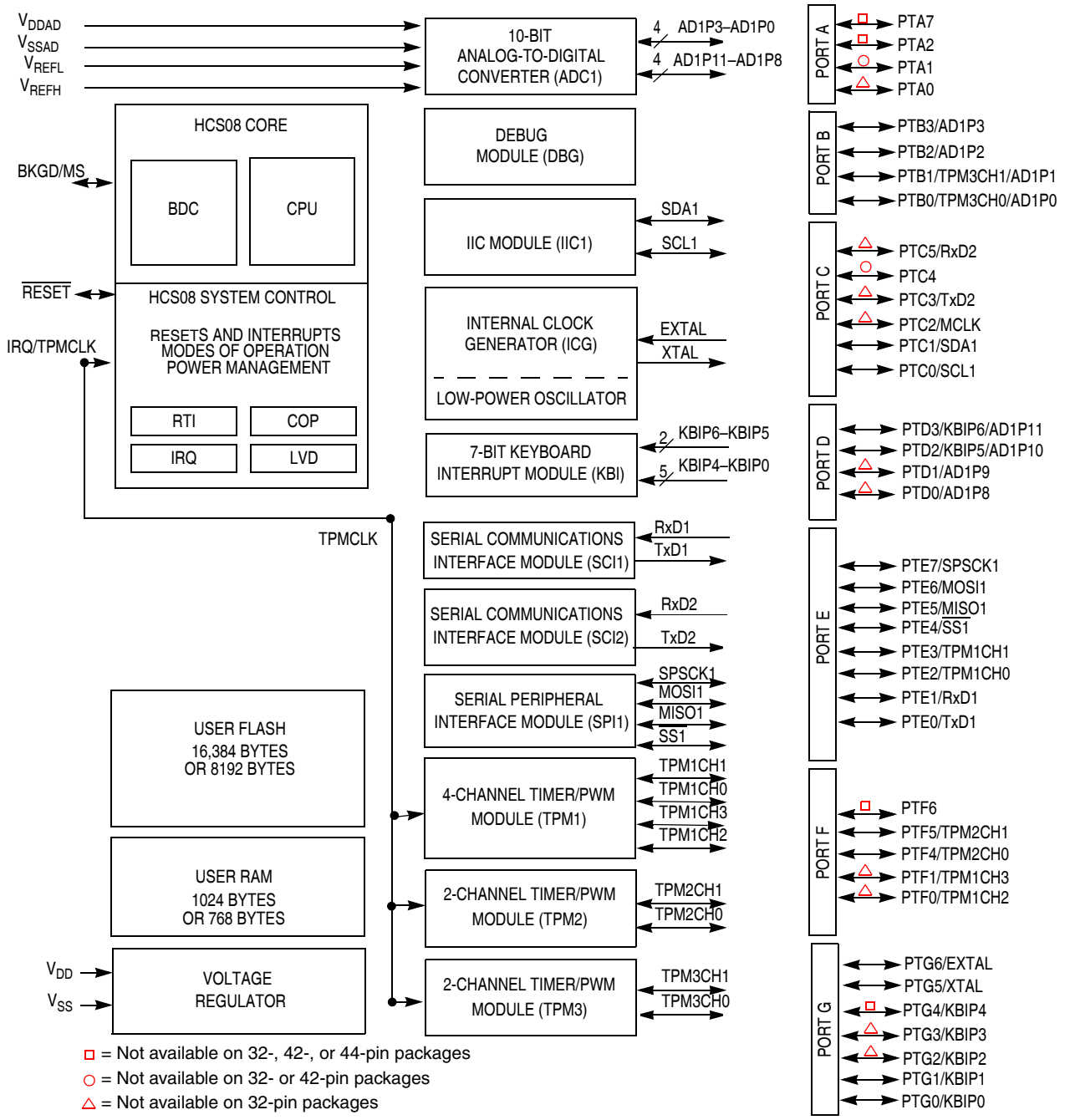
| Consumer and Industrial "AC" Devices | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Feature** | **MC9S08AC16** | | | | **MC9S08AC8** | | | |
| FLASH size (bytes) | 16K | | | | 8K | | | |
| RAM size (bytes) | 1024 | | | | 768 | | | |
| Pin quantity | **48** | **44** | **42** | **32** | **48** | **44** | **42** | **32** |
| ADC channels | 8 | 8 | 8 | 6 | 8 | 8 | 8 | 6 |
| TPM1 channels[1] | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| TPM2 channels | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| TPM3 channels | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| KBI pins | 7 | 6 | 6 | 4 | 7 | 6 | 6 | 4 |
| GPIO pins | 38 | 34 | 32 | 22 | 38 | 34 | 32 | 22 |
| Consumer & Industrial Qualified | yes | | | | yes | | | |
| Automotive Qualified | no | | | | no | | | |

| Automotive "AW" Devices | | | | | | |
|---|---|---|---|---|---|---|
| **Feature** | **MC9S08AW16A** | | | **MC9S08AW8A** | | |
| FLASH size (bytes) | 16K | | | 8K | | |
| RAM size (bytes) | 1024 | | | 768 | | |
| Pin quantity | **48** | **44** | **32** | **48** | **44** | **32** |
| ADC channels | 8 | 8 | 6 | 8 | 8 | 6 |
| TPM1 channels[1] | 4 | 4 | 4 | 4 | 4 | 4 |
| TPM2 channels | 2 | 2 | 2 | 2 | 2 | 2 |
| TPM3 channels | 2 | 2 | 2 | 2 | 2 | 2 |
| KBI pins | 7 | 6 | 4 | 7 | 6 | 4 |
| GPIO pins | 38 | 34 | 22 | 38 | 34 | 22 |
| Consumer & Industrial Qualified | no | | | no | | |
| Automotive Qualified | yes | | | yes | | |

[1] There are 4 channels on TPM1 but two of them (TPM1CH2 and TPM1CH3) are not bonded to 32-pin LQFP package. These two channels can be used for soft timer function.

## 1.2 MCU Block Diagrams

The block diagram shows the structure of the MC9S08AC16 Series MCU.




**Notes**:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to $V_{DD}$. IRQ should not be driven above $V_{DD}$.
4. Pin contains integrated pullup device.
5. PTD3, PTD2, and PTG4 contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 1-1. MC9S08AC16 Block Diagram**

Table 1-2 lists the functional versions of the on-chip modules.

**Table 1-2. Versions of On-Chip Modules**

| Module | | Version |
|---|---|---|
| Analog-to-Digital Converter | (ADC) | 1 |
| Internal Clock Generator | (ICG) | 4 |
| Inter-Integrated Circuit | (IIC) | 2 |
| Keyboard Interrupt | (KBI) | 1 |
| Serial Communications Interface | (SCI) | 4 |
| Serial Peripheral Interface | (SPI) | 3 |
| Timer Pulse-Width Modulator | (TPM) | 3 |
| Central Processing Unit | (CPU) | 2 |

## 1.3 System Clock Distribution

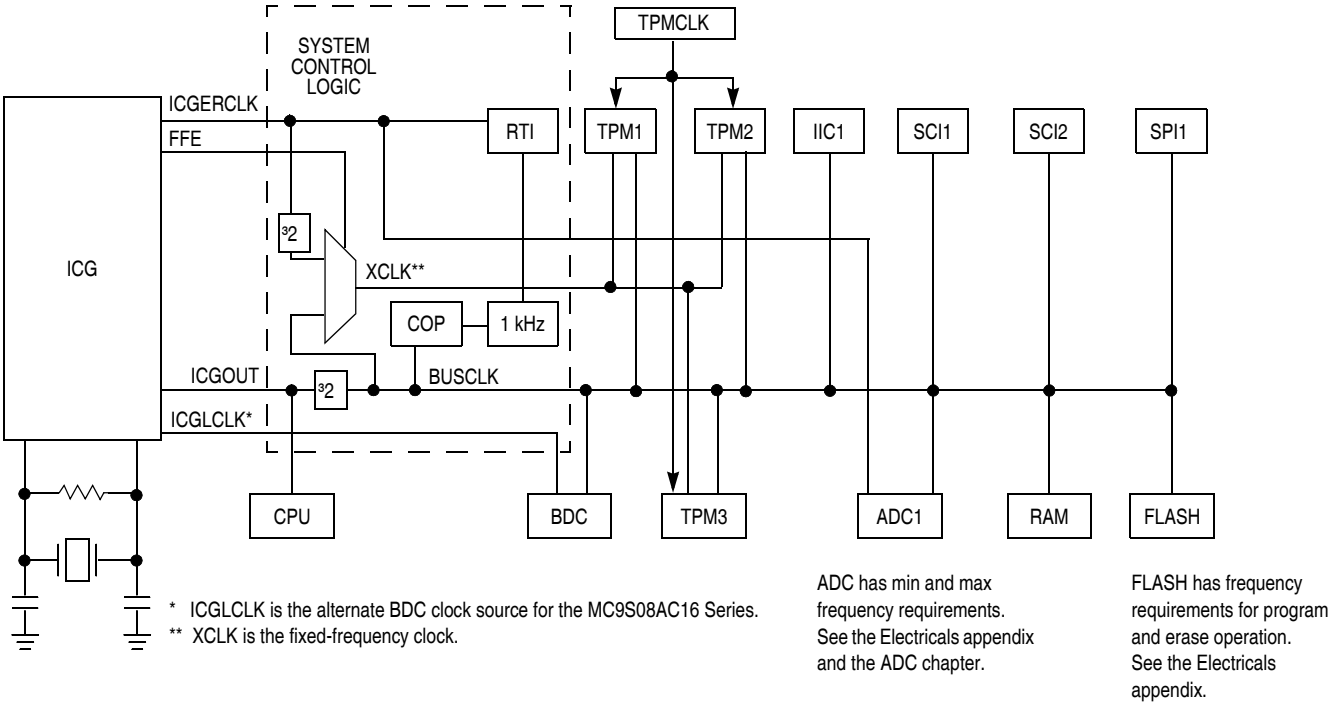



**Figure 1-2. System Clock Distribution Diagram**

Some of the modules inside the MCU have clock source choices. Figure 1-2 shows a simplified clock connection diagram. The ICG supplies the clock sources:

- ICGOUT is an output of the ICG module. It is one of the following:
  - The external crystal oscillator
  - An external clock source
  - The output of the digitally-controlled oscillator (DCO) in the frequency-locked loop sub-module

— Control bits inside the ICG determine which source is connected.

- FFE is a control signal generated inside the ICG. If the frequency of ICGOUT > 4 × the frequency of ICGERCLK, this signal is a logic 1 and the fixed-frequency clock will be ICGERCLK/2. Otherwise the fixed-frequency clock will be BUSCLK.
- ICGLCLK — Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICGERCLK — External reference clock can be selected as the real-time interrupt clock source. Can also be used as the ALTCLK input to the ADC module.

# Chapter 2
# Pins and Connections

## 2.1 Introduction

This chapter describes signals that connect to package pins. It includes a pinout diagram, a table of signal properties, and detailed discussion of signals.

## 2.2 Device Pin Assignment

Figure 2-1 shows the 48-pin QFN pin assignments for the MC9S08AC16 Series device.

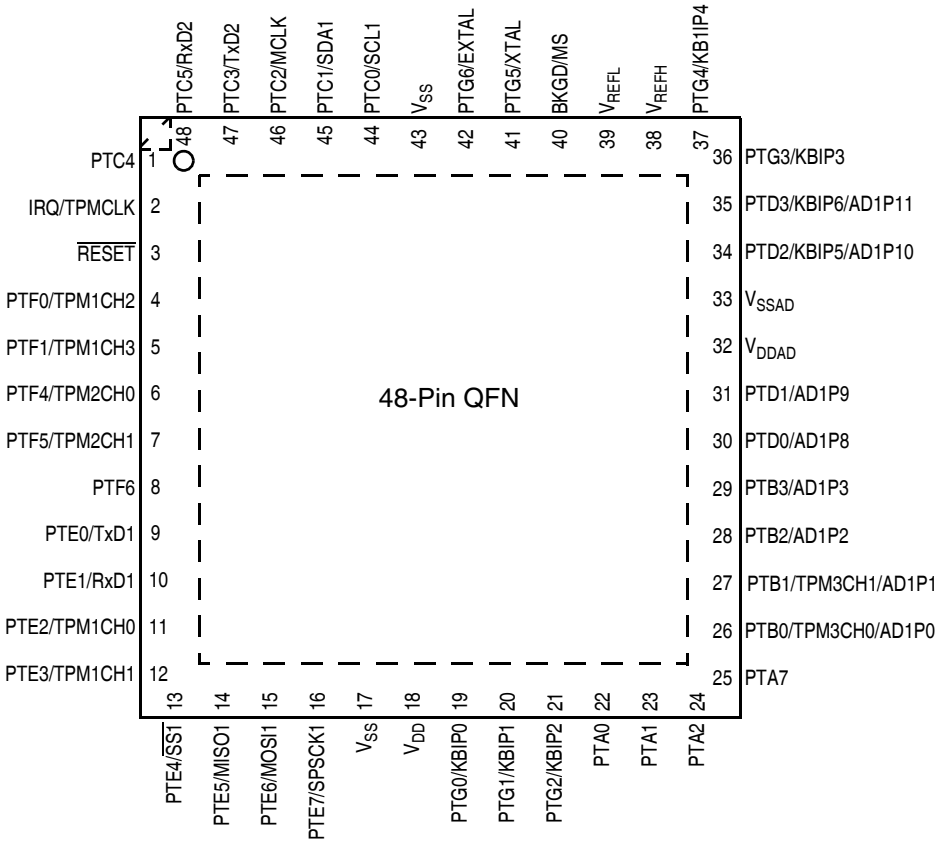



**Figure 2-1. MC9S08AC16 Series in 48-Pin QFN Package**

Figure 2-2. shows the 44-pin LQFP pin assignments for the MC9S08AC16 Series device.

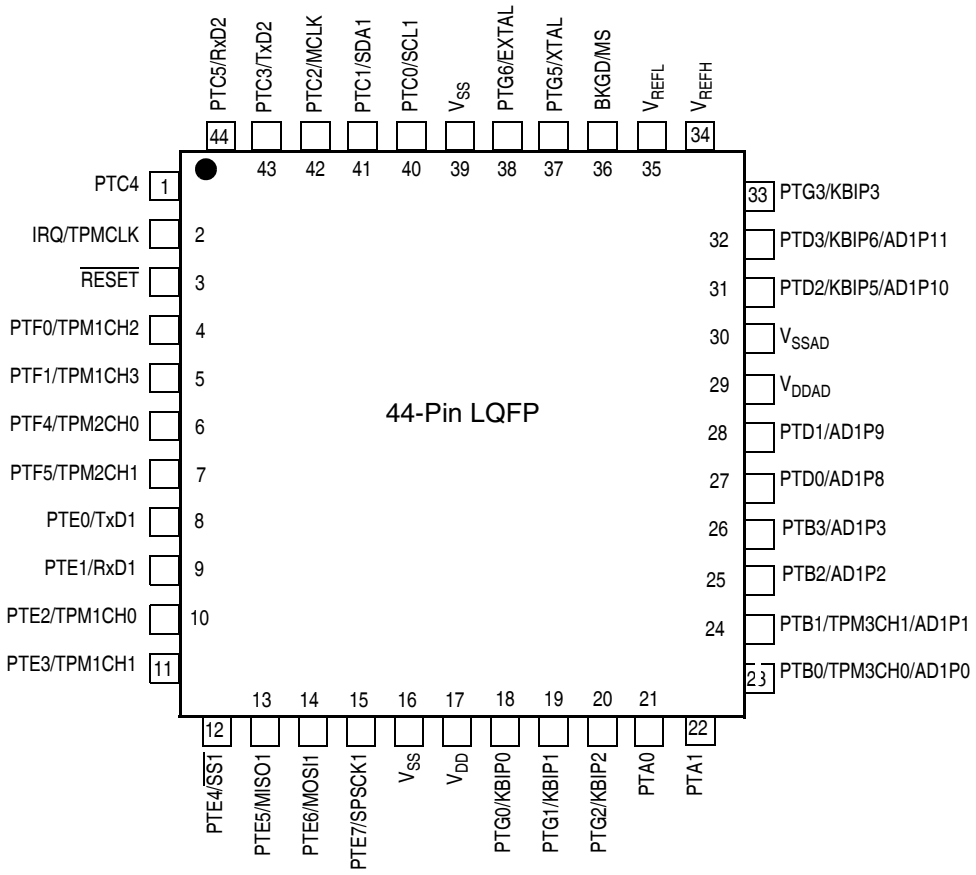



**Figure 2-2. MC9S08AC16 Series in 44-Pin LQFP Package**

Figure 2-3 shows the 42-pin SDIP pin assignments for the MC9S08AC16 Series device.

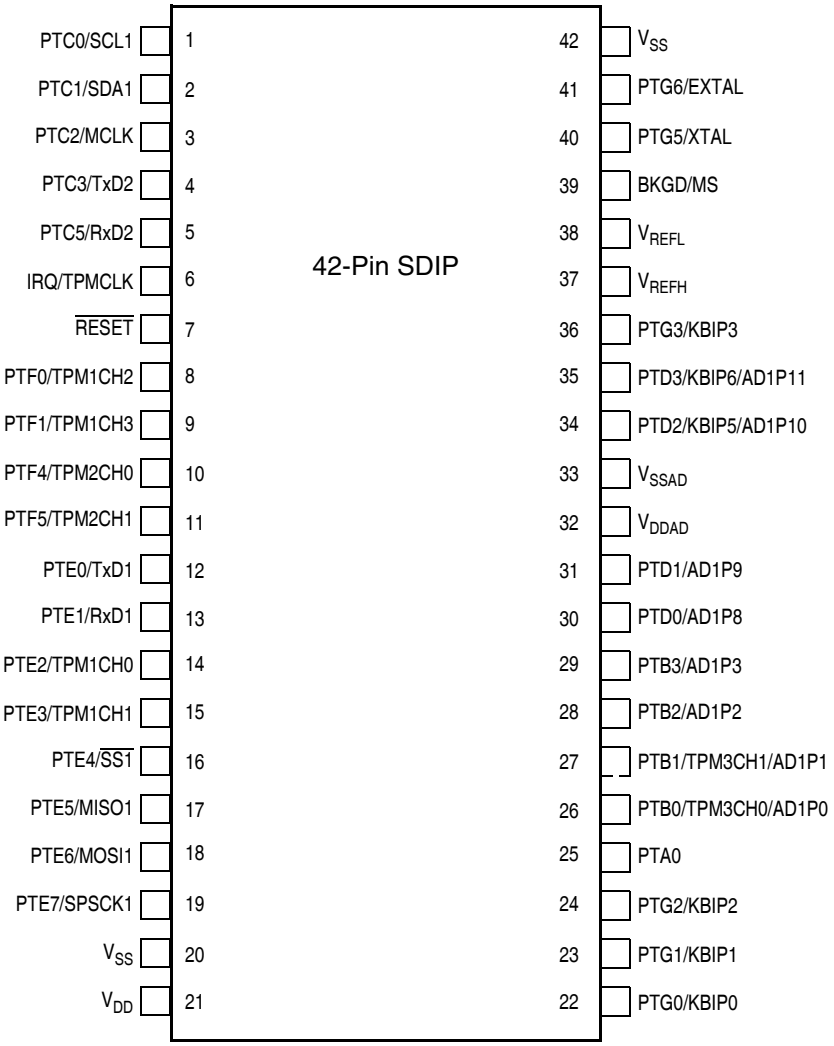



**Figure 2-3. MC9S08AC16 Series in 42-Pin SDIP Package**

Figure 2-4 shows the 32-pin LQFP pin assignments for the MC9S08AC16 Series device.

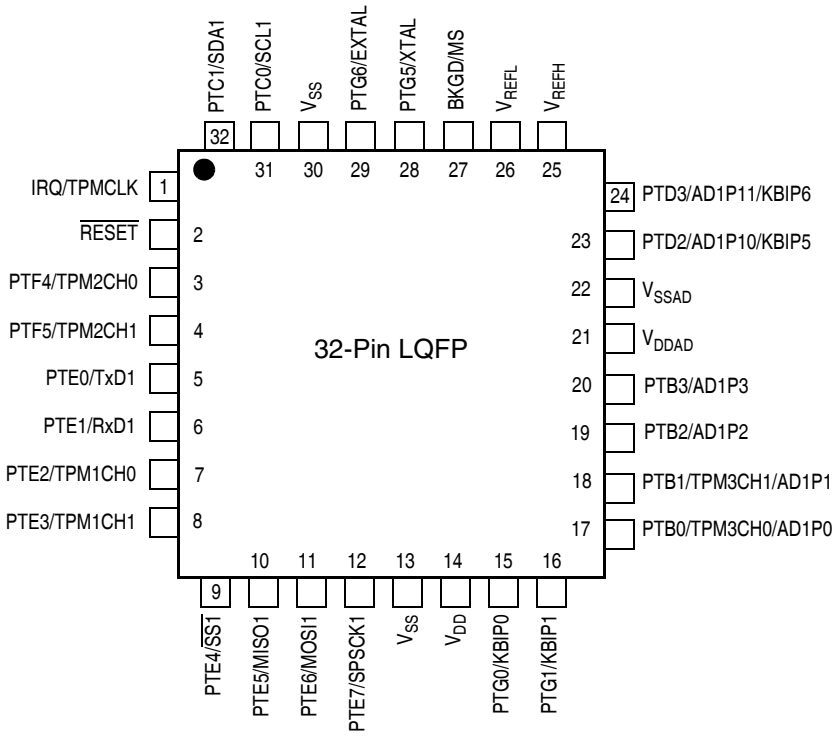



**Figure 2-4. MC9S08AC16 Series in 32-Pin LQFP Package**

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

| Pin Number | | | | <-- Lowest Priority --> Highest | | |
|---|---|---|---|---|---|---|
| 48 | 44 | 42 | 32 | Port Pin | Alt 1 | Alt 2 |
| 1 | 1 | — | — | PTC4 | | |
| 2 | 2 | 6 | 1 | | IRQ | TPMCLK |
| 3 | 3 | 7 | 2 | | | $\overline{\text{RESET}}$ |
| 4 | 4 | 8 | — | PTF0 | TPM1CH2 | |
| 5 | 5 | 9 | — | PTF1 | TPM1CH3 | |
| 6 | 6 | 10 | 3 | PTF4 | TPM2CH0 | |
| 7 | 7 | 11 | 4 | PTF5 | TPM2CH1 | |
| 8 | — | — | — | PTF6 | | |
| 9 | 8 | 12 | 5 | PTE0 | | TxD1 |
| 10 | 9 | 13 | 6 | PTE1 | | RxD1 |
| 11 | 10 | 14 | 7 | PTE2 | TPM1CH0 | |
| 12 | 11 | 15 | 8 | PTE3 | TPM1CH1 | |
| 13 | 12 | 16 | 9 | PTE4 | | $\overline{\text{SS1}}$ |
| 14 | 13 | 17 | 10 | PTE5 | | MISO1 |
| 15 | 14 | 18 | 11 | PTE6 | | MOSI1 |
| 16 | 15 | 19 | 12 | PTE7 | | SPSCK1 |
| 17 | 16 | 20 | 13 | | | $V_{SS}$ |
| 18 | 17 | 21 | 14 | | | $V_{DD}$ |
| 19 | 18 | 22 | 15 | PTG0 | | KBIP0 |
| 20 | 19 | 23 | 16 | PTG1 | | KBIP1 |
| 21 | 20 | 24 | — | PTG2 | | KBIP2 |
| 22 | 21 | 25 | — | PTA0 | | |
| 23 | 22 | — | — | PTA1 | | |
| 24 | — | — | — | PTA2 | | |

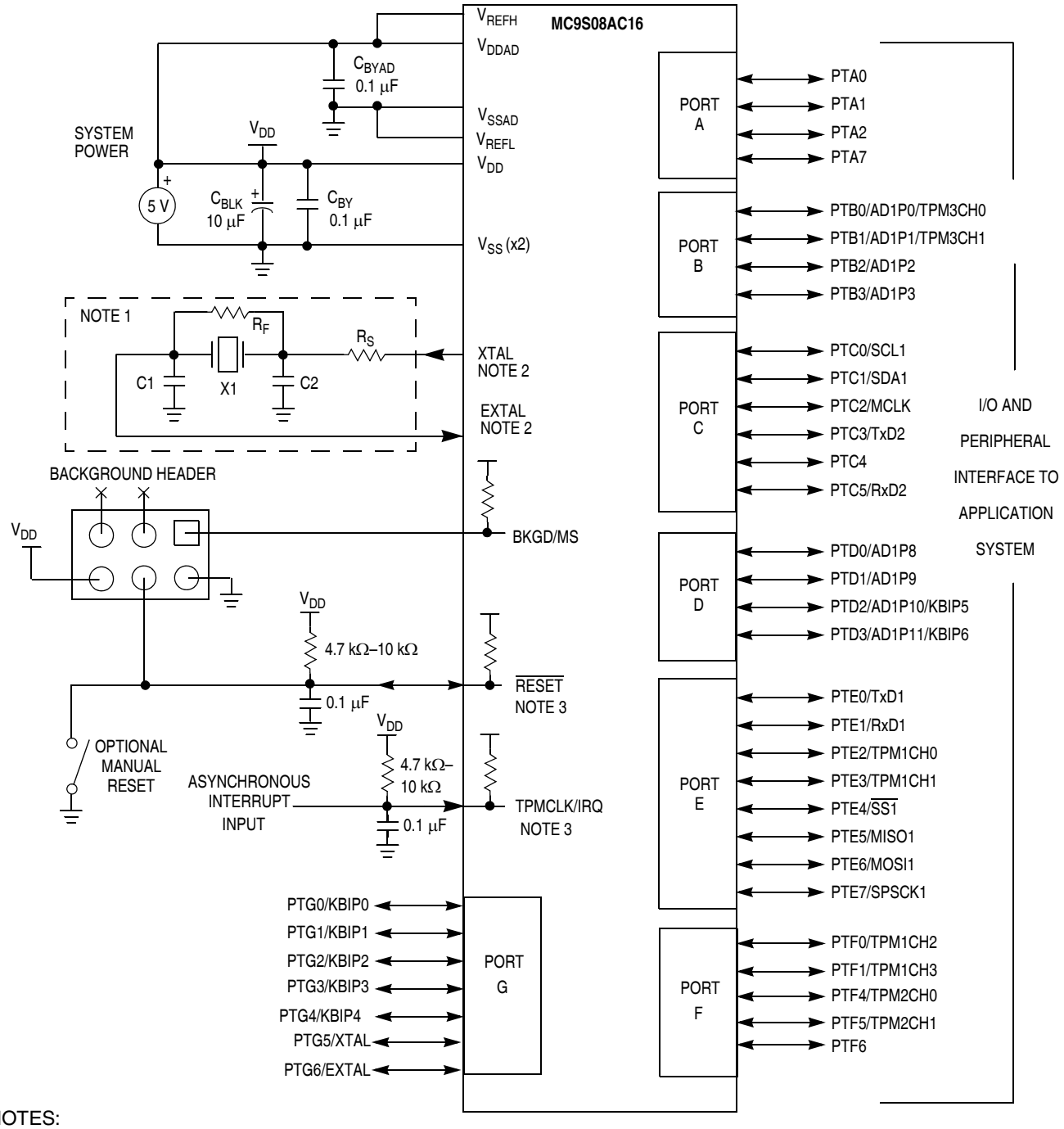
| Pin Number | | | | <-- Lowest Priority --> Highest | | |
|---|---|---|---|---|---|---|
| 48 | 44 | 42 | 32 | Port Pin | Alt 1 | Alt 2 |
| 25 | — | — | — | PTA7 | | |
| 26 | 23 | 26 | 17 | PTB0 | TPM3CH0 | AD1P0 |
| 27 | 24 | 27 | 18 | PTB1 | TPM3CH1 | AD1P1 |
| 28 | 25 | 28 | 19 | PTB2 | AD1P2 | |
| 29 | 26 | 29 | 20 | PTB3 | AD1P3 | |
| 30 | 27 | 30 | — | PTD0 | AD1P8 | |
| 31 | 28 | 31 | — | PTD1 | AD1P9 | |
| 32 | 29 | 32 | 21 | | | $V_{DDAD}$ |
| 33 | 30 | 33 | 22 | | | $V_{SSAD}$ |
| 34 | 31 | 34 | 23 | PTD2 | AD1P10 | KBIP5 |
| 35 | 32 | 35 | 24 | PTD3 | AD1P11 | KBIP6 |
| 36 | 33 | 36 | — | PTG3 | | KBIP3 |
| 37 | — | — | — | PTG4 | KBIP4 | |
| 38 | 34 | 37 | 25 | | | $V_{REFH}$ |
| 39 | 35 | 38 | 26 | | | $V_{REFL}$ |
| 40 | 36 | 39 | 27 | BKGD | MS | |
| 41 | 37 | 40 | 28 | PTG5 | XTAL | |
| 42 | 38 | 41 | 29 | PTG6 | EXTAL | |
| 43 | 39 | 42 | 30 | | | $V_{SS}$ |
| 44 | 40 | 1 | 31 | PTC0 | | SCL1 |
| 45 | 41 | 2 | 32 | PTC1 | | SDA1 |
| 46 | 42 | 3 | — | PTC2 | | MCLK |
| 47 | 43 | 4 | — | PTC3 | | TxD2 |
| 48 | 44 | 5 | — | PTC5 | | RxD2 |

**Table 2-2. Pin Function Reference**

| Signal Function | Example(s) | Reference |
|---|---|---|
| Port Pins | PTAx, PTBx | Chapter 6, “Parallel Input/Output” |
| Serial peripheral interface | $\overline{\text{SS}}$, MISO, MOSI, SPSCK | Chapter 12, “Serial Peripheral Interface (S08SPIV3)” |
| Keyboard interrupts | KBIPx | Chapter 9, “Keyboard Interrupt (S08KBIV1)” |
| Timer/PWM | TCLK, TPMCHx | Chapter 10, “Timer/PWM (S08TPMV3)” |
| Inter-integrated circuit | SCL, SDA | Chapter 13, “Inter-Integrated Circuit (S08IICV2)” |
| Serial communications interface | TxD, RxD | Chapter 11, “Serial Communications Interface (S08SCIV4)” |
| Oscillator/clocking | EXTAL, XTAL | Chapter 8, “Internal Clock Generator (S08ICGV4)” |
| Analog-to-digital | ADPx | Chapter 14, “Analog-to-Digital Converter (S08ADC10V1)” |
| Power/core | BKGD/MS, $V_{DD}$, $V_{SS}$ | Chapter 2, “Pins and Connections” |
| Reset and interrupts | $\overline{\text{RESET}}$, IRQ | Chapter 5, “Resets, Interrupts, and System Configuration” |

## 2.3 Recommended System Connections

Figure 2-5 shows pin connections that are common to almost all MC9S08AC16 Series application systems.




NOTES:
1. Not required if using the internal clock option.
2. XTAL and EXTAL are PTG5 and PTG6 respectively.
3. RC filters on $\overline{\text{RESET}}$ and IRQ are recommended for EMC-sensitive applications.

**Figure 2-5. Basic System Connections**

### 2.3.1 Power ($V_{DD}$, 2 x $V_{SS}$, $V_{DDAD}$, $V_{SSAD}$)

$V_{DD}$ 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 paired $V_{DD}$ and $V_{SS}$ power pins as practical to suppress high-frequency noise. The MC9S08AC16 has a second $V_{SS}$ pin. This pin should be connected to the system ground plane or to the primary $V_{SS}$ pin through a low-impedance connection.

$V_{DDAD}$ and $V_{SSAD}$ 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 analog power pins as practical to suppress high-frequency noise.

### 2.3.2 Oscillator (XTAL, EXTAL)

Out of reset the MCU uses an internally generated clock (self-clocked mode — $f_{Self\_reset}$) equivalent to about 8-MHz crystal rate. This frequency source is used during reset startup and can be enabled as the clock source for stop recovery to avoid the need for a long crystal startup delay. This MCU also contains a trimmable internal clock generator (ICG) module that can be used to run the MCU. For more information on the ICG, see the Chapter 8, “Internal Clock Generator (S08ICGV4).”

The oscillator in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator in either of two frequency ranges selected by the RANGE bit in the ICGC1 register. Rather than a crystal or ceramic resonator, an external oscillator can be connected to the EXTAL input pin.

Refer to Figure 2-5 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 and 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 sizing 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).

### 2.3.3 $\overline{\text{RESET}}$

$\overline{\text{RESET}}$ is a dedicated pin with a pullup 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 reset pin is driven low for approximately 34 bus cycles. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the system control reset status register (SRS).

In EMC-sensitive applications, an external RC filter is recommended on the reset pin. See Figure 2-5 for an example.

### 2.3.4 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/mode select pin, the pin includes an internal pullup 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/MS low during the rising edge of reset which forces the MCU to active background mode.

The BKGD 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 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 pullup device play almost no role in determining rise and fall times on the BKGD pin.

### 2.3.5 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.

### 2.3.6 External Interrupt Pin (IRQ)

The IRQ pin is the input source for the IRQ interrupt and is also the input for the BIH and BIL instructions. If the IRQ function is not enabled, this pin does not perform any function.

In EMC-sensitive applications, an external RC filter is recommended on the IRQ pin. See Figure 2-5 for an example.

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

The remaining pins are shared among general-purpose I/O and on-chip peripheral functions such as timers and serial I/O systems. Immediately after reset, all of these pins are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

**NOTE**

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

For information about controlling these pins as general-purpose I/O pins, see Chapter 6, “Parallel Input/Output.” For information about how and when on-chip peripheral systems use these pins, refer to the appropriate chapter from Table 2-2.

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. See the Chapter 6, “Parallel Input/Output” chapter for more details.

Pullup enable bits for each input pin control whether on-chip pullup devices are enabled whenever the pin is acting as an input even if it is being controlled by an on-chip peripheral module. When the PTD3, PTD2, and PTG4 pins are controlled by the KBI module and are configured for rising-edge/high-level sensitivity, the pullup enable control bits enable pulldown devices rather than pullup devices. Similarly, when IRQ is configured as the IRQ input and is set to detect rising edges, the pullup enable control bit enables a pulldown device rather than a pullup device.

**NOTE**

When an alternative function is first enabled it is possible to get a spurious edge to the module, user software should clear out any associated flags before interrupts are enabled. Table 2-1 illustrates the priority if multiple modules are enabled. The highest priority module will have control over the pin. Selecting a higher priority pin function with a lower priority function already enabled can cause spurious edges to the lower priority module. It is recommended that all modules that share a pin be disabled before enabling another module.

# Chapter 3
# Modes of Operation

## 3.1 Introduction

The operating modes of the MC9S08AC16 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 modefor code development
- Wait mode:
  - CPU shuts down to conserve power
  - System clocks running
  - Full voltage regulation maintained
- Stop modes:
  - System clocks stopped; voltage regulator in standby
  - Stop2 — Partial power down of internal circuits, RAM contents retained
  - Stop3 — All internal circuits powered for fast recovery

## 3.3 Run Mode

This is the normal operating mode for the MC9S08AC16 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 lowat the rising edge of reset
- When a BACKGROUND command isreceived through the BKGD 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's 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 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's 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 MC9S08AC16 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 Chapter 15, "Development Support."

## 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 the system option register is set. In both stop modes, all internal clocks are halted. If the STOPE bit is not set when

the CPU executes a STOP instruction, the MCU will not enter either of the stop modes and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in SPMSC2.

HCS08 devices that are designed for low voltage operation (1.8V to 3.6V) also include stop1 mode. The MC9S08AC16 Series family of devices does not include stop1 mode.

Table 3-1 summarizes the behavior of the MCU in each of the stop modes.

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

| Mode | PPDC | CPU, Digital Peripherals, FLASH | RAM | ICG | ADC | Regulator | I/O Pins | RTI |
|---|---|---|---|---|---|---|---|---|
| Stop2 | 1 | Off | Standby | Off | Disabled | Standby | States held | Optionally on |
| Stop3 | 0 | Standby | Standby | Off[1] | Optionally on | Standby | States held | Optionally on |

[1] Crystal oscillator can be configured to run in stop3. Please see the ICG registers.

## 3.6.1 Stop2 Mode

The stop2 mode provides very low standby power consumption and maintains the contents of RAM and the current state of all of the I/O pins. To enter stop2, the user must execute a STOP instruction with stop2 selected (PPDC = 1) and stop mode enabled (STOPE = 1). In addition, the LVD must not be enabled to operate in stop (LVDSE = LVDE = 1). If the LVD is enabled in stop, then the MCU enters stop3 upon the execution of the STOP instruction regardless of the state of PPDC.

Before entering stop2 mode, the user must save the contents of the I/O port registers, as well as any other memory-mapped registers which they want to restore after exit of stop2, to locations in RAM. Upon exit of stop2, these values can be restored by user software before pin latches are opened.

When the MCU is in stop2 mode, all internal circuits that are powered from the voltage regulator are turned off, except for the RAM. The voltage regulator is in a low-power standby state, as is the ADC. Upon entry into stop2, the states of the I/O pins are latched. The states are held while in stop2 mode and after exiting stop2 mode until a logic 1 is written to PPDACK in SPMSC2.

Exit from stop2 is done by asserting either of the wake-up pins: $\overline{\text{RESET}}$ or IRQ/TPMCLK, or by an RTI interrupt. IRQ/TPMCLK is always an active low input when the MCU is in stop2, regardless of how it was configured before entering stop2.

Upon wake-up from stop2 mode, the MCU will start up as from a power-on reset (POR) except pin states remain latched. The CPU will take the reset vector. The system and all peripherals will be in their default reset states and must be initialized.

After waking up from stop2, the PPDF bit in SPMSC2 is set. This flag may be 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 logic 1 is written to PPDACK in SPMSC2.

To maintain I/O state for pins that were configured as general-purpose I/O, 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

register bits will assume their reset states when the I/O pin latches are opened and the I/O pins will switch to their reset states.

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.

### 3.6.2 Stop3 Mode

To enter stop3, the user must execute a STOP instruction with stop3 selected (PPDC = 0) and stop mode enabled (STOPE = 1). Upon entering the stop3 mode, all of the clocks in the MCU, including the oscillator itself, are halted. The ICG enters its standby state, as does the voltage regulator and the ADC. The states of all of the internal registers and logic, as well as the RAM content, are maintained. The I/O pin states are not latched at the pin as in stop2. Instead they are maintained by virtue of the states of the internal logic driving the pins being maintained.

Exit from stop3 is done by asserting $\overline{\text{RESET}}$ or by an interrupt from one of the following sources: the real-time interrupt (RTI), LVD system, ADC, IRQ, KBI, or SCI.

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

A separate self-clocked source (≈1 kHz) for the real-time interrupt allows a wakeup from stop2 or stop3 mode with no external components. When RTIS2:RTIS1:RTIS0 = 0:0:0, the real-time interrupt function and this 1-kHz source are disabled. Power consumption is lower when the 1-kHz source is disabled, but in that case the real-time interrupt cannot wake the MCU from stop.

### 3.6.3 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. This register is described in Chapter 15, “Development Support” of this data sheet. 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 so background debug communication is still possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If the user attempts to enter stop2 with ENBDM set, the MCU will instead enter stop3.

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. Table 3-2 summarizes the behavior of the MCU in stop when entry into the background debug mode is enabled.

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

| Mode | PPDC | CPU, Digital Peripherals, FLASH | RAM | ICG | ADC | Regulator | I/O Pins | RTI |
|---|---|---|---|---|---|---|---|---|
| Stop3 | 0 | Standby | Standby | Active | Optionally on | Active | States held | Optionally on |

## 3.6.4 LVD Enabled in Stop 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 by setting the LVDE and the LVDSE bits, then the voltage regulator remains active during stop mode. If the user attempts to enter stop2 with the LVD enabled for stop, the MCU will instead enter stop3. Table 3-3 summarizes the behavior of the MCU in stop when the LVD is enabled.

**Table 3-3. LVD Enabled Stop Mode Behavior**

| Mode | PPDC | CPU, Digital Peripherals, FLASH | RAM | ICG | ADC | Regulator | I/O Pins | RTI |
|---|---|---|---|---|---|---|---|---|
| Stop3 | 0 | Standby | Standby | Off | Optionally on | Active | States held | Optionally on |

## 3.6.5 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 are kept alive to the background debug logic, clocks to the peripheral systems are halted to reduce power consumption. Refer to Section 3.6.2, “Stop3 Mode” for specific information on system behavior in stop modes.

**I/O Pins**

- All I/O pin states remain unchanged when the MCU enters stop3 mode.
- If the MCU is configured to go into stop2 mode, all I/O pins states are latched before entering stop.

Memory

- All RAM and register contents are preserved while the MCU is in stop3 mode.
- All registers will be reset upon wake-up from stop2, but the contents of RAM are preserved and pin states remain latched until the PPDACK bit is written. The user may save any memory-mapped register data into RAM before entering stop2 and restore the data upon exit from stop2.
- The contents of the FLASH memory are non-volatile and are preserved in any of the stop modes.

**ICG** — In stop3 mode, the ICG enters its low-power standby state. The oscillator may be kept running when the ICG is in standby by setting OSCSTEN. In stop2 mode, the ICG is turned off. The oscillator cannot be kept running in stop2 even if OSCSTEN is set. If the MCU is configured to go into stop2 mode, the ICG will be reset upon wake-up from stop and must be reinitialized.

**TPM** — When the MCU enters stop mode, the clock to the TPM1 and TPM2 modules stop. The modules halt operation. If the MCU is configured to go into stop2 mode, the TPM modules will be reset upon wake-up from stop and must be reinitialized.

**ADC** — When the MCU enters stop mode, the ADC will enter a low-power standby state unless the asynchronous clock source, ADACK, is enabled. Conversions can occur in stop3 if ADACK is enabled. If the MCU is configured to go into stop2 mode, the ADC will be reset upon wake-up from stop and must be re-initialized.

**KBI** — During stop3, the KBI pins that are enabled continue to function as interrupt sources that are capable of waking the MCU from stop3. The KBI is disabled in stop2 and must be reinitialized after waking up.

**SCI** — When the MCU enters stop mode, the clocks to the SCI1 and SCI2 modules stop. The modules halt operation. If the MCU is configured to go into stop2 mode, the SCI modules will be reset upon wake-up from stop and must be reinitialized.

**SPI** — When the MCU enters stop mode, the clocks to the SPI module stop. The module halts operation. If the MCU is configured to go into stop2 mode, the SPI module will be reset upon wake-up from stop and must be reinitialized.

**IIC** — When the MCU enters stop mode, the clocks to the IIC module stops. The module halts operation. If the MCU is configured to go into stop2 mode, the IIC module will be reset upon wake-up from stop and must be reinitialized.

**Voltage Regulator** — The voltage regulator enters a low-power standby state when the MCU enters either of the stop modes unless the LVD is enabled in stop mode or BDM is enabled.

# Chapter 4
# Memory

## 4.1 MC9S08AC16 Series Memory Map

Figure 4-1 shows the memory maps for the MC9S08AC16 Series MCUs. On-chip memory in the MC9S08AC16 Series of MCU consists of RAM, FLASH program memory for nonvolatile data storage, plus I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x006F)
- High-page registers (0x1800 through 0x185F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

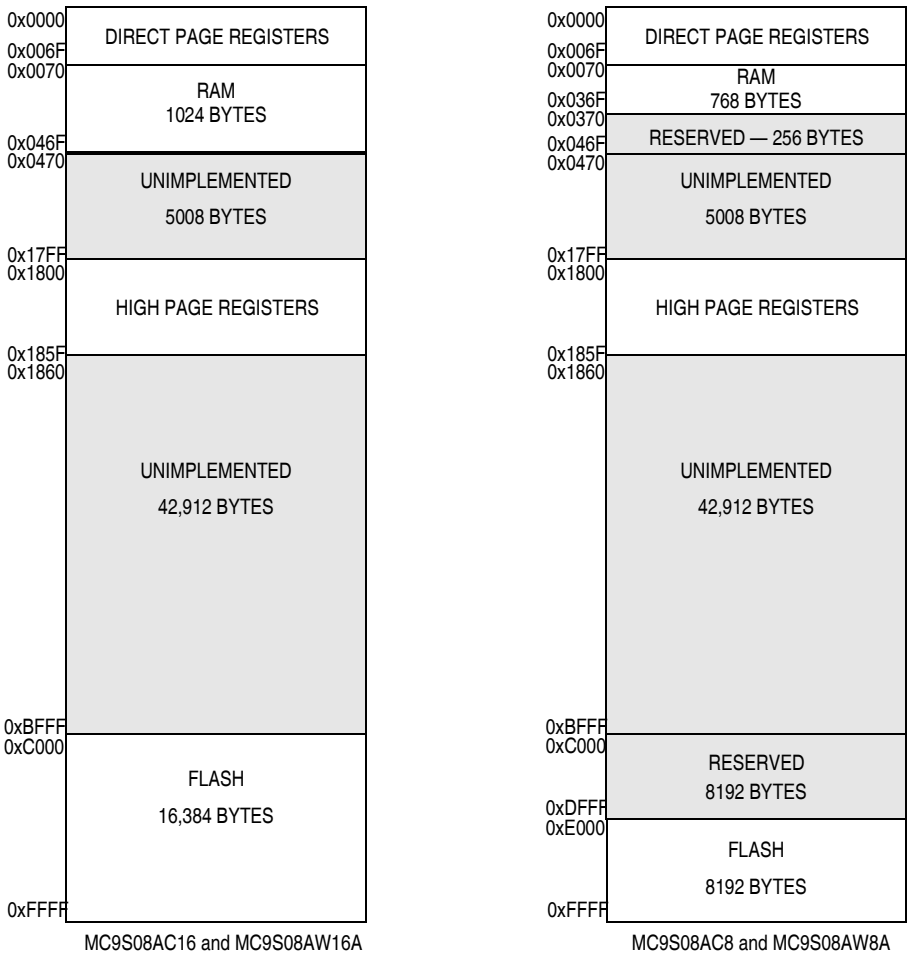



**Figure 4-1. MC9S08AC16 Series Memory Maps**

### 4.1.1 Reset and Interrupt Vector Assignments

Figure 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale-provided equate file for the MC9S08AC16 Series. For more details about resets, interrupts, interrupt priority, and local interrupt mask controls, refer to Chapter 5, "Resets, Interrupts, and System Configuration."

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

| Address (High/Low) | Vector | Vector Name |
|---|---|---|
| 0xFFC0:FFC1 through 0xFFC4:FFC5 | Unused Vector Space (available for user program) | — |
| 0xFFC6:FFC7 | TPM3 overflow | Vtpm3ovf |
| 0xFFC8:FFC9 | TPM3 channel 1 | Vtpm3ch1 |
| 0xFFCA:FFCB | TPM3 channel 0 | Vtpm3ch0 |
| 0xFFCC:FFCD | RTI | Vrti |
| 0xFFCE:FFCF | IIC1 | Viic1 |
| 0xFFD0:FFD1 | ADC1 Conversion | Vadc1 |
| 0xFFD2:FFD3 | KBI | Vkeyboard1 |
| 0xFFD4:FFD5 | SCI2 Transmit | Vsci2tx |
| 0xFFD6:FFD7 | SCI2 Receive | Vsci2rx |
| 0xFFD8:FFD9 | SCI2 Error | Vsci2err |
| 0xFFDA:FFDB | SCI1 Transmit | Vsci1tx |
| 0xFFDC:FFDD | SCI1 Receive | Vsci1rx |
| 0xFFDE:FFDF | SCI1 Error | Vsci1err |
| 0xFFE0:FFE1 | SPI1 | Vspi1 |
| 0xFFE2:FFE3 | TPM2 Overflow | Vtpm2ovf |
| 0xFFE4:FFE5 | TPM2 Channel 1 | Vtpm2ch1 |
| 0xFFE6:FFE7 | TPM2 Channel 0 | Vtpm2ch0 |
| 0xFFE8:FFE9 | TPM1 Overflow | Vtpm1ovf |
| 0xFFEA:FFEB | Unused | — |
| 0xFFEC:FFED | Unused | — |
| 0xFFEE:FFEF | TPM1 Channel 3 | Vtpm1ch3 |
| 0xFFF0:FFF1 | TPM1 Channel 2 | Vtpm1ch2 |
| 0xFFF2:FFF3 | TPM1 Channel 1 | Vtpm1ch1 |
| 0xFFF4:FFF5 | TPM1 Channel 0 | Vtpm1ch0 |
| 0xFFF6:FFF7 | ICG | Vicg |
| 0xFFF8:FFF9 | Low Voltage Detect | Vlvd |
| 0xFFFA:FFFB | IRQ | Virq |
| 0xFFFC:FFFD | SWI | Vswi |
| 0xFFFE:FFFF | Reset | Vreset |

## 4.2 Register Addresses and Bit Assignments

The registers in the MC9S08AC16 Series are divided into these three groups:

- Direct-page registers are located in the first 112 locations in the memory map, so they 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 variables.
- The nonvolatile register area consists of a block of 16 locations in FLASH memory at 0xFFB0–0xFFBF.

  Nonvolatile register locations include:

  — Three values which are loaded into working registers at reset

  — An 8-byte backdoor comparison key which 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 only requires 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 4-4 the whole address in column one is shown in bold. In Table 4-2, Table 4-3, and Table 4-4, 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 | R | R | R | R | PTAD2 | PTAD1 | PTAD0 |
| 0x0001 | **PTADD** | PTADD7 | R | R | R | R | PTADD2 | PTADD1 | PTADD0 |
| 0x0002 | **PTBD** | R | R | R | R | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| 0x0003 | **PTBDD** | R | R | R | R | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| 0x0004 | **PTCD** | 0 | R | PTCD5 | PTCD4 | PTCD3 | PTCD2 | PTCD1 | PTCD0 |
| 0x0005 | **PTCDD** | 0 | R | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| 0x0006 | **PTDD** | R | R | R | R | PTDD3 | PTDD2 | PTDD1 | PTDD0 |
| 0x0007 | **PTDDD** | R | R | R | R | 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** | R | PTFD6 | PTFD5 | PTFD4 | R | R | PTFD1 | PTFD0 |
| 0x000B | **PTFDD** | R | PTFDD6 | PTFDD5 | PTFDD4 | R | R | PTFDD1 | PTFDD0 |
| 0x000C | **PTGD** | 0 | PTGD6 | PTGD5 | PTGD4 | PTGD3 | PTGD2 | PTGD1 | PTGD0 |
| 0x000D | **PTGDD** | 0 | PTGDD6 | PTGDD5 | PTGDD4 | PTGDD3 | PTGDD2 | PTGDD1 | PTGDD0 |
| 0x000E–<br>0x000F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x0010 | **ADC1SC1** | COCO | AIEN | ADCO | ADCH | | | | |
| 0x0011 | **ADC1SC2** | ADACT | ADTRG | ACFE | ACFGT | 0 | 0 | R | R |
| 0x0012 | **ADC1RH** | 0 | 0 | 0 | 0 | 0 | 0 | ADR9 | ADR8 |
| 0x0013 | **ADC1RL** | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |
| 0x0014 | **ADC1CVH** | 0 | 0 | 0 | 0 | 0 | 0 | ADCV9 | ADCV8 |
| 0x0015 | **ADC1CVL** | ADCV7 | ADCV6 | ADCV5 | ADCV4 | ADCV3 | ADCV2 | ADCV1 | ADCV0 |
| 0x0016 | **ADC1CFG** | 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 | Reserved | — | — | — | — | — | — | — | — |
| 0x001E | **KBISC** | 0 | KBEDG6 | KBEDG5 | KBEDG4 | KBF | KBACK | KBIE | KBIMOD |
| 0x001F | **KBIPE** | 0 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| 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 |
| 0x0031–0x0037 | 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 | **ICGC1** | HGO | RANGE | REFS | CLKS | | OSCSTEN | LOCD | 0 |
| 0x0049 | **ICGC2** | LOLRE | MFD | | | LOCRE | RFD | | |
| 0x004A | **ICGS1** | CLKST | | REFST | LOLS | LOCK | LOCS | ERCS | ICGIF |
| 0x004B | **ICGS2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DCOS |
| 0x004C | **ICGFLTU** | 0 | 0 | 0 | 0 | FLT | | | |
| 0x004D | **ICGFLTL** | FLT | | | | | | | |
| 0x004E | **ICGTRM** | TRIM | | | | | | | |
| 0x004F | Reserved | — | — | — | — | — | — | — | — |
| 0x0050 | **SPI1C1** | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| 0x0051 | **SPI1C2** | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| 0x0052 | **SPI1BR** | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| 0x0053 | **SPI1S** | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| 0x0054 | Reserved | — | — | — | — | — | — | — | — |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0055 | **SPI1D** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0056–0x0057 | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x0058 | **IIC1A** | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | 0 |
| 0x0059 | **IIC1F** | MULT | | ICR | | | | | |
| 0x005A | **IIC1C1** | IICEN | IICIE | MST | TX | TXAK | RSTA | 0 | 0 |
| 0x005B | **IIC1S** | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| 0x005C | **IIC1D** | DATA | | | | | | | |
| 0x005D | **IIC1C2** | GCAEN | ADEXT | 0 | 0 | 0 | AD10 | AD9 | AD8 |
| 0x005E–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–0x006F | 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 | ICG | LVD | 0 |
| **0x1801** | **SBDFR** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BDFR |
| **0x1802** | **SOPT** | COPE | COPT | STOPE | — | 0 | 0 | — | — |
| **0x1803** | **SMCLK** | 0 | 0 | 0 | MPE | 0 | MCSEL | | |
| **0x1804 – 0x1805** | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| **0x1806** | **SDIDH** | REV3 | REV2 | REV1 | REV0 | ID11 | ID10 | ID9 | ID8 |
| **0x1807** | **SDIDL** | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| **0x1808** | **SRTISC** | RTIF | RTIACK | RTICLKS | RTIE | 0 | RTIS2 | RTIS1 | RTIS0 |
| **0x1809** | **SPMSC1** | LVDF | LVDACK | LVDIE | LVDRE | LVDSE | LVDE | 0[1] | BGBE |
| **0x180A** | **SPMSC2** | LVWF | LVWACK | LVDV | LVWV | PPDF | PPDACK | — | PPDC |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x180B | Reserved | — | — | — | — | — | — | — | — |
| 0x180C | SOPT2 | COPCLKS | — | — | — | — | — | — | — |
| 0x180D–0x180F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 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 | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1820 | FCDIV | DIVLD | PRDIV8 | DIV5 | DIV4 | DIV3 | DIV2 | DIV1 | DIV0 |
| 0x1821 | FOPT | KEYEN | FNORED | 0 | 0 | 0 | 0 | SEC01 | SEC00 |
| 0x1822 | Reserved | — | — | — | — | — | — | — | — |
| 0x1823 | FCNFG | 0 | 0 | KEYACC | 0 | 0 | 0 | 0 | 0 |
| 0x1824 | FPROT | FPS7 | FPS6 | FPS5 | FPS4 | FPS3 | FPS2 | FPS1 | FPDIS |
| 0x1825 | FSTAT | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| 0x1826 | FCMD | FCMD7 | FCMD6 | FCMD5 | FCMD4 | FCMD3 | FCMD2 | FCMD1 | FCMD0 |
| 0x1827–0x182F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1830 | TPM3SC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x1831 | TPM3CNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1832 | TPM3CNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1833 | TPM3MODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1834 | TPM3MODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1835 | TPM3C0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x1836 | TPM3C0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1837 | TPM3C0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1838 | TPM3C1SC | CH1F | CH1IE | MS1B | MS1A | ELS1B | ELS1A | 0 | 0 |
| 0x1839 | TPM3C1VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x183A | TPM3C1VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x183B 0x183F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1840 | PTAPE | PTAPE7 | R | R | R | R | PTAPE2 | PTAPE1 | PTAPE0 |
| 0x1841 | PTASE | PTASE7 | R | R | R | R | PTASE2 | PTASE1 | PTASE0 |
| 0x1842 | PTADS | PTADS7 | R | R | R | R | PTADS2 | PTADS1 | PTADS0 |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1843 | Reserved | — | — | — | — | — | — | — | — |
| 0x1844 | **PTBPE** | R | R | R | R | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| 0x1845 | **PTBSE** | R | R | R | R | PTBSE3 | PTBSE2 | PTBSE1 | PTBSE0 |
| 0x1846 | **PTBDS** | R | R | R | R | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| 0x1847 | Reserved | — | — | — | — | — | — | — | — |
| 0x1848 | **PTCPE** | 0 | R | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| 0x1849 | **PTCSE** | 0 | R | PTCSE5 | PTCSE4 | PTCSE3 | PTCSE2 | PTCSE1 | PTCSE0 |
| 0x184A | **PTCDS** | 0 | R | PTCDS5 | PTCDS4 | PTCDS3 | PTCDS2 | PTCDS1 | PTCDS0 |
| 0x184B | Reserved | — | — | — | — | — | — | — | — |
| 0x184C | **PTDPE** | R | R | R | R | PTDPE3 | PTDPE2 | PTDPE1 | PTDPE0 |
| 0x184D | **PTDSE** | R | R | R | R | PTDSE3 | PTDSE2 | PTDSE1 | PTDSE0 |
| 0x184E | **PTDDS** | R | R | R | R | PTDDS3 | PTDDS2 | PTDDS1 | PTDDS0 |
| 0x184F | Reserved | — | — | — | — | — | — | — | — |
| 0x1850 | **PTEPE** | PTEPE7 | PTEPE6 | PTEPE5 | PTEPE4 | PTEPE3 | PTEPE2 | PTEPE1 | PTEPE0 |
| 0x1851 | **PTESE** | PTESE7 | PTESE6 | PTESE5 | PTESE4 | PTESE3 | PTESE2 | PTESE1 | PTESE0 |
| 0x1852 | **PTEDS** | PTEDS7 | PTEDS6 | PTEDS5 | PTEDS4 | PTEDS3 | PTEDS2 | PTEDS1 | PTEDS0 |
| 0x1853 | Reserved | — | — | — | — | — | — | — | — |
| 0x1854 | **PTFPE** | R | PTFPE6 | PTFPE5 | PTFPE4 | R | R | PTFPE1 | PTFPE0 |
| 0x1855 | **PTFSE** | R | PTFSE6 | PTFSE5 | PTFSE4 | R | R | PTFSE1 | PTFSE0 |
| 0x1856 | **PTFDS** | R | PTFDS6 | PTFDS5 | PTFDS4 | R | R | PTFDS1 | PTFDS0 |
| 0x1857 | Reserved | — | — | — | — | — | — | — | — |
| 0x1858 | **PTGPE** | 0 | PTGPE6 | PTGPE5 | PTGPE4 | PTGPE3 | PTGPE2 | PTGPE1 | PTGPE0 |
| 0x1859 | **PTGSE** | 0 | PTGSE6 | PTGSE5 | PTGSE4 | PTGSE3 | PTGSE2 | PTGSE1 | PTGSE0 |
| 0x185A | **PTGDS** | 0 | PTGDS6 | PTGDS5 | PTGDS4 | PTGDS3 | PTGDS2 | PTGDS1 | PTGDS0 |
| 0x185B–<br>0x185F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |

[1] This reserved bit must always be written to 0.

Nonvolatile FLASH registers, shown in Table 4-4, are located in the FLASH memory. These registers include an 8-byte backdoor key which optionally 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 4-4. Nonvolatile Register Summary**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0xFFB0 – 0xFFB7 | NVBACKKEY | 8-Byte Comparison Key | | | | | | | |
| 0xFFB8 – 0xFFBB | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| 0xFFBC | Reserved for storage of 250 kHz ICGTRM value | — | — | — | — | — | — | — | — |
| 0xFFBD | NVPROT | FPS7 | FPS6 | FPS5 | FPS4 | FPS3 | FPS2 | FPS1 | FPDIS |
| 0xFFBE | Reserved for storage of 243 kHz ICGTRM value | — | — | — | — | — | — | — | — |
| 0xFFBF | NVOPT | KEYEN | FNORED | 0 | 0 | 0 | 0 | SEC01 | SEC00 |

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 (SEC01:SEC00) to the unsecured state (1:0).

## 4.3 RAM

The MC9S08AC16 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 when 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 provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08AC16 Series, it is usually best to re-initialize 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-provided 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 through code executing from non-secure memory. See Section 4.5, “Security” for a detailed description of the security feature.

## 4.4 FLASH

The FLASH memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the FLASH memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for FLASH 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/D.

### 4.4.1 Features

Features of the FLASH memory include:

- FLASH Size
  - MC9S08AC16 and MC9S08AW16A— 16,384 bytes (32 pages of 512 bytes each)
  - MC9S08AC8 and MC9S08AW8A— 8192 bytes (16 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fastprogram and erase operation
- Up to 100,000 program/erase cycles attypical voltage and temperature
- Flexible block protection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

### 4.4.2 Program and Erase Times

Before any program or erase command can be accepted, the FLASH clock divider register (FCDIV) must be written to set the internal clock for the FLASH module to a frequency ($f_{FCLK}$) between 150 kHz and 200 kHz (see Section 4.6.1, “FLASH Clock Divider Register (FCDIV)”). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. 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 are used by the command processor to complete a program or erase command.

Table 4-5 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 4-5. Program and Erase Times**

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

[1] Excluding start/end overhead

### 4.4.3 Program and Erase Command Execution

The steps for executing any of the commands are listed below. The FCDIV register must be initialized and any error flags cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the FLASH array. The address and data information from this write is latched into the FLASH 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 page erase commands, the address may be any address in the 512-byte page of FLASH to be erased. For mass erase and blank check commands, the address can be any address in the FLASH memory. Whole pages of 512 bytes are the smallest block of FLASH that may be erased. In the 60K version, there are two instances where the size of a block that is accessible to the user is less than 512 bytes: the first page following RAM, and the first page following the high page registers. These pages are overlapped by the RAM and high page registers respectively.

**NOTE**

Do not program any byte in the FLASH more than once after a successful erase operation. Reprogramming bits to a byte which is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire FLASH memory. Programming without first erasing may disturb data stored in the FLASH.

2. Write the command code for the desired command to FCMD. The five valid commands are blank check (0x05), byte program (0x20), burst program (0x25), page erase (0x40), and mass erase (0x41). 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 FLASH 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. The FCDIV register must be initialized before using any FLASH commands. This only must be done once following a reset.

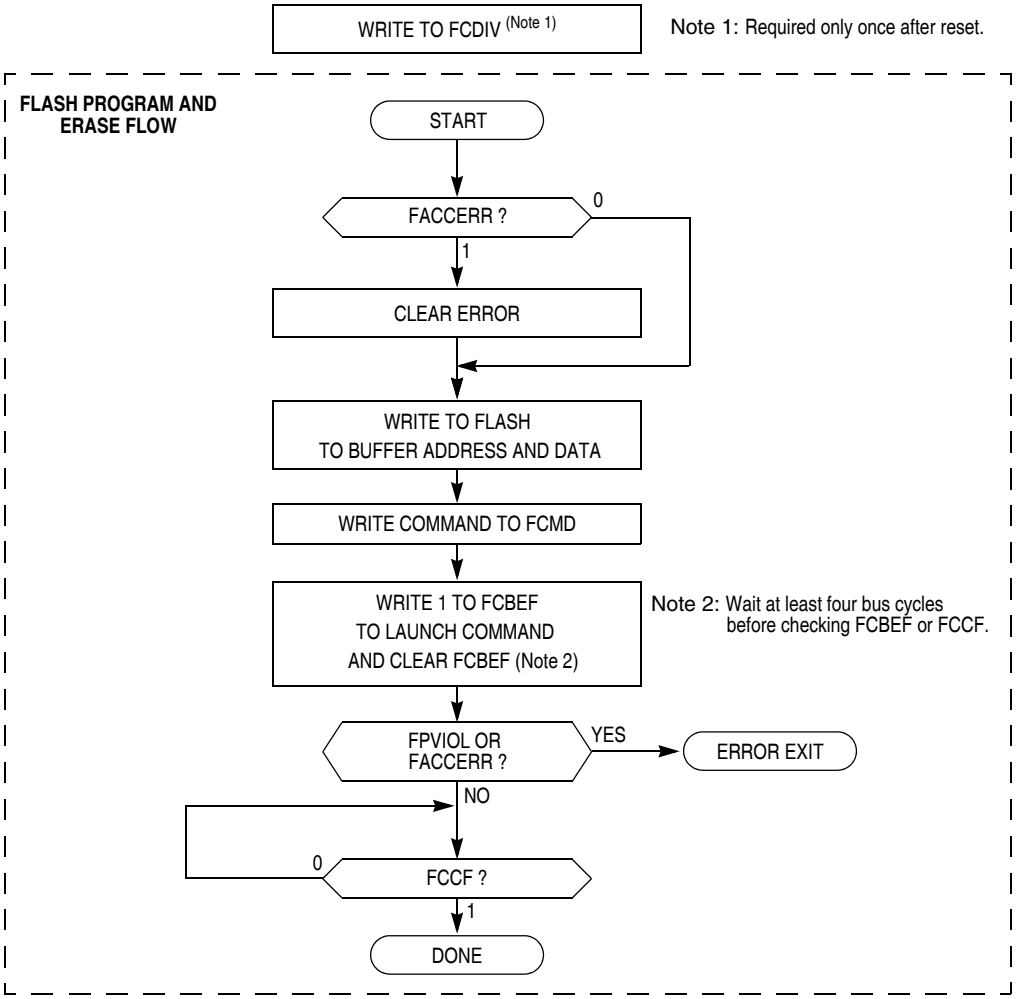



**Figure 4-2. FLASH Program and Erase Flowchart**

## 4.4.4 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 then remains enabled after completion of the burst program operation if these two conditions are met:

- The next burst program command has been queued before the current program operation has completed.

- The next sequential address selects a byte on the same physical row as the current byte being programmed. A row of FLASH memory consists of 64 bytes. A byte within a row is selected by addresses A5 through A0. A new row begins when addresses A5 through A0 are all zero.

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. In the case 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.

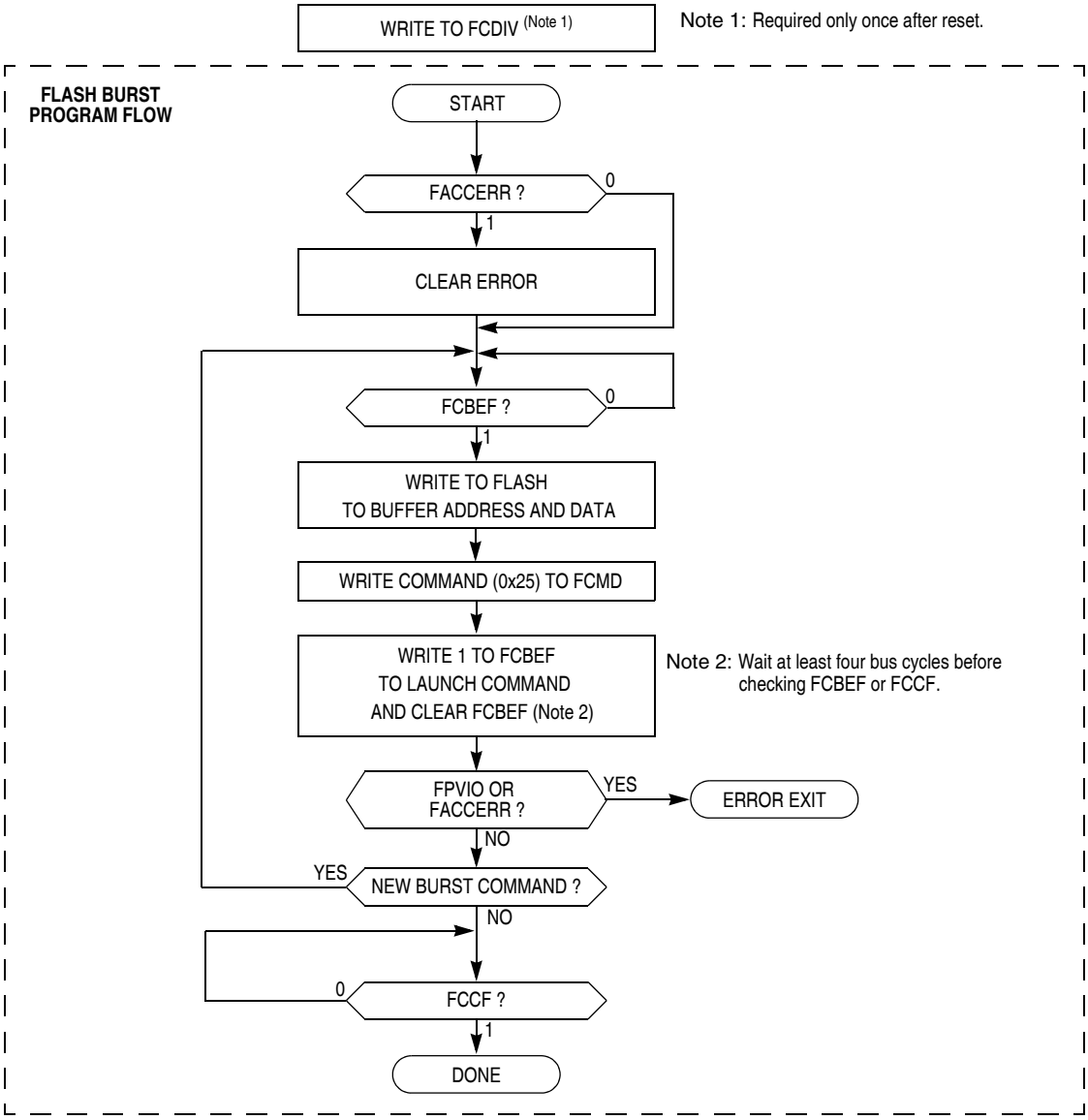



**Figure 4-3. FLASH Burst Program Flowchart**

### 4.4.5 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 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 five allowed codes (0x05, 0x20, 0x25, 0x40, or 0x41) to FCMD
- Writing any FLASH control register other than the 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, or page erase command code (0x20, 0x25, or 0x40) with a background debug command while the MCU is secured (The background debug controller can only do blank check and mass erase commands when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command

### 4.4.6 FLASH Block Protection

The block protection feature prevents the protected region of FLASH from program or erase changes. Block protection is controlled through the FLASH protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of FLASH, 0xFFFF. (See Section 4.6.4, “FLASH 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. FPROT cannot be changed directly from application software so a runaway program cannot alter the block protection settings. Because NVPROT is within the last 512 bytes of FLASH, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands, which allows a way to erase and reprogram a protected FLASH memory.

The block protection mechanism is illustrated in Figure 4-4. The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, to protect the last 1536 bytes of memory (addresses 0xFA00 through 0xFFFF), the FPS bits must be set to 1111 100, which results in the value 0xF9FF as the last address of unprotected

memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT) must be programmed to logic 0 to enable block protection. Therefore the value 0xF8 must be programmed into NVPROT to protect addresses 0xFA00 through 0xFFFF.

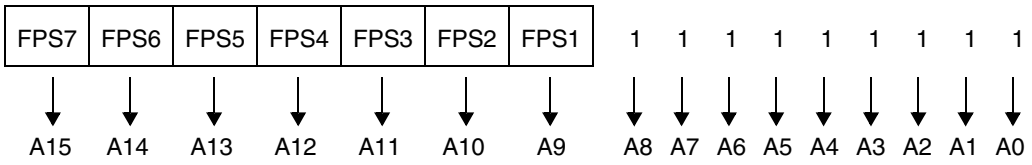



**Figure 4-4. Block Protection Mechanism**

One use for block protection is to block protect an area of FLASH memory for a bootloader program. This bootloader program then can be used to erase the rest of the FLASH memory and reprogram it. Because the bootloader is protected, it remains intact even if MCU power is lost in the middle of an erase and reprogram operation.

### 4.4.7 Vector Redirection

Whenever any block protection is enabled, 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 zero. For redirection to occur, at least some portion but not all of the FLASH memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFD) are redirected, though the reset vector (0xFFFE:FFFF) is not.

For example, if 512 bytes of FLASH are protected, the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFD) are redirected to the locations 0xFDC0–0xFDFD. Now, if an SPI interrupt is taken for instance, the values in the locations 0xFDE0:FDE1 are used for the vector instead of the values in the locations 0xFFE0:FFE1. 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 Security

The MC9S08AC16 Series includes circuitry to prevent unauthorized access to the contents of FLASH and RAM memory. When security is engaged, FLASH 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 nonvolatile register bits (SEC01:SEC00) 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 done at the same time the FLASH memory is programmed. The 1:0 state

disengages security and 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 SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 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 still be used for background memory access commands, but the MCU cannot enter active background mode except by holding BKGD/MS 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 done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX should not be used for these writes because these writes cannot be done 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 just written matches the key stored in the FLASH locations, SEC01:SEC00 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 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 512-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 SEC01:SEC00 = 1:0.

## 4.6 FLASH Registers and Control Bits

The FLASH module has nine 8-bit registers in the high-page register space, three locations in the nonvolatile register space in FLASH memory which are copied into three 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 4-4 for the absolute address assignments for all FLASH registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

### 4.6.1 FLASH Clock Divider Register (FCDIV)

Bit 7 of this register is a read-only status flag. Bits 6 through 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.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DIVLD | PRDIV8 | DIV5 | DIV4 | DIV3 | DIV2 | DIV1 | DIV0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 4-5. FLASH Clock Divider Register (FCDIV)**

**Table 4-6. 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.<br>1 FCDIV has been written since reset; erase and program operations enabled for FLASH. |
| 6<br>PRDIV8 | **Prescale (Divide) FLASH Clock by 8**<br>0 Clock input to the FLASH clock divider is the bus rate clock.<br>1 Clock input to the FLASH clock divider is the bus rate clock divided by 8. |
| 5:0<br>DIV[5:0] | **Divisor for FLASH Clock Divider** — The FLASH 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 DIV5:DIV0 field plus one. The resulting frequency of the internal FLASH clock must fall within the range of 200 kHz to 150 kHz for proper FLASH operations. Program/Erase timing pulses are one cycle of this internal FLASH 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, Equation 4-2, and Table 4-6. |

$$\text{if PRDIV8 = 0 — } f_{FCLK} = f_{Bus} \div ([DIV5:DIV0] + 1) \qquad \textit{Eqn. 4-1}$$

$$\text{if PRDIV8 = 1 — } f_{FCLK} = f_{Bus} \div (8 \times ([DIV5:DIV0] + 1)) \qquad \textit{Eqn. 4-2}$$

Table 4-7 shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.

**Table 4-7. FLASH Clock Divider Settings**

| $f_{Bus}$ | PRDIV8 (Binary) | DIV5:DIV0 (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 |

## 4.6.2 FLASH Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into FOPT. Bits 5 through 2 are not used and always read 0. This register may be read at any time, but writes have no meaning or effect. 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 | 0 | 0 | 0 | 0 | SEC01 | SEC00 |
| W | | | | | | | | |
| Reset | This register is loaded from nonvolatile location NVOPT during reset. | | | | | | | |

= Unimplemented or Reserved

**Figure 4-6. FLASH Options Register (FOPT)**

**Table 4-8. 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, “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. |
| 1:0<br>SEC0[1:0] | **Security State Code** — This 2-bit field determines the security state of the MCU as shown in Table 4-9. When the MCU is secure, the contents of RAM and FLASH memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to Section 4.5, “Security.” |

**Table 4-9. Security States**

| SEC01:SEC00 | Description |
|---|---|
| 0:0 | secure |
| 0:1 | secure |
| 1:0 | unsecured |
| 1:1 | secure |

SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of FLASH.

## 4.6.3 FLASH Configuration Register (FCNFG)

Bits 7 through 5 may be read or written at any time. Bits 4 through 0 always read 0 and cannot be written.

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

= Unimplemented or Reserved

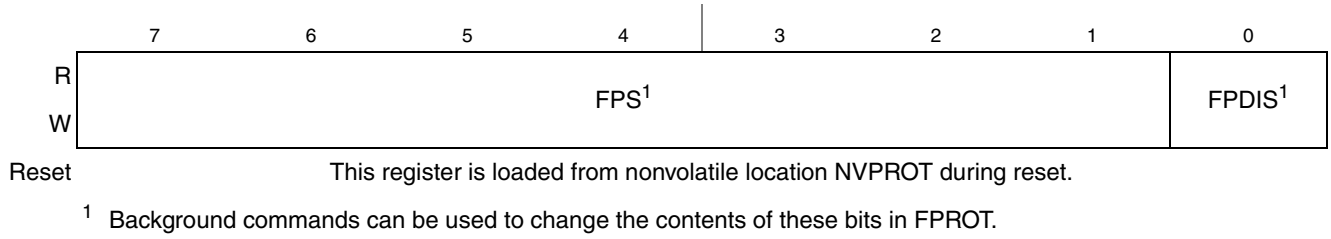
**Figure 4-7. FLASH Configuration Register (FCNFG)**

**Table 4-10. FCNFG Register Field Descriptions**

| Field | Description |
|---|---|
| 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, “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. |

## 4.6.4 FLASH Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT are copied from FLASH into FPROT. This register can be read at any time. If FPDIS = 0, protection can be increased, i.e., a smaller value of FPS can be written. If FPDIS = 1, writes do not change protection.




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

**Figure 4-8. FLASH Protection Register (FPROT)**

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

| Field | Description |
|---|---|
| 7:1<br>FPS[7:1] | **FLASH Protect Select Bits** — When FPDIS = 0, this 7-bit field determines the ending address of unprotected FLASH locations at the high address end of the FLASH. Protected FLASH locations cannot be erased or programmed. |
| 0<br>FPDIS | **FLASH Protection Disable**<br>0 FLASH block specified by FPS[7:1] is block protected (program and erase not allowed).<br>1 No FLASH block is protected. |

## 4.6.5 FLASH Status Register (FSTAT)

Bits 3, 1, and 0 always read 0 and writes have no meaning or effect. The remaining five bits are status bits that can be read at any time. Writes to these bits have special meanings that are discussed in the bit descriptions.

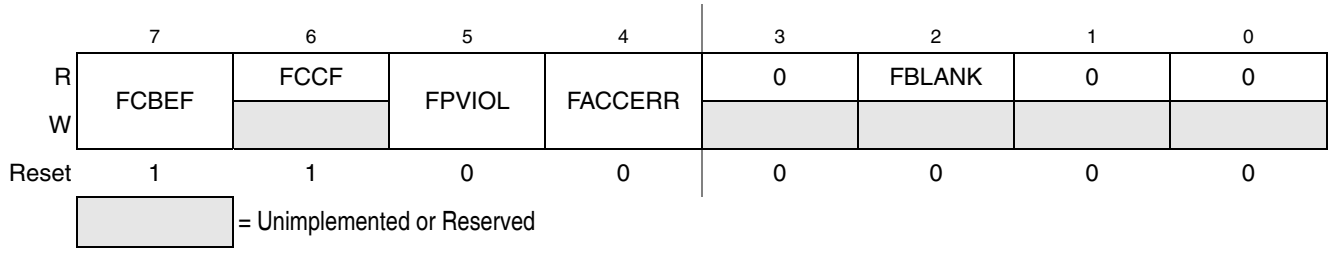



**Figure 4-9. FLASH Status Register (FSTAT)**

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

| Field | Description |
|---|---|
| 7<br>FCBEF | **FLASH 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 one 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 may be written to the command buffer. |
| 6<br>FCCF | **FLASH 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 FCBEF is cleared to register a command that attempts to erase or program a location in a protected block (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. |
| 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 4.4.5, “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 | **FLASH Verified as All Blank (erased) Flag** — FBLANK is set automatically at the conclusion of a blank check command if the entire FLASH 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 array is not completely erased.<br>1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the FLASH array is completely erased (all 0xFF). |

## 4.6.6 FLASH Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in Table 4-14. Refer to Section 4.4.3, “Program and Erase Command Execution” for a detailed discussion of FLASH programming and erase operations.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | FCMD7 | FCMD6 | FCMD5 | FCMD4 | FCMD3 | FCMD2 | FCMD1 | FCMD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-10. FLASH Command Register (FCMD)**

**Table 4-13. FCMD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>FCMD[7:0] | **FLASH Command Bits** — See Table 4-14 |

**Table 4-14. FLASH Commands**

| Command | FCMD | Equate File Label |
|---|---|---|
| Blank check | 0x05 | mBlank |
| Byte program | 0x20 | mByteProg |
| Byte program — burst mode | 0x25 | mBurstProg |
| Page erase (512 bytes/page) | 0x40 | mPageErase |
| Mass erase (all FLASH) | 0x41 | mMassErase |

All other command codes are illegal and generate an access error.

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 System Configuration

## 5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08AC16 Series. Some interrupt sources from peripheral modules are discussed in greater detail within other chapters of this data manual. This chapter 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 and real-time interrupt (RTI), are not part of on-chip peripheral systems with their own sections but are part of the system control logic.

## 5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation:
  - Power-on detection (POR)
  - Low voltage detection (LVD) with enable
  - External $\overline{\text{RESET}}$ pin
  - COP watchdog with enable and two timeout choices
  - Illegal opcode
  - Illegal address
  - Serial command from a background debug host
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see Table 5-11)

## 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 pullup 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. SP is forced to 0x00FF at reset.

The following sources of reset are available on the MC9S08AC16 Series:

- Power-on reset (POR)
- Low-voltage detect (LVD)

- Computer operating properly (COP) timer
- Illegal opcode detect
- Illegal address detect
- Background debug forced reset
- The reset pin ($\overline{\text{RESET}}$)
- Clock generator loss of lock and loss of clock reset

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

## 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 COPE becomes set in SOPT enabling the COP watchdog (see Section 5.9.4, “System Options Register (SOPT),” for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The COPCLKS bit in SOPT2 (see Section 5.9.10, “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 is an associated short and long time-out controlled by COPT in SOPT. Table 5-1 summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the bus clock source and the associated long time-out ($2^{18}$ cycles).

**Table 5-1. COP Configuration Options**

| Control Bits | | Clock Source | COP Overflow Count |
|---|---|---|---|
| COPCLKS | COPT | | |
| 0 | 0 | ~1 kHz | $2^5$ cycles (32 ms)[1] |
| 0 | 1 | ~1 kHz | $2^8$ cycles (256 ms)[1] |
| 1 | 0 | Bus | $2^{13}$ cycles |
| 1 | 1 | Bus | $2^{18}$ cycles |

[1] Values are shown in this column based on $t_{RTI}$ = 1 ms. See $t_{RTI}$ in the appendix Section A.10.1, “Control Timing,” for the tolerance of this value.

Even if the application will use the reset default settings of COPE, COPCLKS, and COPT, the user must write to the write-once SOPT and SOPT2 registers during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost. The initial writes to SOPT and SOPT2 will reset the COP counter.

The write to SRS that services (clears) the COP counter must 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.

In background debug mode, the COP counter will not increment.

When the bus clock source is selected, the COP counter does not increment while the system is in stop mode. The COP counter resumes as soon as the MCU exits stop mode.

When the 1-kHz clock source is selected, the COP counter is re-initialized to zero upon entry to stop mode. The COP counter begins from zero after the MCU exits 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 until and unless the local interrupt enable is a logic 1 to enable the interrupt. 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 masks (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 may 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 off the stack.

**NOTE**

For compatibility with the M68HC08, 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.

When two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-2).

### 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.

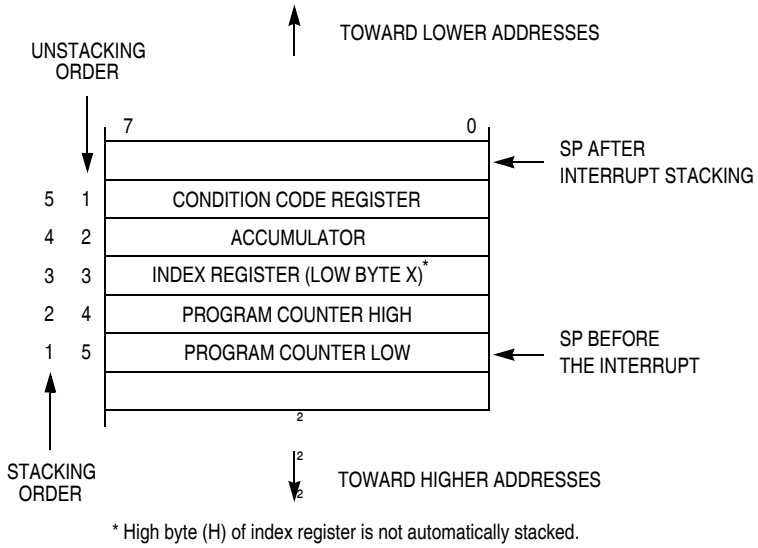



**Figure 5-1. Interrupt Stack Frame**

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 causing the interrupt must be acknowledged (cleared) before returning from the ISR. Typically, the flag should be 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 IRQ 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.

**NOTE**

This 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 IRQ pin may be as low as $V_{DD} - 0.7$ V. The internal gates connected to this pin are pulled all the way to $V_{DD}$.

**NOTE**

When enabling the IRQ pin for use, the IRQF will be set, and should be cleared prior to enabling the interrupt. When configuring the pin for falling edge and level sensitivity in a 5V system, it is necessary to wait at least 6 cycles between clearing the flag and enabling the interrupt.

#### 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 5-2 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 5-2. Vector Summary**

| Vector Priority | Vector No. | Address (High/Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|---|
| Lower | 29 – 31 | 0xFFC0/FFC1 – 0xFFC4/0xFFC5 | Unused vector space (available for user program) | | | | |
| | 28 | 0xFFC6/FFC7 | Vtpm3ovf | TPM3 | TOF | TOIE | TPM3 overflow |
| | 27 | 0xFFC8/FFC9 | Vtpm3ch1 | TPM3 | CH1F | CH1IE | TPM3 channel 1 |
| | 26 | 0xFFCA/FFCB | Vtpm3ch0 | TPM3 | CH0F | CH0IF | TPM3 channel 0 |
| | 25 | 0xFFCC/FFCD | Vrti | System control | RTIF | RTIE | Real-time interrupt |
| | 24 | 0xFFCE/FFCF | Viic1 | IIC1 | IICIF | IICIE | IIC1 |
| | 23 | 0xFFD0/FFD1 | Vadc1 | ADC1 | COCO | AIEN | ADC1 |
| | 22 | 0xFFD2/FFD3 | Vkeyboard [1] | KBI | KBF | KBIE | KBI pins |
| | 21 | 0xFFD4/FFD5 | Vsci2tx | SCI2 | TDRE<br>TC | TIE<br>TCIE | SCI2 transmit |
| | 20 | 0xFFD6/FFD7 | Vsci2rx | SCI2 | IDLE<br>RDRF | ILIE<br>RIE | SCI2 receive |
| | 19 | 0xFFD8/FFD9 | Vsci2err | SCI2 | OR<br>NF<br>FE<br>PF | ORIE<br>NFIE<br>FEIE<br>PFIE | SCI2 error |
| | 18 | 0xFFDA/FFDB | Vsci1tx | SCI1 | TDRE<br>TC | TIE<br>TCIE | SCI1 transmit |
| | 17 | 0xFFDC/FFDD | Vsci1rx | SCI1 | IDLE<br>RDRF | ILIE<br>RIE | SCI1 receive |
| | 16 | 0xFFDE/FFDF | Vsci1err | SCI1 | OR<br>NF<br>FE<br>PF | ORIE<br>NFIE<br>FEIE<br>PFIE | SCI1 error |
| | 15 | 0xFFE0/FFE1 | Vspi1 | SPI1 | SPIF<br>MODF<br>SPTEF | SPIE<br>SPIE<br>SPTIE | SPI1 |
| | 14 | 0xFFE2/FFE3 | Vtpm2ovf | TPM2 | TOF | TOIE | TPM2 overflow |
| | 13 | 0xFFE4/FFE5 | Vtpm2ch1 | TPM2 | CH1F | CH1IE | TPM2 channel 1 |
| | 12 | 0xFFE6/FFE7 | Vtpm2ch0 | TPM2 | CH0F | CH0IE | TPM2 channel 0 |
| | 11 | 0xFFE8/FFE9 | Vtpm1ovf | TPM1 | TOF | TOIE | TPM1 overflow |
| | 10 | 0xFFEA/FFEB | Unused vector space | | | | |
| | 9 | 0xFFEC/FFED | Unused vector space | | | | |
| | 8 | 0xFFEE/FFEF | Vtpm1ch3 | TPM1 | CH3F | CH3IE | TPM1 channel 3 |
| | 7 | 0xFFF0/FFF1 | Vtpm1ch2 | TPM1 | CH2F | CH2IE | TPM1 channel 2 |
| | 6 | 0xFFF2/FFF3 | Vtpm1ch1 | TPM1 | CH1F | CH1IE | TPM1 channel 1 |
| | 5 | 0xFFF4/FFF5 | Vtpm1ch0 | TPM1 | CH0F | CH0IE | TPM1 channel 0 |
| | 4 | 0xFFF6/FFF7 | Vicg | ICG | ICGIF<br>(LOLS/LOCS) | LOLRE/LOCRE | ICG |
| | 3 | 0xFFF8/FFF9 | Vlvd | System control | LVDF | LVDIE | Low-voltage detect |
| | 2 | 0xFFFA/FFFB | Virq | IRQ | IRQF | IRQIE | IRQ pin |
| | 1 | 0xFFFC/FFFD | Vswi | Core | SWI Instruction | — | Software interrupt |
| Higher | 0 | 0xFFFE/FFFF | Vreset | System control | COP<br>LVD<br>$\overline{\text{RESET}}$ pin<br>Illegal opcode | COPE<br>LVDRE<br>—<br>— | Watchdog timer<br>Low-voltage detect<br>External pin<br>Illegal opcode |

## 5.6 Low-Voltage Detect (LVD) System

The MC9S08AC16 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 an LVD circuit with a user selectable trip voltage, either high ($V_{LVDH}$) or low ($V_{LVDL}$). The LVD circuit is enabled when LVDE in SPMSC1 is high and the trip voltage is selected by LVDV in SPMSC2. The LVD is disabled upon entering any of the stop modes unless the LVDSE bit is set. If LVDSE and LVDE are both set, then the MCU cannot enter stop2, and the current consumption in stop3 with the LVD enabled will be greater.

### 5.6.1 Power-On Reset Operation

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

### 5.6.2 LVD Reset Operation

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

### 5.6.3 LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF will be set and an LVD interrupt will occur.

### 5.6.4 Low-Voltage Warning (LVW)

The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching, but is still above, the LVD voltage. The LVW does not have an interrupt associated with it. There are two user selectable trip voltages for the LVW, one high ($V_{LVWH}$) and one low ($V_{LVWL}$). The trip voltage is selected by LVWV in SPMSC2. Setting the LVW trip voltage equal to the LVD trip voltage is not recommended. Typical use of the LVW would be to select $V_{LVWH}$ and $V_{LVDL}$.

## 5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts. The RTI can accept two sources of clocks, the 1-kHz internal clock or an external clock if available. The 1-kHz internal clock source is completely independent of any bus clock source and is used only by the RTI module and, on some MCUs, the COP watchdog. To use an external clock source, it must be available and active. The RTICLKS bit in SRTISC is used to select the RTI clock source.

Either RTI clock source can be used when the MCU is in run, wait or stop3 mode. When using the external oscillator in stop3, it must be enabled in stop (OSCSTEN = 1) and configured for low bandwidth operation (RANGE = 0). Only the internal 1-kHz clock source can be selected to wake the MCU from stop2 mode.

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS2:RTIS1:RTIS0) used to disable the clock source to the real-time interrupt or select one of seven wakeup periods. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The RTI can be disabled by writing each bit of RTIS to zeroes, and no interrupts will be generated. See Section 5.9.7, “System Real-Time Interrupt Status and Control Register (SRTISC),” for detailed information about this register.

## 5.8 MCLK Output

The PTC2 pin is shared with the MCLK clock output. Setting the pin enable bit, MPE, causes the PTC2 pin to output a divided version of the internal MCU bus clock. The divide ratio is determined by the MCSEL bits. When MPE is set, the PTC2 pin is forced to operate as an output pin regardless of the state of the port data direction control bit for the pin. If the MCSEL bits are all 0s, the pin is driven low. The slew rate and drive strength for the pin are controlled by PTCSE2 and PTCDS2, respectively. The maximum clock output frequency is limited if slew rate control is enabled, see the electrical chapter for pin rise and fall times with slew rate enabled.

## 5.9 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 the direct-page register summary 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 SOPT 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.”

## 5.9.1 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 5-3. 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. |

## 5.9.2 System Reset Status Register (SRS)

This register includes seven 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 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 | ICG | LVD | 0 |
| W | Writing any value to SRS address clears COP watchdog timer. | | | | | | | |
| POR | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| LVR: | U | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Any other reset: | 0 | Note[1] | Note[1] | Note[1] | 0 | Note[1] | 0 | 0 |

U = Unaffected by reset

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

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

**Table 5-4. 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 (LVR) status bit is also set to indicate that the reset occurred while the internal supply was below the LVR 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 may 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. |

**Table 5-4. SRS Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ILAD | **Illegal Address** — Reset was caused by an attempt to access a designated illegal address.<br>0 Reset not caused by an illegal address access.<br>1 Reset caused by an illegal address access.<br><br>Illegal address areas in the MC9S08AC16 are:<br>0x0470 - 0x17FF — Gap from end of RAM to start of high page registers<br>0x1860 - 0xBFFF — Gap from end of high page registers to start of Flash memory<br><br>Unused and reserved locations in register areas are not considered illegal addresses and do not trigger illegal address resets. |
| 2<br>ICG | **Internal Clock Generation Module Reset** — Reset was caused by an ICG module reset.<br>0 Reset not caused by ICG module.<br>1 Reset caused by ICG module. |
| 1<br>LVD | **Low Voltage Detect** — If the LVDRE and LVDSE bits are 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. |

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

This 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 5-5. SBDFR Register Field Descriptions**

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

## 5.9.4 System Options Register (SOPT)

This register may be read at any time. Bits 3 and 2 are unimplemented and always read 0. This is a write-once register so only the first write after reset is honored. Any subsequent attempt to write to SOPT (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT 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 | COPE | COPT | STOPE | | 0 | 0 | | |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |

= Unimplemented or Reserved

**Figure 5-5. System Options Register (SOPT)**

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

| Field | Description |
|---|---|
| 7<br>COPE | **COP Watchdog Enable** — This write-once bit defaults to 1 after reset.<br>0 COP watchdog timer disabled.<br>1 COP watchdog timer enabled (force reset on timeout). |
| 6<br>COPT | **COP Watchdog Timeout** — This write-once bit defaults to 1 after reset.<br>0 Short timeout period selected.<br>1 Long timeout period selected. |
| 5<br>STOPE | **Stop Mode Enable** — This write-once bit defaults to 0 after reset, which disables 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.1Stop mode enabled. |

## 5.9.5 System MCLK Control Register (SMCLK)

This register is used to control the MCLK clock output.

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

= Unimplemented or Reserved

**Figure 5-6. System MCLK Control Register (SMCLK)**

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

| Field | Description |
|---|---|
| 4<br>MPE | **MCLK Pin Enable** — This bit is used to enable the MCLK function.<br>0 MCLK output disabled.<br>1 MCLK output enabled on PTC2 pin. |
| 2:0<br>MCSEL | **MCLK Divide Select** — These bits are used to select the divide ratio for the MCLK output according to the formula below when the MCSEL bits are not equal to all zeroes. In the case that the MCSEL bits are all zero and MPE is set, the pin is driven low. See Equation 5-1.<br>**MCLK frequency = Bus Clock frequency ÷ (2 * MCSEL)** ***Eqn. 5-1*** |

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

This read-only register is 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 | | | | | ID11 | ID10 | ID9 | ID8 |
| W | | | | | | | | |
| Reset | — | — | — | — | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

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

| Field | Description |
|---|---|
| 7:4<br>Reserved | **Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.** |
| 3:0<br>ID[11:8] | **Part Identification Number** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08AC16 Series is hard coded to the value 0x012. 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 | 1 | 0 | 0 | 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** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08AC16 Series is hard coded to the value 0x012. See also ID bits in Table 5-8. |

## 5.9.7 System Real-Time Interrupt Status and Control Register (SRTISC)

This register contains one read-only status flag, one write-only acknowledge bit, three read/write delay selects, and three unimplemented bits, which always read 0.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIF | 0 | RTICLKS | RTIE | 0 | RTIS2 | RTIS1 | RTIS0 |
| W | | RTIACK | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 5-9. System RTI Status and Control Register (SRTISC)**

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

| Field | Description |
|---|---|
| 7<br>RTIF | **Real-Time Interrupt Flag** — This read-only status bit indicates the periodic wakeup timer has timed out.<br>0 Periodic wakeup timer not timed out.<br>1 Periodic wakeup timer timed out. |
| 6<br>RTIACK | **Real-Time Interrupt Acknowledge** — This write-only bit is used to acknowledge real-time interrupt request (write 1 to clear RTIF). Writing 0 has no meaning or effect. Reads always return logic 0. |
| 5<br>RTICLKS | **Real-Time Interrupt Clock Select** — This read/write bit selects the clock source for the real-time interrupt.<br>0 Real-time interrupt request clock source is internal 1-kHz oscillator.<br>1 Real-time interrupt request clock source is external clock. |
| 4<br>RTIE | **Real-Time Interrupt Enable** — This read-write bit enables real-time interrupts.<br>0 Real-time interrupts disabled.<br>1 Real-time interrupts enabled. |
| 2:0<br>RTIS[2:0] | **Real-Time Interrupt Delay Selects** — These read/write bits select the wakeup delay for the RTI. The clock source for the real-time interrupt is a self-clocked source which oscillates at about 1 kHz, is independent of other MCU clock sources. Using external clock source the delays will be crystal frequency divided by value in RTIS2:RTIS1:RTIS0. See Table 5-11. |

**Table 5-11. Real-Time Interrupt Frequency**

| RTIS2:RTIS1:RTIS0 | 1-kHz Clock Source Delay[1] | Using External Clock Source Delay (Crystal Frequency) |
|---|---|---|
| 0:0:0 | Disable periodic wakeup timer | Disable periodic wakeup timer |
| 0:0:1 | 8 ms | divide by 256 |
| 0:1:0 | 32 ms | divide by 1024 |
| 0:1:1 | 64 ms | divide by 2048 |
| 1:0:0 | 128 ms | divide by 4096 |
| 1:0:1 | 256 ms | divide by 8192 |
| 1:1:0 | 512 ms | divide by 16384 |
| 1:1:1 | 1.024 s | divide by 32768 |

[1] Normal values are shown in this column based on $f_{RTI}$ = 1 kHz. See Appendix A, "Electrical Characteristics and Timing Specifications," $f_{RTI}$ for the tolerance on these values.

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1[1] | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVDF | 0 | LVDIE | LVDRE[2] | LVDSE[2] | LVDE[2] | | BGBE |
| W | | LVDACK | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

= Unimplemented or Reserved

[1] Bit 1 is a reserved bit that must always be written to 0.

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

**Figure 5-10. System Power Management Status and Control 1 Register (SPMSC1)**

**Table 5-12. SPMSC1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LVDF | **Low-Voltage Detect Flag** — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event. |
| 6<br>LVDACK | **Low-Voltage Detect Acknowledge** — This write-only bit is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return 0. |
| 5<br>LVDIE | **Low-Voltage Detect Interrupt Enable** — This read/write bit enables hardware interrupt requests for LVDF.<br>0 Hardware interrupt disabled (use polling).<br>1 Request a hardware interrupt when LVDF = 1. |
| 4<br>LVDRE | **Low-Voltage Detect Reset Enable** — This read/write bit enables LVDF events to generate a hardware reset (provided LVDE = 1).<br>0 LVDF does not generate hardware resets.<br>1 Force an MCU reset when LVDF = 1. |

**Table 5-12. SPMSC1 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 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 read/write 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** — The BGBE bit is used to enable an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels.<br>0 Bandgap buffer disabled.<br>1 Bandgap buffer enabled. |

## 5.9.9 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.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVWF | 0 | LVDV | LVWV | PPDF | 0 | | PPDC[1] |
| W | | LVWACK | | | | PPDACK | | |
| Power-on reset: | 0[(2)] | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LVD reset: | 0[(2)] | 0 | U | U | 0 | 0 | 0 | 0 |
| Any other reset: | 0[(2)] | 0 | U | U | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved  U = Unaffected by reset

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

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

**Figure 5-11. System Power Management Status and Control 2 Register (SPMSC2)**

**Table 5-13. SPMSC2 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 **not** present.<br>1 Low voltage warning is present or was present. |
| 6<br>LVWACK | **Low-Voltage Warning Acknowledge** — The LVWACK bit is the low-voltage warning acknowledge.<br>Writing a 1 to LVWACK clears LVWF to a 0 if a low voltage warning is not present. |
| 5<br>LVDV | **Low-Voltage Detect Voltage Select** — The LVDV bit selects the LVD trip point voltage ($V_{LVD}$).<br>0 Low trip point selected ($V_{LVD} = V_{LVDL}$).<br>1 High trip point selected ($V_{LVD} = V_{LVDH}$). |
| 4<br>LVWV | **Low-Voltage Warning Voltage Select** — The LVWV bit selects the LVW trip point voltage ($V_{LVW}$).<br>0 Low trip point selected ($V_{LVW} = V_{LVWL}$).<br>1 High trip point selected ($V_{LVW} = V_{LVWH}$). |
| 3<br>PPDF | **Partial Power Down Flag** — The PPDF bit indicates that the MCU has exited the stop2 mode.<br>0 Not stop2 mode recovery.<br>1 Stop2 mode recovery. |
| 2<br>PPDACK | **Partial Power Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF bit. |
| 0<br>PPDC | **Partial Power Down Control** — The write-once PPDC bit controls whether stop2 or stop3 mode is selected.<br>0 Stop3 mode enabled.<br>1 Stop2, partial power down, mode enabled. |

## 5.9.10 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the MC9S08AC16 Series devices.

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

= Unimplemented or Reserved

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

**Figure 5-12. System Options Register 2 (SOPT2)**

**Table 5-14. 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.<br>0 Internal 1-kHz clock is source to COP.<br>1 Bus clock is source to COP. |

# Chapter 6
# Parallel Input/Output

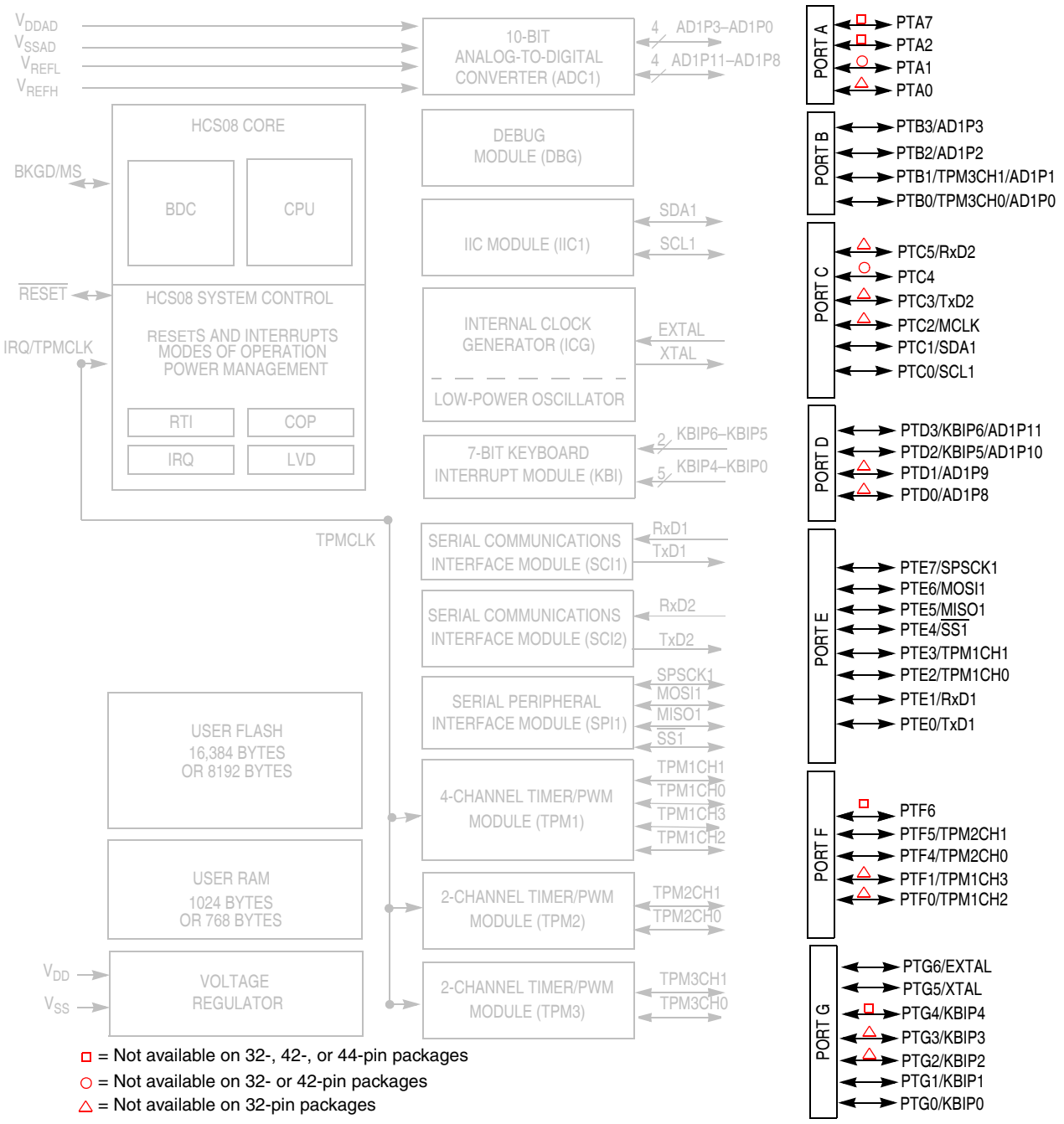
## 6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9S08AC16 has seven I/O ports which include a total of 38 general-purpose I/O pins. See Chapter 2, “Pins and Connections” for more information about the logic and hardware aspects of these pins.

Many of these pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts. When these other modules are not controlling the port pins, they revert to general-purpose I/O control.

**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 should either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.




**Notes**:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to $V_{DD}$. IRQ should not be driven above $V_{DD}$.
4. Pin contains integrated pullup device.
5. PTD3, PTD2, and PTG4 contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 6-1. Block Diagram Highlighting Parallel Input/Output Pins**

## 6.2 Features

Parallel I/O and Pin Control features, depending on package choice, include:

- A total of 38 general-purpose I/O pins in seven ports
- Hysteresis input buffers
- Software-controlled pullups on each input pin
- Software-controlled slew rate output buffers
- Four port A pins
- Four port B pins shared with ADC1 and TPM3
- Six port C pins shared with SCI2, IIC1, and MCLK
- Four port D pins shared with ADC1, KBI, and TPM1 and TPM2 external clock inputs
- Eight port E pins shared with SCI1, TPM1, and SPI1
- Five port F pins shared with TPM1 and TPM2
- Seven port G pins shared with XTAL, EXTAL, and KBI

## 6.3 Pin Descriptions

The MC9S08AC16 Series has a total of 38 parallel I/O pins in seven ports (PTA–PTG). Not all pins are bonded out in all packages. Consult the pin assignment in Chapter 2, “Pins and Connections,” for available parallel I/O pins. All of these pins are available for general-purpose I/O when they are not used by other on-chip peripheral systems.

After reset, the shared peripheral functions are disabled so that the pins are controlled by the parallel I/O. All of the parallel I/O 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 pullups disabled (PTxPEn = 0).

The following paragraphs discuss each port and the software controls that determine each pin’s use.

### 6.3.1 Port A

| Port A | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | MCU Pin: | PTA7 | R | R | R | R | PTA2 | PTA1 | PTA0 |

**Figure 6-2. Port A Pin Names**

Port A pins are general-purpose I/O pins. Parallel I/O function is controlled by the port A data (PTAD) and data direction (PTADD) registers which are located in page zero register space. The pin control registers, pullup enable (PTAPE), slew rate control (PTASE), and drive strength select (PTADS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

## 6.3.2 Port B

| Port B | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MCU Pin: | R | R | R | R | PTB3/ TPM3CH0/ AD1P3 | PTB2/ TPM3CH1/ AD1P2 | PTB1/ AD1P1 | PTB0/ AD1P0 |

**Figure 6-3. Port B Pin Names**

Port B pins are general-purpose I/O pins. Parallel I/O function is controlled by the port B data (PTBD) and data direction (PTBDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTBPE), slew rate control (PTBSE), and drive strength select (PTBDS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

Port B general-purpose I/O are shared with the ADC and TPM3 timer channels. Any pin enabled as an ADC input will have the general-purpose I/O function disabled. When any TPM3 function is enabled, the direction (input or output) is controlled by the TPM3 and not by the data direction register of the parallel I/O port. Refer to Chapter 10, “Timer/PWM (S08TPMV3),” for more information about using port B pins as TPM channels. Refer to Chapter 14, “Analog-to-Digital Converter (S08ADC10V1)” for more information about using port B as analog inputs.

## 6.3.3 Port C

| Port C | Bit 7 | 6 | 5 | 3 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MCU Pin: | 0 | R | PTC5/ RxD2 | PTC4 | PTC3/ TxD2 | PTC2/ MCLK | PTC1/ SDA1 | PTC0/ SCL1 |

**Figure 6-4. Port C Pin Names**

Port C pins are general-purpose I/O pins. Parallel I/O function is controlled by the port C data (PTCD) and data direction (PTCDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTCPE), slew rate control (PTCSE), and drive strength select (PTCDS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

Port C general-purpose I/O is shared with SCI2, IIC, and MCLK. When any shared function is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. Also, for pins which are configured as outputs by the shared function, the output data is controlled by the shared function and not by the port data register.

Refer to Chapter 11, “Serial Communications Interface (S08SCIV4)” for more information about using port C pins as SCI pins.

Refer to Chapter 13, “Inter-Integrated Circuit (S08IICV2)” for more information about using port C pins as IIC pins.

Refer to Chapter 5, “Resets, Interrupts, and System Configuration” for more information about using PTC2 as the MCLK pin.

### 6.3.4 Port D

| Port D | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MCU Pin: | R | R | R | R | PTD3/ AD1P11/ KBIP6 | PTD2/ AD1P10/ KBIP5 | PTD1/ AD1P9 | PTD0/ AD1P8 |

**Figure 6-5. Port D Pin Names**

Port D pins are general-purpose I/O pins. Parallel I/O function is controlled by the port D data (PTDD) and data direction (PTDDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTDPE), slew rate control (PTDSE), and drive strength select (PTDDS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

Port D general-purpose I/O are shared with the ADC and KBI. When any of these shared functions is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. When a pin is shared with both the ADC and a digital peripheral function, the ADC has higher priority. For example, in the case that both the ADC and the KBI are configured to use PTD7 then the pin is controlled by the ADC module.

Refer to Chapter 10, “Timer/PWM (S08TPMV3)” for more information about using port D pins as TPM external clock inputs.

Refer to Chapter 14, “Analog-to-Digital Converter (S08ADC10V1)” for more information about using port D pins as analog inputs.

Refer to Chapter 9, “Keyboard Interrupt (S08KBIV1)” for more information about using port D pins as keyboard inputs.

### 6.3.5 Port E

| Port E | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MCU Pin: | PTE7/ SPSCK1 | PTE6/ MOSI1 | PTE5/ MISO1 | PTE4/ $\overline{SS1}$ | PTE3/ TPM1CH1 | PTE2/ TPM1CH0 | PTE1/ RxD1 | PTE0/ TxD1 |

**Figure 6-6. Port E Pin Names**

Port E pins are general-purpose I/O pins. Parallel I/O function is controlled by the port E data (PTED) and data direction (PTEDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTEPE), slew rate control (PTESE), and drive strength select (PTEDS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

Port E general-purpose I/O is shared with SCI1, SPI, and TPM1 timer channels. When any of these shared functions is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. Also, for pins which are configured as outputs by the shared function, the output data is controlled by the shared function and not by the port data register.

Refer to Chapter 11, “Serial Communications Interface (S08SCIV4)” for more information about using port E pins as SCI pins.

Refer to Chapter 12, “Serial Peripheral Interface (S08SPIV3)” for more information about using port E pins as SPI pins.

Refer to Chapter 10, “Timer/PWM (S08TPMV3)” for more information about using port E pins as TPM channel pins.

## 6.3.6 Port F

| Port F | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MCU Pin: | R | PTF6 | PTF5/ TPM2CH1 | PTF4/ TPM2CH0 | R | R | PTF1/ TPM1CH3 | PTF0/ TPM1CH2 |

**Figure 6-7. Port F Pin Names**

Port F pins are general-purpose I/O pins. Parallel I/O function is controlled by the port F data (PTFD) and data direction (PTFDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTFPE), slew rate control (PTFSE), and drive strength select (PTFDS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

Port F general-purpose I/O is shared with TPM1 and TPM2 timer channels. When any of these shared functions is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. Also, for pins which are configured as outputs by the shared function, the output data is controlled by the shared function and not by the port data register.

Refer to Chapter 10, “Timer/PWM (S08TPMV3)” for more information about using port F pins as TPM channel pins.

## 6.3.7 Port G

| Port G | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MCU Pin: | 0 | PTG6/ EXTAL | PTG5/ XTAL | PTG4/ KBIP4 | PTG3/ KBIP3 | PTG2/ KBIP2 | PTG1/ KBIP1 | PTG0/ KBIP0 |

**Figure 6-8. Port G Pin Names**

Port G pins are general-purpose I/O pins. Parallel I/O function is controlled by the port G data (PTGD) and data direction (PTGDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTGPE), slew rate control (PTGSE), and drive strength select (PTGDS) are located in the high page registers. Refer to Section 6.4, “Parallel I/O Control” for more information about general-purpose I/O control and Section 6.5, “Pin Control” for more information about pin control.

Port G general-purpose I/O is shared with KBI, XTAL, and EXTAL. When a pin is enabled as a KBI input, the pin functions as an input regardless of the state of the associated PTG data direction register bit. When the external oscillator is enabled, PTG5 and PTG6 function as oscillator pins. In this case the associated parallel I/O and pin control registers have no control of the pins.

Refer to Chapter 8, “Internal Clock Generator (S08ICGV4)” for more information about using port G pins as XTAL and EXTAL pins.

Refer to Chapter 9, “Keyboard Interrupt (S08KBIV1)” for more information about using port G pins as keyboard inputs.

## 6.4 Parallel I/O Control

Reading and writing of parallel I/O is done through the port data registers. The direction, 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 below.

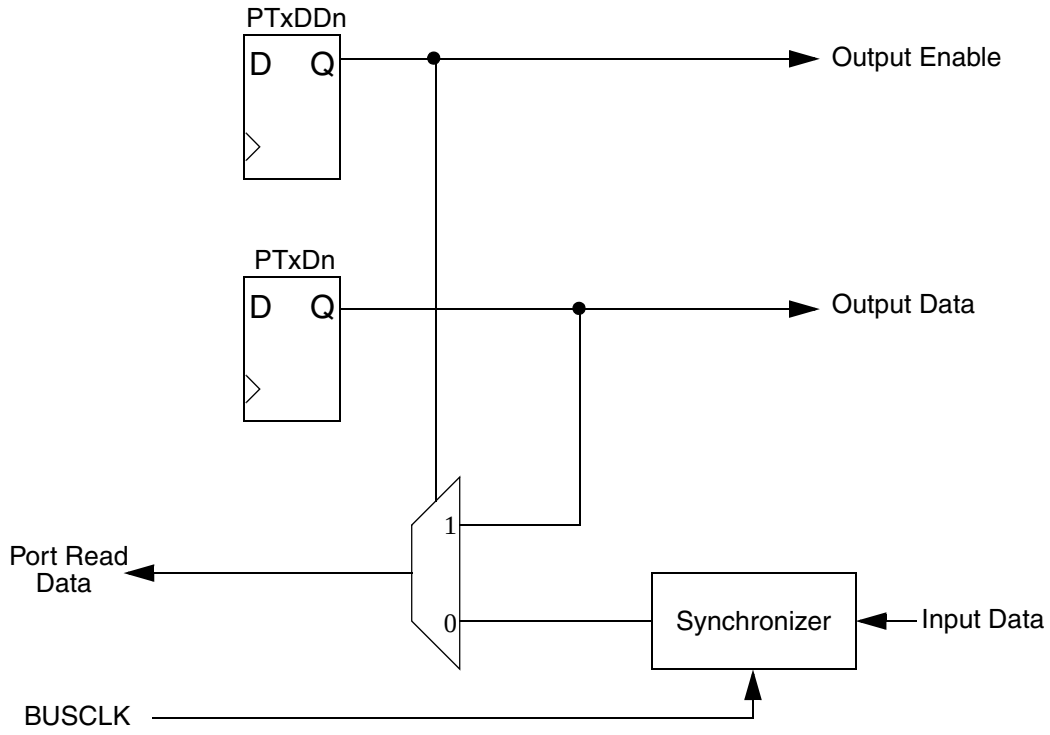



**Figure 6-9. Parallel I/O Block Diagram**

The data direction control bits determine whether the pin output driver is enabled, and they control what is read for port data register reads. Each port pin has a data direction register bit. When PTxDDn = 0, the corresponding pin is an input and reads of PTxD return the pin value. When PTxDDn = 1, the corresponding pin is an output and reads of PTxD return the last value written to the port data register. When a peripheral module or system function is in control of a port pin, the data direction register bit still controls what is returned for reads of the port data register, even though the peripheral system has overriding control of the actual pin direction.

When a shared analog function is enabled for a pin, all digital pin functions are disabled. A read of the port data register returns a value of 0 for any bits which have shared analog functions enabled. 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.

## 6.5 Pin Control

The pin control registers are located in the high page register block of the memory. These registers are used to control pullups, slew rate, and drive strength for the I/O pins. The pin control registers operate independently of the parallel I/O registers.

### 6.5.1 Internal Pullup Enable

An internal pullup device can be enabled for each port pin by setting the corresponding bit in one of the pullup enable registers (PTxPEn). The pullup 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 pullup enable register bit. The pullup device is also disabled if the pin is controlled by an analog function.

### 6.5.2 Output Slew Rate Control Enable

Slew rate control can be enabled for each port pin by setting the corresponding bit in one of the slew rate control registers (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 which are configured as inputs.

### 6.5.3 Output Drive Strength Select

An output pin can be selected to have high output drive strength by setting the corresponding bit in one of the drive strength select registers (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 chip 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.

## 6.6 Pin Behavior in Stop Modes

Depending on the stop mode, I/O functions differently as the result of executing a STOP instruction. An explanation of I/O 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, I/O data previously stored in RAM, before the STOP instruction was executed, peripherals may require being initialized and restored to their pre-stop condition. 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's 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.

## 6.7 Parallel I/O and Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports and pin control functions. These parallel I/O registers are located in page zero of the memory map and the pin control registers are located in the high page register section of memory.

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

### 6.7.1 Port A I/O Registers (PTAD and PTADD)

Port A parallel I/O function is controlled by the registers listed below.

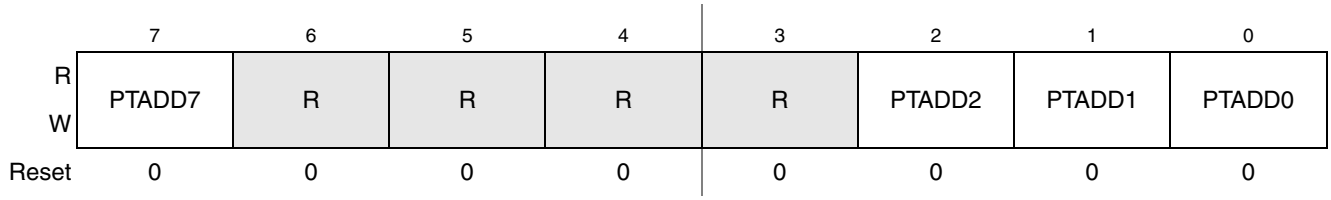
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAD7 | R | R | R | R | PTAD2 | PTAD1 | PTAD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-10. Port A Data Register (PTAD)[1]**

[1] Bits 6 through 3 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 7, 2:0<br>PTADn | **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 pullups disabled. |



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

**Figure 6-11. Data Direction for Port A Register (PTADD)[1]**

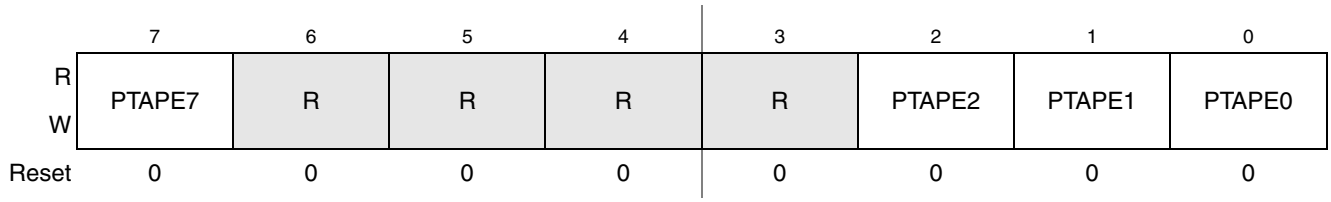
[1] Bits 6 through 3 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 7, 2:0<br>PTADDn | **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. |

## 6.7.2 Port A Pin Control Registers (PTAPE, PTASE, PTADS)

In addition to the I/O control, port A pins are controlled by the registers listed below.



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

**Figure 6-12. Internal Pullup Enable for Port A (PTAPE)[1]**

[1] Bits 6 through 3 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 7, 2:0<br>PTAPEn | **Internal Pullup Enable for Port A Bits** — Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port A bit n.<br>1 Internal pullup device enabled for port A bit n. |

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

**Figure 6-13. Output Slew Rate Control Enable for Port A (PTASE)[1]**

1 Bits 6 through 3 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 7, 2:0<br>PTASEn | **Output Slew Rate Control Enable for Port A Bits** — Each of these control bits determine whether 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. |

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

**Figure 6-14. Output Drive Strength Selection for Port A (PTADS)[1]**

1 Bits 6 through 3 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 7, 2:0<br>PTADSn | **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.<br>0 Low output drive enabled for port A bit n.<br>1 High output drive enabled for port A bit n. |

## 6.7.3 Port B I/O Registers (PTBD and PTBDD)

Port B parallel I/O function is controlled by the registers in this section.

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

**Figure 6-15. Port B Data Register (PTBD)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTBD[3:0] | **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 pullups disabled. |

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

**Figure 6-16. Data Direction for Port B (PTBDD)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTBDD[3:0] | **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. |

## 6.7.4 Port B Pin Control Registers (PTBPE, PTBSE, PTBDS)

In addition to the I/O control, port B pins are controlled by the registers listed below.

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

**Figure 6-17. Internal Pullup Enable for Port B (PTBPE)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTBPE[3:0] | **Internal Pullup Enable for Port B Bits** — Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port B bit n.<br>1 Internal pullup device enabled for port B bit n. |

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

**Figure 6-18. Output Slew Rate Control Enable (PTBSE)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTBSE[3:0] | **Output Slew Rate Control Enable for Port B Bits**— Each of these control bits determine whether 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. |

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

**Figure 6-19. Output Drive Strength Selection for Port B (PTBDS)[1]**

1 Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTBDS[3:0] | **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.<br>0 Low output drive enabled for port B bit n.<br>1 High output drive enabled for port B bit n. |

## 6.7.5 Port C I/O Registers (PTCD and PTCDD)

Port C parallel I/O function is controlled by the registers listed below.

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

**Figure 6-20. Port C Data Register (PTCD)[1]**

1 Bit 6 is a reserved bit that must always be written to 0.

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

| Field | Description |
|---|---|
| 5:0<br>PTCD[5:0] | **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 pullups disabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | R | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-21. Data Direction for Port C (PTCDD)[1]**

1 Bit 6 is a reserved bit that must always be written to 0.

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

| Field | Description |
|---|---|
| 5:0<br>PTCDD[5:0] | **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. |

## 6.7.6 Port C Pin Control Registers (PTCPE, PTCSE, PTCDS)

In addition to the I/O control, port C pins are controlled by the registers listed below.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | R | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-22. Internal Pullup Enable for Port C (PTCPE)[1]**

1 Bit 6 is a reserved bit that must always be written to 0.

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

| Field | Description |
|---|---|
| 5:0<br>PTCPE[5:0] | **Internal Pullup Enable for Port C Bits** — Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port C bit n.<br>1 Internal pullup device enabled for port C bit n. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | R | PTCSE5 | PTCSE4 | PTCSE3 | PTCSE2 | PTCSE1 | PTCSE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-23. Output Slew Rate Control Enable for Port C (PTCSE)[1]**

1 Bit 6 is a reserved bit that must always be written to 0.

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

| Field | Description |
|---|---|
| 5:0<br>PTCSE[5:0] | **Output Slew Rate Control Enable for Port C Bits** — Each of these control bits determine whether 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. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | R | PTCDS5 | PTCDS4 | PTCDS3 | PTCDS2 | PTCDS1 | PTCDS0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-24. Output Drive Strength Selection for Port C (PTCDS)[1]**

1 Bit 6 is a reserved bit that must always be written to 0.

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

| Field | Description |
|---|---|
| 5:0<br>PTCDS[5:0] | **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.<br>0 Low output drive enabled for port C bit n.<br>1 High output drive enabled for port C bit n. |

## 6.7.7 Port D I/O Registers (PTDD and PTDDD)

Port D parallel I/O function is controlled by the registers listed below.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R | R | R | R | PTDD3 | PTDD2 | PTDD1 | PTDD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-25. Port D Data Register (PTDD)[1]**

1 Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTDD[3:0] | **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 pullups disabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R | R | R | R | PTDDD3 | PTDDD2 | PTDDD1 | PTDDD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-26. Data Direction for Port D (PTDDD)[1]**

1 Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTDDD[3:0] | **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. |

## 6.7.8 Port D Pin Control Registers (PTDPE, PTDSE, PTDDS)

In addition to the I/O control, port D pins are controlled by the registers listed below.

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

**Figure 6-27. Internal Pullup Enable for Port D (PTDPE)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTDPE[3:0] | **Internal Pullup Enable for Port D Bits** — Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port D bit n.<br>1 Internal pullup device enabled for port D bit n. |

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

**Figure 6-28. Output Slew Rate Control Enable for Port D (PTDSE)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTDSE[3:0] | **Output Slew Rate Control Enable for Port D Bits** — Each of these control bits determine whether 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. |

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

**Figure 6-29. Output Drive Strength Selection for Port D (PTDDS)[1]**

[1] Bits 7 through 4 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 3:0<br>PTDDS[3:0] | **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.<br>0 Low output drive enabled for port D bit n.<br>1 High output drive enabled for port D bit n. |

## 6.7.9 Port E I/O Registers (PTED and PTEDD)

Port E parallel I/O function is controlled by the registers listed below.

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

**Figure 6-30. Port E Data Register (PTED)**

**Table 6-21. 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 pullups disabled. |

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

**Figure 6-31. Data Direction for Port E (PTEDD)**

**Table 6-22. 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. |

## 6.7.10 Port E Pin Control Registers (PTEPE, PTESE, PTEDS)

In addition to the I/O control, port E pins are controlled by the registers listed below.

| | 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 6-32. Internal Pullup Enable for Port E (PTEPE)**

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

| Field | Description |
|---|---|
| 7:0<br>PTEPE[7:0] | **Internal Pullup Enable for Port E Bits**— Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port E bit n.<br>1 Internal pullup device enabled for port E bit n. |

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

**Figure 6-33. Output Slew Rate Control Enable for Port E (PTESE)**

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

| Field | Description |
|---|---|
| 7:0<br>PTESE[7:0] | **Output Slew Rate Control Enable for Port E Bits** — Each of these control bits determine whether 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. |

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

**Figure 6-34. Output Drive Strength Selection for Port E (PTEDS)**

**Table 6-25. 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.<br>0 Low output drive enabled for port E bit n.<br>1 High output drive enabled for port E bit n. |

## 6.7.11 Port F I/O Registers (PTFD and PTFDD)

Port F parallel I/O function is controlled by the registers listed below.

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

**Figure 6-35. Port F Data Register (PTFD)[1]**

[1] Bits 7, 3 and 2 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 6:4, 1:0<br>PTFDn | **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 pullups disabled. |

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

**Figure 6-36. Data Direction for Port F (PTFDD)[1]**

[1] Bits 7, 3 and 2 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 6:4, 1:0<br>PTFDDn | **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. |

## 6.7.12 Port F Pin Control Registers (PTFPE, PTFSE, PTFDS)

In addition to the I/O control, port F pins are controlled by the registers listed below.

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

**Figure 6-37. Internal Pullup Enable for Port F (PTFPE)[1]**

1 Bits 7, 3 and 2 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 6:4, 1:0<br>PTFPEn | **Internal Pullup Enable for Port F Bits** — Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port F bit n.<br>1 Internal pullup device enabled for port F bit n. |

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

**Figure 6-38. Output Slew Rate Control Enable for Port F (PTFSE)[1]**

1 Bits 7, 3 and 2 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 6:4, 1:0<br>PTFSEn | **Output Slew Rate Control Enable for Port F Bits** — Each of these control bits determine whether 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. |

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

**Figure 6-39. Output Drive Strength Selection for Port F (PTFDS)[1]**

1 Bits 7, 3 and 2 are reserved bits that must always be written to 0.

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

| Field | Description |
|---|---|
| 6:4, 1:0<br>PTFDSn | **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.<br>0 Low output drive enabled for port F bit n.<br>1 High output drive enabled for port F bit n. |

## 6.7.13 Port G I/O Registers (PTGD and PTGDD)

Port G parallel I/O function is controlled by the registers listed below.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | PTGD6 | PTGD5 | PTGD4 | PTGD3 | PTGD2 | PTGD1 | PTGD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-40. Port G Data Register (PTGD)**

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

| Field | Description |
|---|---|
| 6:0<br>PTGD[6:0] | **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 pullups disabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PTGDD6 | PTGDD5 | PTGDD4 | PTGDD3 | PTGDD2 | PTGDD1 | PTGDD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-41. Data Direction for Port G (PTGDD)**

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

| Field | Description |
|---|---|
| 6:0<br>PTGDD[6:0] | **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. |

## 6.7.14 Port G Pin Control Registers (PTGPE, PTGSE, PTGDS)

In addition to the I/O control, port G pins are controlled by the registers listed below.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PTGPE6 | PTGPE5 | PTGPE4 | PTGPE3 | PTGPE2 | PTGPE1 | PTGPE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-42. Internal Pullup Enable for Port G Bits (PTGPE)**

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

| Field | Description |
|---|---|
| 6:0<br>PTGPE[6:0] | **Internal Pullup Enable for Port G Bits** — Each of these control bits determines if the internal pullup 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 pullup devices are disabled.<br>0 Internal pullup device disabled for port G bit n.<br>1 Internal pullup device enabled for port G bit n. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PTGSE6 | PTGSE5 | PTGSE4 | PTGSE3 | PTGSE2 | PTGSE1 | PTGSE0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-43. Output Slew Rate Control Enable for Port G Bits (PTGSE)**

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

| Field | Description |
|---|---|
| 6:0<br>PTGSE[6:0] | **Output Slew Rate Control Enable for Port G Bits**— Each of these control bits determine whether 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. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | PTGDS6 | PTGDS5 | PTGDS4 | PTGDS3 | PTGDS2 | PTGDS1 | PTGDS0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-44. Output Drive Strength Selection for Port G (PTGDS)**

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

| Field | Description |
|---|---|
| 6:0<br>PTGDS[6:0] | **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.<br>0 Low output drive enabled for port G bit n.<br>1 High output drive enabled for port G bit n. |

# Chapter 7
# Central Processor Unit (S08CPUV2)

## 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*.

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.

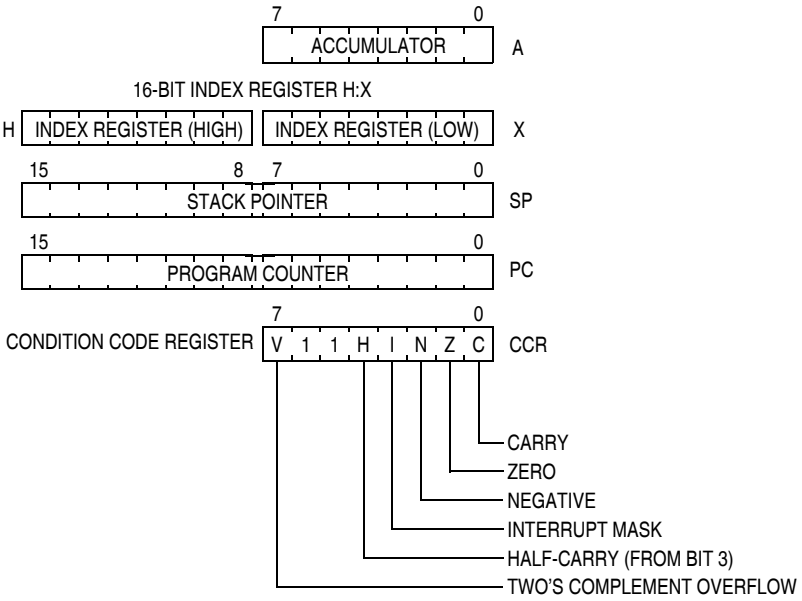



**Figure 7-1. CPU Registers**

### 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*.

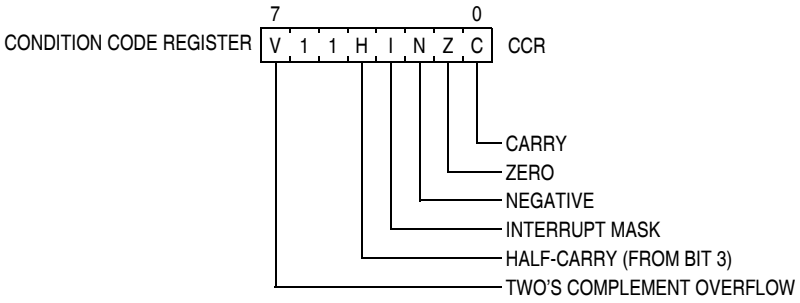



**Figure 7-2. Condition Code Register**

**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.