

MC9S08QA4 MC9S08QA2

Reference Manual

HCS08 Microcontrollers

Related Documentation:

- **MC9S08QA4 (Data Sheet)**
Contains pin assignments and diagrams, all electrical specifications, and mechanical drawing outlines.

Find the most current versions of all documents at:
<http://www.freescale.com>

MC9S08QA4RM
Rev. 2
3/2008

freescale.com

MC9S08QA4 Features

8-Bit HCS08 Central Processor Unit (CPU)

- Up to 20 MHz CPU at 3.6 V to 1.8 V across temperature range of -40°C to 85°C
- HC08 instruction set with added BGND instruction
- Support for up to 32 interrupt/reset sources

On-Chip Memory

- Flash read/program/erase over full operating voltage and temperature
- Random-Access memory (RAM)
- Security circuitry to prevent unauthorized access to RAM and flash contents

Power-Saving Modes

- Two very low power stop modes
- Peripheral clock enable register can disable clocks to unused modules, thereby reducing currents
- Very low power real time counter for use in run, wait, and stop modes with internal clock sources

Clock Source Options

- Internal Clock Source (ICS) — Internal clock source module with a frequency-locked-loop (FLL) controlled by internal reference; precision trimming of internal reference, which allows 0.2% resolution and 2% deviation over temperature and voltage; support of bus frequencies from 1 MHz to 10 MHz

System Protection

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1 kHz internal clock source or bus clock
- Low-Voltage detection with reset or interrupt
- Selectable trip points
- Illegal opcode detection with reset

- Illegal address detection with reset
- Flash block protection

Development Support

- Single-Wire background debug interface
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging

Peripherals

- ADC — 4-channel, 10-bit resolution; $1.7\text{ mV}/^{\circ}\text{C}$ temperature sensor; automatic compare function; internal bandgap reference channel; operation in stop3; full functionality from 3.6 V to 1.8 V.
- ACMP — analog comparator with selectable interrupt on rising, falling, or either edge of comparator output; compare option to fixed internal bandgap reference voltage; output can be tied internally to TPM input capture; operation in stop3
- TPM — One 1-channel timer/pulse-width modulator (TPM) module; selectable input capture, output compare, or buffered edge- or center-aligned PWM on each channel; ACMP output which can be tied internally to input capture
- MTIM — 8-bit modulo timer module with 8-bit prescaler
- KBI — 4-pin keyboard interrupt module with software selectable polarity on edge or edge/level modes

Input/Output

- Four GPIOs, one input-only pin and one output-only pin
- Hysteresis and configurable pullup device on all input pins; configurable slew rate and drive strength on all output pins except PTA5

Package Options

- 8-pin SOIC, 8-pin DIP, 8-pin DFN

MC9S08QA4 MCU Series Reference Manual

Covers: MC9S08QA4
MC9S08QA2

MC9S08QA4
Rev. 2
3/2008

Revision History

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

<http://freescale.com>

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

Revision Number	Revision Date	Description of Changes
1	1/2008	Initial public release
2	3/2008	Changed Chapter 11, "Internal Clock Source (S08ICSV1)" to use the ICSV1.

List of Chapters

Chapter Number	Title	Page
Chapter 1	Device Overview	17
Chapter 2	External Signal Description	21
Chapter 3	Modes of Operation	27
Chapter 4	Memory Map and Register Definition.	33
Chapter 5	Resets, Interrupts, and General System Control	51
Chapter 6	Parallel Input/Output Control	69
Chapter 7	Central Processor Unit (S08CPUV2)	75
Chapter 8	Keyboard Interrupt (S08KBIV2)	95
Chapter 9	Analog Comparator (S08ACMPV2).	103
Chapter 10	Analog-to-Digital Converter (S08ADC10V1)	111
Chapter 11	Internal Clock Source (S08ICSV1)	139
Chapter 12	Modulo Timer (S08MTIMV1)	151
Chapter 13	Timer/Pulse-Width Modulator (S08TPMV2).	161
Chapter 14	Development Support	177

Contents

Section Number	Title	Page
Chapter 1		
Device Overview		
1.1	Introduction	17
1.1.1	Devices in the MC9S08QA4 Series	17
1.1.2	MCU Block Diagram	18
1.2	System Clock Distribution	19
Chapter 2		
External Signal Description		
2.1	Device Pin Assignment	21
2.2	Recommended System Connections	21
2.2.1	Power	22
2.2.2	Reset (Input Only)	23
2.2.3	Background / Mode Select (BKGD/MS)	23
2.2.4	General-Purpose I/O and Peripheral Ports	24
Chapter 3		
Modes of Operation		
3.1	Introduction	27
3.2	Features	27
3.3	Run Mode	27
3.4	Active Background Mode	27
3.5	Wait Mode	28
3.6	Stop Modes	29
3.6.1	Stop3 Mode	29
3.6.2	Stop2 Mode	30
3.6.3	Stop1 Mode	31
3.6.4	On-Chip Peripheral Modules in Stop Modes	32
Chapter 4		
Memory Map and Register Definition		
4.1	MC9S08QA4 Series Memory Map	33
4.2	Reset and Interrupt Vector Assignments	34
4.3	Register Addresses and Bit Assignments	35
4.4	RAM	38
4.5	Flash	39
4.5.1	Features	39
4.5.2	Program and Erase Times	39
4.5.3	Program and Erase Command Execution	40

4.5.4	Burst Program Execution	41
4.5.5	Access Errors	43
4.5.6	Flash Block Protection	43
4.5.7	Vector Redirection	44
4.6	Security	44
4.7	Flash Registers and Control Bits	45
4.7.1	Flash Clock Divider Register (FCDIV)	46
4.7.2	Flash Options Register (FOPT and NVOPT)	47
4.7.3	Flash Configuration Register (FCNFG)	48
4.7.4	Flash Protection Register (FPROT and NVPROT)	48
4.7.5	Flash Status Register (FSTAT)	49
4.7.6	Flash Command Register (FCMD)	50

Chapter 5

Resets, Interrupts, and General System Control

5.1	Introduction	51
5.2	Features	51
5.3	MCU Reset	51
5.4	Computer Operating Properly (COP) Watchdog	52
5.5	Interrupts	53
5.5.1	Interrupt Stack Frame	54
5.5.2	External Interrupt Request Pin ($\overline{\text{IRQ}}$)	54
5.5.3	Interrupt Vectors, Sources, and Local Masks	55
5.6	Low-Voltage Detect (LVD) System	56
5.6.1	Power-On Reset Operation	57
5.6.2	LVD Reset Operation	57
5.6.3	LVD Interrupt Operation	57
5.6.4	Low-Voltage Warning (LVW)	57
5.7	Real-Time Interrupt (RTI)	57
5.8	Reset, Interrupt, and System Control Registers and Control Bits	58
5.8.1	Interrupt Pin Request Status and Control Register (IRQSC)	59
5.8.2	System Reset Status Register (SRS)	60
5.8.3	System Background Debug Force Reset Register (SBDFR)	61
5.8.4	System Options Register 1 (SOPT1)	62
5.8.5	System Options Register 2 (SOPT2)	63
5.8.6	System Device Identification Register (SDIDH, SDIDL)	64
5.8.7	System Real-Time Interrupt Status and Control Register (SRTISC)	65
5.8.8	System Power Management Status and Control 1 Register (SPMSC1)	66
5.8.9	System Power Management Status and Control 2 Register (SPMSC2)	67
5.8.10	System Power Management Status and Control 3 Register (SPMSC3)	68

Chapter 6

Parallel Input/Output Control

6.1	Port Data and Data Direction	69
6.2	Pin Control — Pullup, Slew Rate, and Drive Strength	70

6.3	Pin Behavior in Stop Modes	70
6.4	Parallel I/O Registers	71
6.4.1	Port A Registers	71
6.4.2	Port A Control Registers	72

Chapter 7 Central Processor Unit (S08CPUV2)

7.1	Introduction	75
7.1.1	Features	75
7.2	Programmer's Model and CPU Registers	76
7.2.1	Accumulator (A)	76
7.2.2	Index Register (H:X)	76
7.2.3	Stack Pointer (SP)	77
7.2.4	Program Counter (PC)	77
7.2.5	Condition Code Register (CCR)	77
7.3	Addressing Modes	79
7.3.1	Inherent Addressing Mode (INH)	79
7.3.2	Relative Addressing Mode (REL)	79
7.3.3	Immediate Addressing Mode (IMM)	79
7.3.4	Direct Addressing Mode (DIR)	79
7.3.5	Extended Addressing Mode (EXT)	80
7.3.6	Indexed Addressing Mode	80
7.4	Special Operations	81
7.4.1	Reset Sequence	81
7.4.2	Interrupt Sequence	81
7.4.3	Wait Mode Operation	82
7.4.4	Stop Mode Operation	82
7.4.5	BGND Instruction	83
7.5	HCS08 Instruction Set Summary	84

Chapter 8 Keyboard Interrupt (S08KBIV2)

8.1	Introduction	95
8.1.1	Features	97
8.1.2	Modes of Operation	97
8.1.3	Block Diagram	97
8.2	External Signal Description	98
8.3	Register Definition	98
8.3.1	KBI Status and Control Register (KBISC)	98
8.3.2	KBI Pin Enable Register (KBIPE)	99
8.3.3	KBI Edge Select Register (KBIES)	99
8.4	Functional Description	100
8.4.1	Edge Only Sensitivity	100
8.4.2	Edge and Level Sensitivity	100
8.4.3	KBI Pullup/Pulldown Resistors	101

8.4.4	KBI Initialization	101
-------	--------------------------	-----

Chapter 9

Analog Comparator (S08ACMPV2)

9.1	Introduction	103
9.1.1	ACMP Configuration Information	103
9.1.2	ACMP/TPM Configuration Information	103
9.1.3	Features	105
9.1.4	Modes of Operation	105
9.1.5	Block Diagram	105
9.2	External Signal Description	107
9.3	Register Definition	107
9.3.1	ACMP Status and Control Register (ACMPSC)	108
9.4	Functional Description	109

Chapter 10

Analog-to-Digital Converter (S08ADC10V1)

10.1	Introduction	111
10.1.1	Module Configurations	112
10.1.2	Features	115
10.1.3	Block Diagram	115
10.2	External Signal Description	116
10.2.1	Analog Power (V_{DDAD})	117
10.2.2	Analog Ground (V_{SSAD})	117
10.2.3	Voltage Reference High (V_{REFH})	117
10.2.4	Voltage Reference Low (V_{REFL})	117
10.2.5	Analog Channel Inputs (ADx)	117
10.3	Register Definition	117
10.3.1	Status and Control Register 1 (ADCSC1)	117
10.3.2	Status and Control Register 2 (ADCSC2)	119
10.3.3	Data Result High Register (ADCRH)	120
10.3.4	Data Result Low Register (ADCRL)	120
10.3.5	Compare Value High Register (ADCCVH)	121
10.3.6	Compare Value Low Register (ADCCVL)	121
10.3.7	Configuration Register (ADCCFG)	121
10.3.8	Pin Control 1 Register (APCTL1)	123
10.3.9	Pin Control 2 Register (APCTL2)	124
10.3.10	Pin Control 3 Register (APCTL3)	125
10.4	Functional Description	126
10.4.1	Clock Select and Divide Control	126
10.4.2	Input Select and Pin Control	127
10.4.3	Hardware Trigger	127
10.4.4	Conversion Control	127
10.4.5	Automatic Compare Function	130
10.4.6	MCU Wait Mode Operation	130

10.4.7	MCU Stop3 Mode Operation	130
10.4.8	MCU Stop1 and Stop2 Mode Operation	131
10.5	Initialization Information	131
10.5.1	ADC Module Initialization Example	131
10.6	Application Information	133
10.6.1	External Pins and Routing	133
10.6.2	Sources of Error	135

Chapter 11 Internal Clock Source (S08ICSV1)

11.1	Introduction	139
11.1.1	Module Configuration	139
11.1.2	Factory Trim Value	139
11.1.3	Features	141
11.1.4	Modes of Operation	141
11.1.5	Block Diagram	142
11.2	External Signal Description	143
11.3	Register Definition	143
11.3.1	ICS Control Register 1 (ICSC1)	143
11.3.2	ICS Control Register 2 (ICSC2)	144
11.3.3	ICS Trim Register (ICSTRM)	145
11.3.4	ICS Status and Control (ICSSC)	145
11.4	Functional Description	146
11.4.1	Operational Modes	146
11.4.2	Mode Switching	148
11.4.3	Bus Frequency Divider	148
11.4.4	Low Power Bit Usage	149
11.4.5	Internal Reference Clock	149
11.4.6	Optional External Reference Clock	149
11.4.7	Fixed Frequency Clock	149

Chapter 12 Modulo Timer (S08MTIMV1)

12.1	Introduction	151
12.1.1	MTIM/TPM Configuration Information	151
12.1.2	Features	153
12.1.3	Modes of Operation	153
12.1.4	Block Diagram	154
12.2	External Signal Description	154
12.3	Register Definition	154
12.3.1	MTIM1 Status and Control Register (MTIM1SC)	156
12.3.2	MTIM1 Clock Configuration Register (MTIM1CLK)	157
12.3.3	MTIM1 Counter Register (MTIM1CNT)	158
12.3.4	MTIM1 Modulo Register (MTIM1MOD)	158
12.4	Functional Description	159

12.4.1 MTIM Operation Example	160
-------------------------------------	-----

Chapter 13 Timer/Pulse-Width Modulator (S08TPMV2)

13.1 Introduction	161
13.1.1 ACMP/TPM Configuration Information	161
13.1.2 MTIM/TPM Configuration Information	161
13.1.3 Features	163
13.1.4 Block Diagram	163
13.2 External Signal Description	165
13.2.1 External TPM Clock Sources	165
13.2.2 TPMxCHn — TPMx Channel n I/O Pins	165
13.3 Register Definition	165
13.3.1 Timer Status and Control Register (TPMxSC)	166
13.3.2 Timer Counter Registers (TPMxCNTH:TPMxCNTL)	167
13.3.3 Timer Counter Modulo Registers (TPMxMODH:TPMxMODL)	168
13.3.4 Timer Channel n Status and Control Register (TPMxCnSC)	169
13.3.5 Timer Channel Value Registers (TPMxCnVH:TPMxCnVL)	170
13.4 Functional Description	171
13.4.1 Counter	171
13.4.2 Channel Mode Selection	172
13.4.3 Center-Aligned PWM Mode	174
13.5 TPM Interrupts	175
13.5.1 Clearing Timer Interrupt Flags	175
13.5.2 Timer Overflow Interrupt Description	175
13.5.3 Channel Event Interrupt Description	176
13.5.4 PWM End-of-Duty-Cycle Events	176

Chapter 14 Development Support

14.1 Introduction	177
14.1.1 Module Configuration	177
14.1.2 Features	178
14.2 Background Debug Controller (BDC)	178
14.2.1 BKGD Pin Description	179
14.2.2 Communication Details	180
14.2.3 BDC Commands	182
14.2.4 BDC Hardware Breakpoint	185
14.3 On-Chip Debug System (DBG)	186
14.3.1 Comparators A and B	186
14.3.2 Bus Capture Information and FIFO Operation	186
14.3.3 Change-of-Flow Information	187
14.3.4 Tag vs. Force Breakpoints and Triggers	187
14.3.5 Trigger Modes	188
14.3.6 Hardware Breakpoints	190

14.4	Register Definition	190
14.4.1	BDC Registers and Control Bits	190
14.4.2	System Background Debug Force Reset Register (SBD FR)	192
14.4.3	DBG Registers and Control Bits	193

Chapter 1

Device Overview

1.1 Introduction

The MC9S08QA4 series 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 features associated with each device in this series.

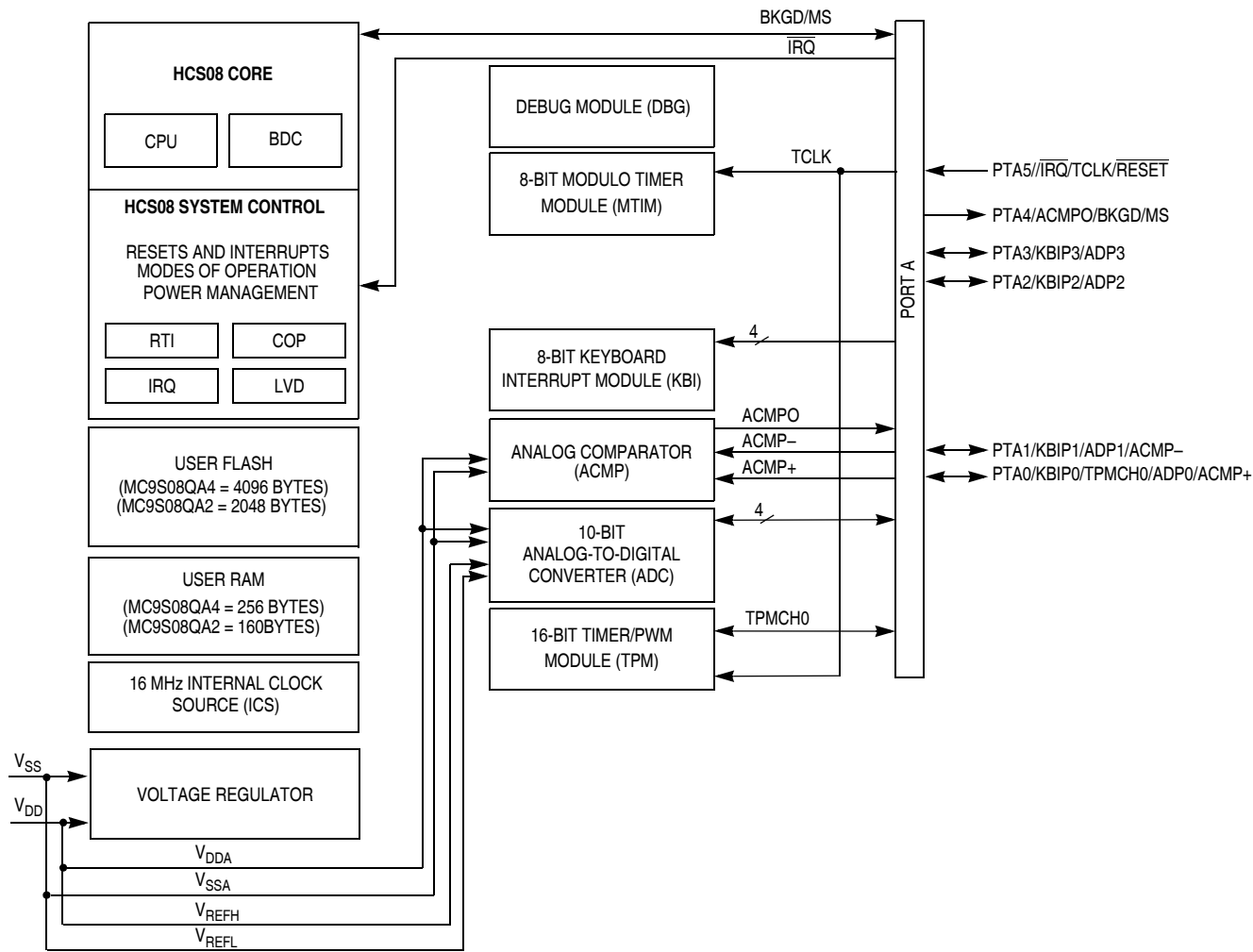
1.1.1 Devices in the MC9S08QA4 Series

[Table 1-1](#) summarizes the features available in the MC9S08QA4 series of MCUs.

Table 1-1. Devices in the MC9S08QA4 Series

Feature	Device	
	MC9S08QA4	MC9S08QA2
Package	8-Pin	8-Pin
Flash	4K	2K
RAM	256	160
ICS	yes	
ACMP	yes	
ADC	4-ch	
DBG	yes	
IRQ	yes	
KBI	4-pin	
MTIM	yes	
TPM	1-ch	
I/O pins	4 I/O 1 output only 1 input only	
Package Types	8 DFN 8 SOIC 8 PDIP	

1.1.2 MCU Block Diagram



NOTES:

- 1 Port pins are software configurable with pullup device if input port.
- 2 Port pins are software configurable for output drive strength.
- 3 Port pins are software configurable for output slew rate control.
- 4 $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- 5 RESET contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- 6 PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
- 7 When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 1-1. MC9S08QA4 Series Block Diagram

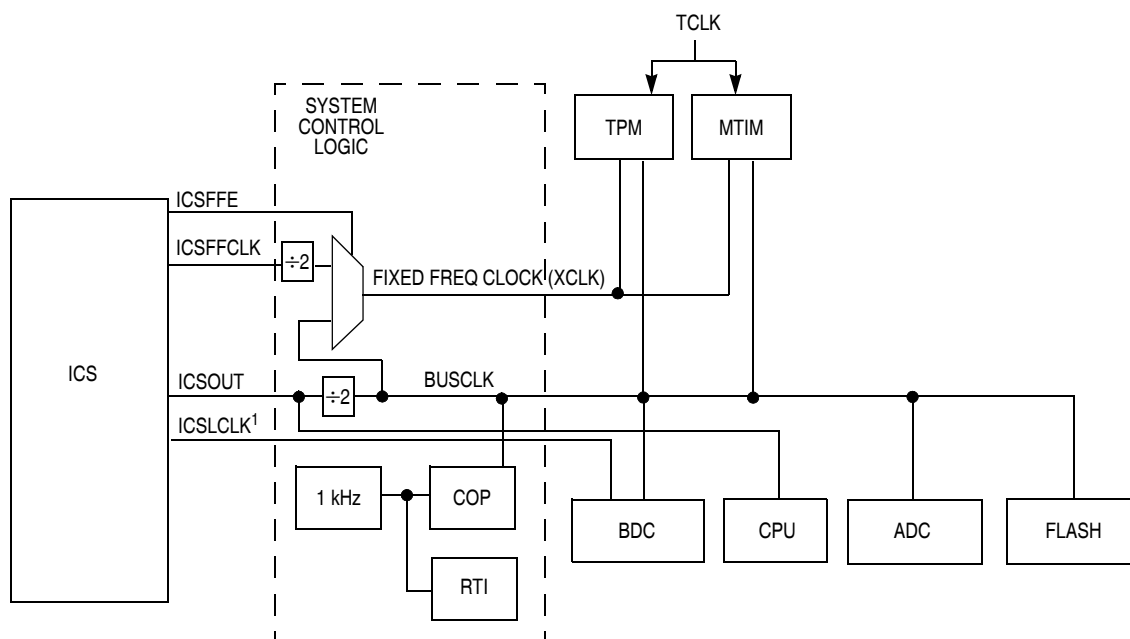
Table 1-2 provides the functional versions of the on-chip modules.

Table 1-2. Versions of On-Chip Modules

Module	Version
Analog Comparator (ACMP)	2
Analog-to-Digital Converter (ADC)	1
Central Processing Unit (CPU)	2
Internal Clock Source (ICS)	1
Keyboard Interrupt (KBI)	2
Modulo Timer (MTIM)	1
Timer Pulse-Width Modulator (TPM)	2
Debug Module (DBG)	2

1.2 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs as shown. The clock inputs to the modules indicate which clock(s) are used to drive the module function. All memory mapped registers associated with the modules are clocked with BUSCLK.



¹ ICSLCLK is the alternate BDC clock source for the MC9S08QA4 series.

² ADC has min and max frequency requirements. See the ADC chapter and *MC9S08QA4 Series Data Sheet*.

³ Flash has frequency requirements for program and erase operation. See *MC9S08QA4 Series Data Sheet*.

Figure 1-2. System Clock Distribution Diagram



Chapter 2

External Signal Description

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

2.1 Device Pin Assignment

Figure 2-1 shows the pin assignments for the available packages. Table 1-1 lists package types available for each device in the series.

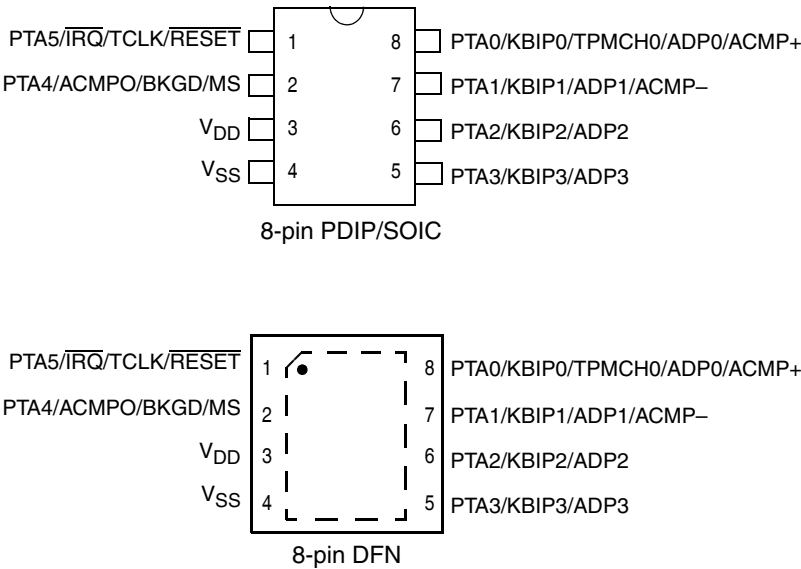
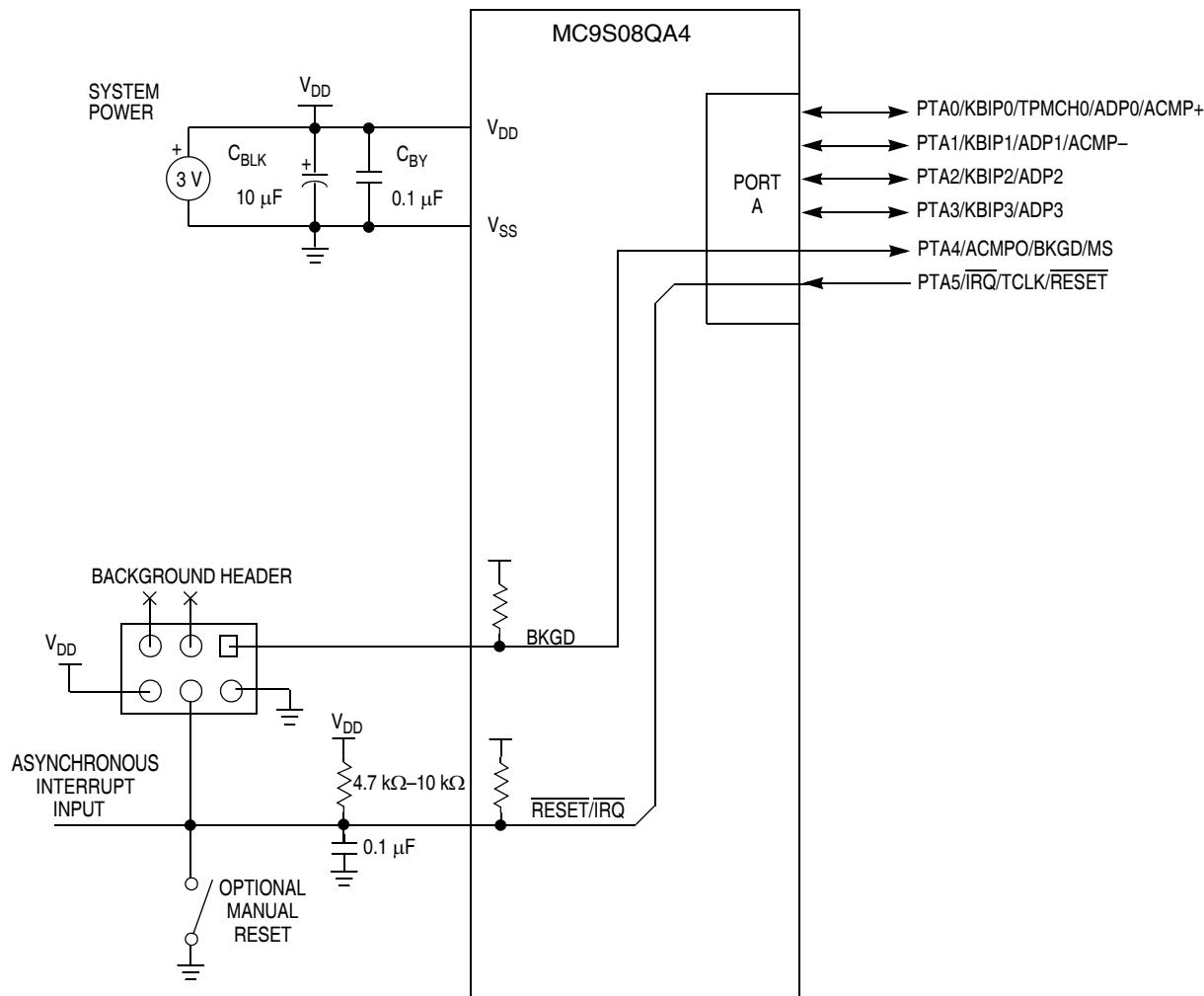


Figure 2-1. 8-Pin Packages

2.2 Recommended System Connections

Figure 2-2 shows pin connections that are common to almost all MC9S08QA4 series application systems.



NOTES:

1. The $\overline{\text{RESET}}$ pin can only be used to reset into user mode; you cannot enter BDM using the $\overline{\text{RESET}}$ pin. BDM can be entered by holding MS low during POR or writing a 1 to BDFR in SBDFFR with MS low after issuing the BDM command.
2. IRQ feature has optional internal pullup device.
3. RC filter on $\overline{\text{RESET/IRQ}}$ pin recommended for noisy environments.

Figure 2-2. Basic System Connections

2.2.1 Power

V_{DD} and V_{SS} are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry, ACMP and ADC modules, and to an internal voltage regulator. The internal voltage regulator provides a 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: a bulk electrolytic capacitor, such as a 10- μF tantalum capacitor, to provide bulk charge storage for the overall system, and a bypass capacitor, such as a 0.1- μF ceramic capacitor, located as near to the MCU power pins as practical to suppress high-frequency noise.

2.2.2 $\overline{\text{Reset}}$ (Input Only)

After a power-on reset (POR), the PTA5/ $\overline{\text{IRQ}}$ / $\overline{\text{TCLK}}$ / $\overline{\text{RESET}}$ pin defaults to a general-purpose input port pin, PTA5. Setting RSTPE in SOPT1 configures the pin to be the $\overline{\text{RESET}}$ input pin. After configured as $\overline{\text{RESET}}$, the pin will remain $\overline{\text{RESET}}$ until the next POR. The $\overline{\text{RESET}}$ pin can be used to reset the MCU from an external source when the pin is driven low. When enabled as the $\overline{\text{RESET}}$ pin (RSTPE = 1), an internal pullup device is automatically enabled.

NOTE

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD} .

The voltage measured on the internally pulled up $\overline{\text{RESET}}$ pin will not be pulled to V_{DD} . The internal gates connected to this pin are pulled to V_{DD} . The $\overline{\text{RESET}}$ pullup must not be used to pull up components external to the MCU.

NOTE

In EMC-sensitive applications, an external RC filter is recommended on the $\overline{\text{RESET}}$ pin, if enabled. See [Figure 2-2](#) for an example.

2.2.3 Background / Mode Select (BKGD/MS)

During a power-on-reset (POR) or background debug force reset (see [5.8.3, “System Background Debug Force Reset Register \(SBD FR\)”](#) for more information), the PTA4/ACMPO/BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin (BKGDPE = 1), an internal pullup device is automatically enabled.

The background debug communication function is enabled when BKGDPE in SOPT1 is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTA4/ACMPO/BKGD/MS pin's alternative pin functions.

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of the internal reset after a POR or force BDC reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during a POR or immediately after issuing a background debug force reset, which will force 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 maximum bus clock rate, so there must 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.

2.2.4 General-Purpose I/O and Peripheral Ports

Members of the MC9S08QA4 series of MCUs support up to four general-purpose I/O pins, one input-only pin, and one output-only pin, which are shared with on-chip peripheral functions (timers, serial I/O, ADC, keyboard interrupts, etc.). On each MC9S08QA4 series device, there is one input-only and one output-only port pin.

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pullup device.

For information about controlling these pins as general-purpose I/O pins, see [Chapter 6, “Parallel Input/Output Control.”](#) For information about how and when on-chip peripheral systems use these pins, see the appropriate chapter referenced in [Table 2-2](#).

Immediately after reset, all pins that are not output-only are configured as high-impedance general-purpose inputs with internal pullup devices disabled. After reset, the output-only port function is not enabled but is configured for low output drive strength with slew rate control enabled. The PTA4 pin defaults to BKGD/MS on any reset.

NOTE

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

2.2.4.1 Pin Control Registers

To select drive strength or to enable slew rate control or pullup devices, the user writes to the appropriate pin control register located in the high page register block of the memory map. The pin control registers operate independently of the parallel I/O registers and allow control of a port on an individual pin basis.

2.2.4.1.1 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 by 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.

The KBI module, when enabled for rising edge detection, causes an enabled internal pull device to be configured as a pulldown.

2.2.4.2 Output Slew Rate Control

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 that are configured as inputs.

2.2.4.3 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 if pins are enabled as high drive.

Table 2-1. Pin Sharing Priority

PIN	Priority				
	← Lowest Highest →				
8-pin	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4
1	PTA5 ¹	$\overline{\text{IRQ}}$	TCLK		$\overline{\text{RESET}}$
2	PTA4		ACMPO	BKGD	MS
3					V _{DD}
4					V _{SS}
5	PTA3	KBIP3	ADP3		
6	PTA2	KBIP2	ADP2		
7	PTA1	KBIP1		ADP1 ²	ACMP ₋ ²
8	PTA0	KBIP0	TPMCH0	ADP0 ²	ACMP ₊ ²

¹ Pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD}. The voltage measured on the internally pulled-up $\overline{\text{RESET}}$ pin will not be pulled to V_{DD}. The internal gates connected to this pin are pulled to V_{DD}.

² If ACMP and ADC are both enabled, both will have access to the pin.

Table 2-2. Pin Function Reference

Signal Function	Example(s)	Reference
Port Pins	PTAx	Chapter 6, “Parallel Input/Output Control”
Analog comparator	ACMPO, ACMP ₋ , ACMP ₊	Chapter 9, “Analog Comparator (S08ACMPV2)”
Keyboard interrupts	KBIPx	Chapter 8, “Keyboard Interrupt (S08KBIV2)”
Timer/PWM	TCLK, TPMCHx	Chapter 13, “Timer/Pulse-Width Modulator (S08TPMV2)”
Analog-to-digital	ADPx	Chapter 10, “Analog-to-Digital Converter (S08ADC10V1)”
Power/core	BKGD/MS, V _{DD} , V _{SS}	Chapter 2, “External Signal Description”
Reset and interrupts	$\overline{\text{RESET}}$, $\overline{\text{IRQ}}$	Chapter 5, “Resets, Interrupts, and General System Control”

NOTE

When an alternative function is first enabled, it is possible to get a spurious edge to the module. User software must clear out any associated flags before interrupts are enabled. [Table 2-1](#) shows 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 MC9S08QA4 series are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each mode are described.

3.2 Features

- Active background mode for code development
- Wait mode:
 - CPU halts operation to conserve power
 - System clocks continue to run
 - Full voltage regulation is maintained
- Stop modes: CPU and bus clocks stopped
 - Stop1: Full powerdown of internal circuits for maximum power savings
 - Stop2: Partial powerdown of internal circuits; RAM contents retained
 - Stop3: All internal circuits powered for fast recovery; RAM and register contents are retained

3.3 Run Mode

Run is the normal operating mode for the MC9S08QA4 series. This mode is selected upon the MCU exiting reset if the BKGD/MS pin is high. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE: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), provides the means for analyzing MCU operation during software development.

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low during POR or immediately after issuing a background debug force reset (see [5.8.3, “System Background Debug Force Reset Register \(SBD FR\)”](#))
- When a BACKGROUND command is received 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 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 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 MC9S08QA4 series MCUs are shipped from the Freescale 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 14, “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 the condition code register (CCR) is cleared when the CPU enters wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, background debug commands can be used on restrictions.

Only the BACKGROUND command and memory-access-with-status commands are available while 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 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 three stop modes is entered upon execution of a STOP instruction when STOPE in SOPT1 is set. In any stop mode, the bus and CPU clocks are halted. The ICS module can be configured to leave the reference clocks running. See [Chapter 11, “Internal Clock Source \(S08ICSV1\)”](#), for more information.

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

Table 3-1. Stop Mode Selection

STOPE	ENBDM ¹	LVDE	LVDSE	PDC	PPDC	Stop Mode
0	x	x		x	x	Stop modes disabled; illegal opcode reset if STOP instruction executed
1	1	x		x	x	Stop3 with BDM enabled ²
1	0	Both bits must be 1		x	x	Stop3 with voltage regulator active
1	0	Either bit a 0		0	x	Stop3
1	0	Either bit a 0		1	1	Stop2
1	0	Either bit a 0		1	0	Stop1

¹ ENBDM is located in the BDCSCR which is only accessible through BDC commands; see [Section 14.4.1.1, “BDC Status and Control Register \(BDCSCR\)”](#).

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

NOTE

The PORT register must be initialized as 0xFF in order to ensure that the low power specifications will be met. User software must write 0xFF to location 0x03.

3.6.1 Stop3 Mode

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

Stop3 can be exited by asserting $\overline{\text{RESET}}$, or by an interrupt from one of the following sources: the real-time interrupt (RTI), LVD, ADC, $\overline{\text{IRQ}}$, or the KBI.

If stop3 is exited by means of the $\overline{\text{RESET}}$ pin, then the MCU is reset and operation will resume after taking the reset vector. Exit by means of one of the internal interrupt sources will result in the MCU taking the appropriate interrupt vector.

NOTE

The PORT register must be initialized as 0xFF in order to ensure that the low power specifications will be met. User software must write 0xFF to location 0x03.

3.6.1.1 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 (LVDE and LVDSE bits in SPMSC1 both set) at the time the CPU executes a STOP instruction, then the voltage regulator remains active during stop mode.

For the ADC to operate, the LVD must remain enabled when entering stop3.

3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in [Chapter 14, “Development Support.”](#) If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation.

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

3.6.2 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions shown in [Table 3-1](#). Most of the internal circuitry of the MCU is powered off in stop2 as in stop1, with the exception of the RAM. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting the wake-up pin (PTA5) on the MCU.

NOTE

PTA5/ $\overline{\text{IRQ}}$ / $\overline{\text{TCLK}}$ / $\overline{\text{RESET}}$ always functions as an active-low wakeup input when the MCU is in stop2, regardless of how the pin is configured before entering stop2. The pullup is not automatically enabled. To use the internal pullup, set the PTAPE5 bit in the PTAPE register.

NOTE

The PORT register must be initialized as 0xFF in order to ensure that the low power specifications will be met. User software must write 0xFF to location 0x03.

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

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

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

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

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

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

3.6.3 Stop1 Mode

Stop1 mode is entered by executing a STOP instruction under the conditions shown in [Table 3-1](#). Most of the internal circuitry of the MCU is powered off in stop1, providing the lowest possible standby current. Upon entering stop1, all I/O pins automatically transition to their default reset states.

Exit from stop1 is performed by asserting the wake-up pin (PTA5) on the MCU.

NOTE

PTA5/ $\overline{\text{IRQ}}$ / $\overline{\text{TCLK}}$ / $\overline{\text{RESET}}$ always functions as an active-low wakeup input when the MCU is in stop1, regardless of how the pin is configured before entering stop1. The pullup is not automatically enabled. To use the internal pullup, set the PTAPE5 bit in the PTAPE register.

NOTE

The PORT register must be initialized as 0xFF in order to ensure that the low power specifications will be met. User software must write 0xFF to location 0x03.

In addition, the real-time interrupt (RTI) can wake the MCU from stop1 if enabled.

Upon wake-up from stop1 mode, the MCU starts up as from a power-on reset (POR):

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

In addition to the above, upon waking up from stop1, the PDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop1 recovery routine. PDF remains set until a 1 is written to PPDACK in SPMSC2.

3.6.4 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to [Section 3.6.3, “Stop1 Mode,”](#) [Section 3.6.2, “Stop2 Mode,”](#) and [Section 3.6.1, “Stop3 Mode,”](#) for specific information on system behavior in stop modes.

Table 3-2. Stop Mode Behavior

Peripheral	Mode		
	Stop1	Stop2	Stop3
CPU	Off	Off	Standby
RAM	Off	Standby	Standby
Flash	Off	Off	Standby
Parallel Port Registers	Off	Off	Standby
ADC	Off	Off	Optionally On ¹
ACMP	Off	Off	Standby
ICS	Off	Off	Optionally On ²
MTIM	Off	Off	Standby
TPM	Off	Off	Standby
Voltage Regulator	Off	Standby	Standby
I/O Pins	Hi-Z	States Held	States Held

¹ Requires the asynchronous ADC clock and LVD to be enabled, else in standby.

² IRCLKEN and IREFSTEN set in ICSC1, else in standby.

Chapter 4

Memory Map and Register Definition

4.1 MC9S08QA4 Series Memory Map

As shown in [Figure 4-1](#), on-chip memory in the MC9S08QA4 series of MCUs consists of RAM, flash program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into these groups:

- Direct-page registers (0x0000 through 0x005F)
- High-page registers (0x1800 through 0x184F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

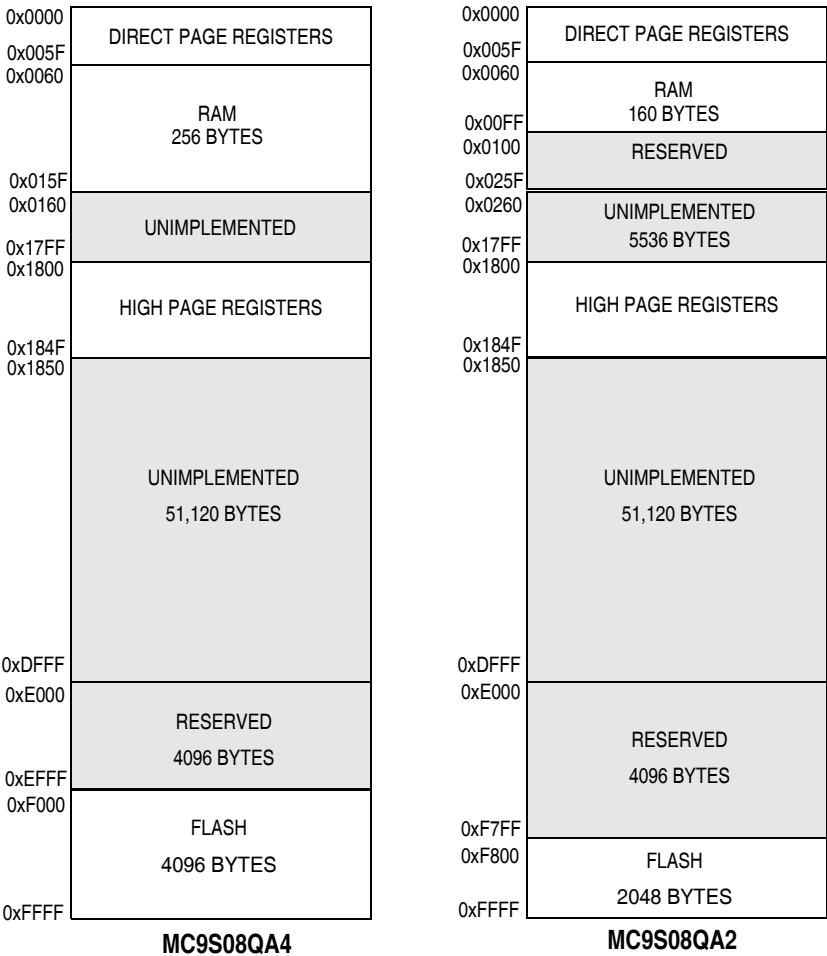


Figure 4-1. MC9S08QA4 Series Memory Map

4.2 Reset and Interrupt Vector Assignments

Table 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale Semiconductor-provided equate file for the MC9S08QA4 series.

Table 4-1. Reset and Interrupt Vectors

Address (High:Low)	Vector	Vector Name
0xFFC0:FFC1 ↑ 0xFFCE:FFCF	Unused Vector Space (available for user program)	
0xFFD0:FFD1	RTI	Vrti
0xFFD2:FFD3	Reserved	—
0xFFD4:FFD5	Reserved	—
0xFFD6:FFD7	ACMP	Vacmp
0xFFD8:FFD9	ADC Conversion	Vadc
0xFFDA:FFDB	KBI Interrupt	Vkeyboard
0xFFDC:FFDD	Reserved	Viic
0xFFDE:FFDF	Reserved	Vscitx
0xFFE0:FFE1	Reserved	Vscirx
0xFFE2:FFE3	Reserved	Vscierr
0xFFE4:FFE5	Reserved	Vspi
0xFFE6:FFE7	MTIM Overflow	Vmtim
0xFFE8:FFE9	Reserved	—
0xFFEA:FFEB	Reserved	—
0xFFEC:FFED	Reserved	—
0xFFEE:FFEF	Reserved	—
0xFFFF0:FFF1	TPM Overflow	Vtpmovf
0xFFFF2:FFF3	Reserved	—
0xFFFF4:FFF5	TPM Channel 0	Vtpmch0
0xFFFF6:FFF7	Reserved	—
0xFFFF8:FFF9	Low Voltage Detect	Vlvd
0xFFFFA:FFFB	IRQ	Virq
0xFFFFC:FFFD	SWI	Vswi
0xFFFFE:FFFF	Reset	Vreset

4.3 Register Addresses and Bit Assignments

The registers in the MC9S08QA4 series are divided into these groups:

- Direct-page registers are located in the first 96 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located from 0x1800 and above in the memory map. This leaves more room in the direct page for more frequently used registers and RAM.
- The nonvolatile register area consists of a block of 16 locations in flash memory at 0xFFB0–0xFFBF. Nonvolatile register locations include:
 - NVPROT and NVOPT, which are loaded into working registers at reset.
 - An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory.

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

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

The direct page registers in [Table 4-2](#) can use the more efficient direct addressing mode that requires only the lower byte of the address. Because of this, the lower byte of the address in column one is shown in bold text. In [Table 4-3](#) and [Table 4-4](#), 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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	0	0	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	0	0	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	Reserved	—	—	—	—	—	—	—	—
0x0003	PORT	PORT							
0x0004	Reserved	—	—	—	—	—	—	—	—
0x000B		—	—	—	—	—	—	—	—
0x000C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x000D	KBIPE	0	0	0	0	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x000E	KBIES	0	0	0	0	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x000F	IRQSC	0	IRQPDD	0	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x0010	ADCSC1	COCO	AIEN	ADCO	ADCH				
0x0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	—	—	—	—
0x0012	ADCRH	0	0	0	0	0	0	ADR9	ADR8
0x0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0

Table 4-2. Direct-Page Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0014	ADCCVH	0	0	0	0	0	0	ADCV9	ADCV8
0x0015	ADCCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	APCTL1	0	0	0	0	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	Reserved	0	0	0	0	0	0	0	0
0x0019	Reserved	0	0	0	0	0	0	0	0
0x001A	ACMPSC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD	
0x001B– 0x0037	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x0038	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0039	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x003A	ICSTRM	TRIM							
0x003B	ICSSC	0	0	0	0	CLKST		OSCINIT	FTRIM
0x003C	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x003D	MTIMCLK	0	0	CLKS			PS		
0x003E	MTIMCNT	COUNT							
0x003F	MTIMMOD	MOD							
0x0040	TPMSC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0041	TPMCNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0042	TPMCNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0043	TPMMODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0044	TPMMODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0045	TPMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0046	TPMC0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0047	TPMC0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0048– 0x005F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBDFFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPE	COPT	STOPE	—	0	0	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	0	0	0	0	0	0	ACIC
0x1804	Reserved	—	—	—	—	—	—	—	—
0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

Table 4-3. High-Page Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1808	SRTISC	RTIF	RTIACK	0	RTIE	0	RTIS		
0x1809	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x180A	SPMSC2	0	0	0	PDF	PPDF	PPDACK	PDC	PPDC
0x180B	Reserved	—	—	—	—	—	—	—	—
0x180C	SPMSC3	LVWF	LVWACK	LVDV	LVWV	—	—	—	—
0x180D– 0x180F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGFHH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBGCH	DBGCH	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	DBGTH	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	DBGS	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
0x1819– 0x181F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1820	FCDIV	DIVLD	PRDIV8	DIV					
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS							FPDIS
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD							
0x1827– 0x183F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1840	PTAPE	0	0	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	PTASE	0	0	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	0	0	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843– 0x1847	Reserved	—	—	—	—	—	—	—	—

Nonvolatile flash registers, shown in [Table 4-4](#), are located in the flash memory. These registers include an 8-byte backdoor key that 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
0xFFAE	Reserved for Storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Reserved for Storage of ICSTRM	TRIM							
0xFFB0 – 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8 – 0xFFBC	Unused	—	—	—	—	—	—	—	—
0xFFBD	NVPROT	FPS							FPDIS
0xFFBE	Unused	—	—	—	—	—	—	—	—
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).

4.4 RAM

The MC9S08QA4 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 or after wakeup from stop1, 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 (V_{RAM}).

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08QA4 series, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor-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 4.6, “Security,”](#) for a detailed description of the security feature.

The RAM array is not automatically initialized out of reset.

4.5 Flash

The flash memory is used 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.

4.5.1 Features

Features of the flash memory include:

- Flash size
 - MC9S08QA4: 4,096 bytes (8 pages of 512 bytes each)
 - MC9S08QA2: 2,048 bytes (4 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for flash and RAM
- Auto power-down for low-frequency read accesses

4.5.2 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 4.7.1, “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 ($1/f_{\text{FCLK}}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses is used by the command processor to complete a program or erase command.

[Table 4-5](#) 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 $t_{\text{FCLK}} = 1/f_{\text{FCLK}}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{\text{FCLK}} = 5 \mu\text{s}$. 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 μ s
Byte program (burst)	4	20 μ s ¹
Page erase	4000	20 ms
Mass erase	20,000	100 ms

¹ Excluding start/end overhead

NOTE

If the COP is enabled during an erase function, make sure the COP is serviced during the erase command execution.

4.5.3 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.

NOTE

Do not program any byte in the flash more than once after a successful erase operation. Reprogramming bits to a byte that 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 must be done only once following a reset.

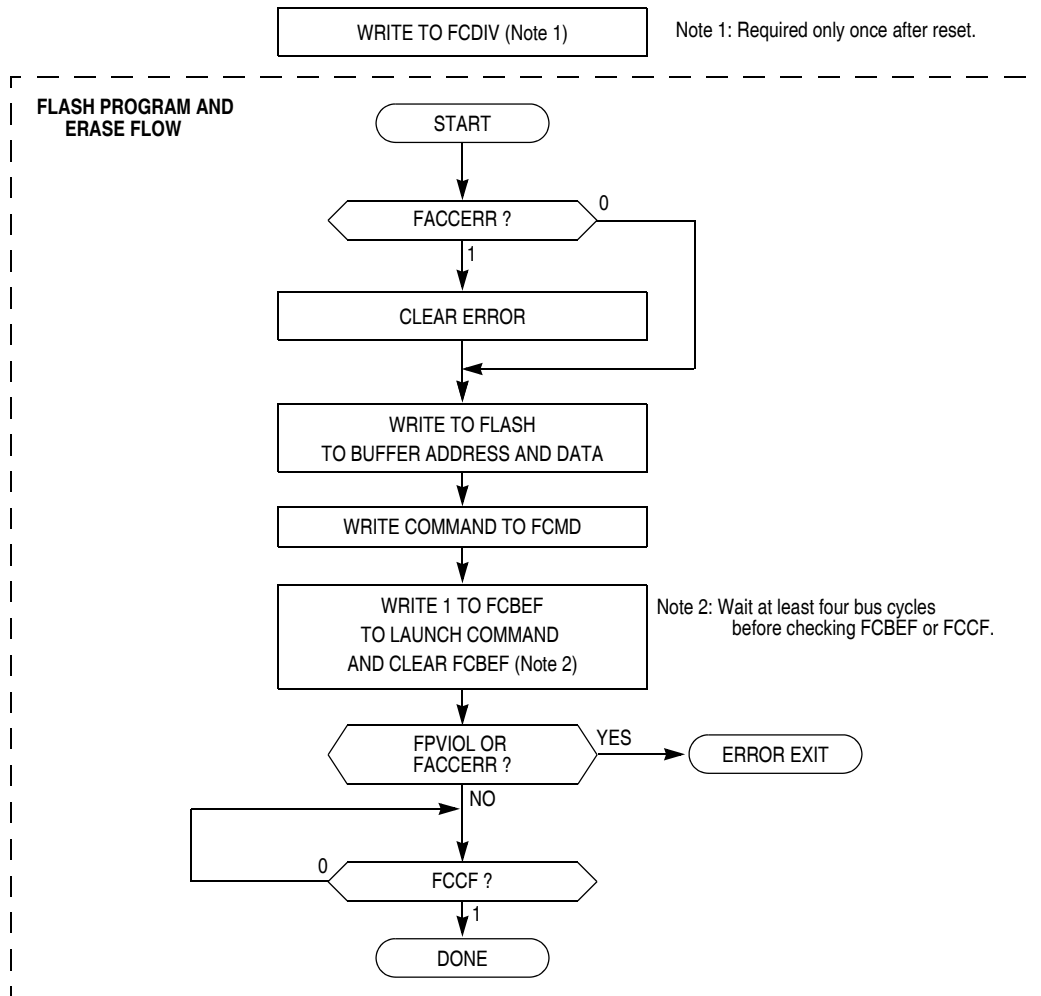


Figure 4-2. Flash Program and Erase Flowchart

4.5.4 Burst Program Execution

The burst program command is used to program sequential bytes of data in less time than would be required using the standard program command. This is possible because the high voltage to the flash array does not need to be disabled between program operations. Ordinarily, when a program or erase command is issued, an internal charge pump associated with the flash memory must be enabled to supply high voltage to the array. Upon completion of the command, the charge pump is turned off. When a burst program command is issued, the charge pump is enabled and 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. If the next sequential address is the beginning of a new row, the program time for that byte will be the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump will be disabled and high voltage removed from the array.

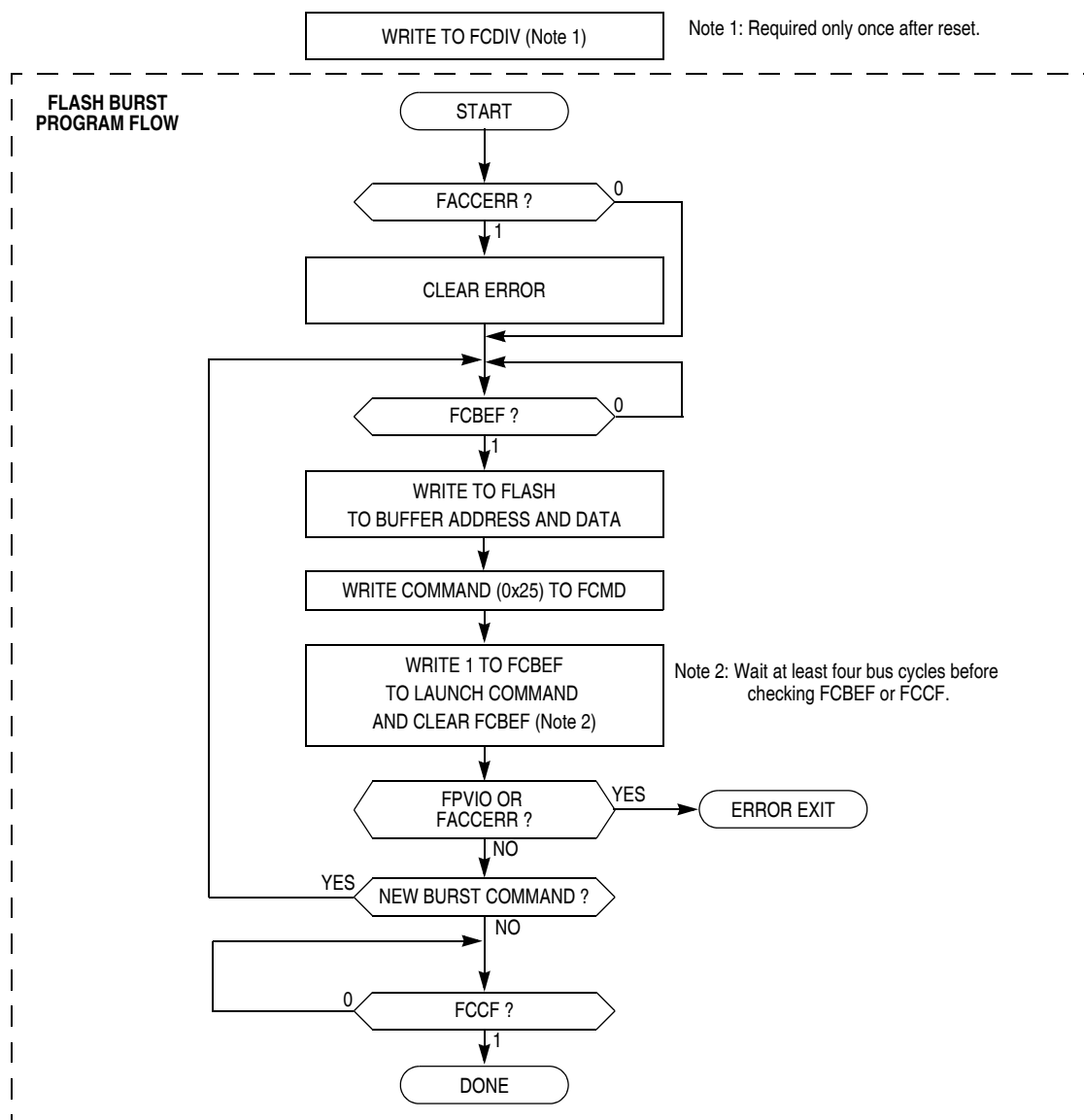


Figure 4-3. Flash Burst Program Flowchart

4.5.5 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
- Accessing (read or write) 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 entering 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

4.5.6 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 4.7.4, “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.

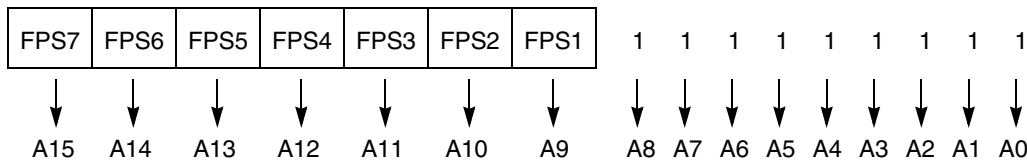


Figure 4-4. Block Protection Mechanism

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.

4.5.7 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.

4.6 Security

The MC9S08QA4 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 must 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 (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.

4.7 Flash Registers and Control Bits

The flash module has six 8-bit registers in the high-page register space. Two locations (NVOPT, NVPROT) in the nonvolatile register space in flash memory are copied into corresponding high-page

control registers (FOPT, FPROT) 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 Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.7.1 Flash Clock Divider Register (FCDIV)

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

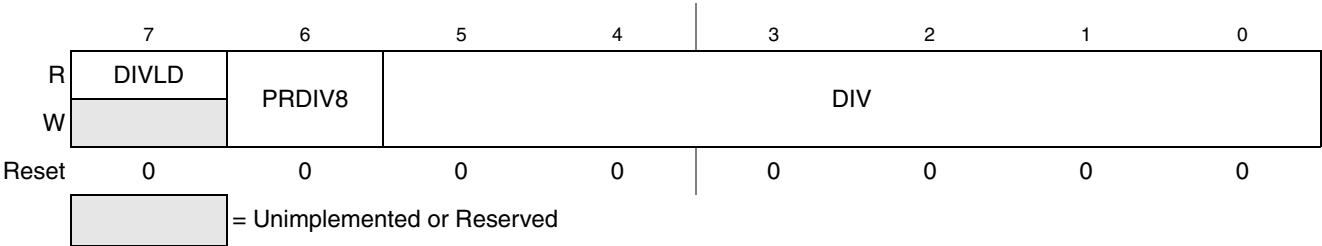


Figure 4-5. Flash Clock Divider Register (FCDIV)

Table 4-6. FCDIV Register Field Descriptions

Field	Description
7 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. 0 FCDIV has not been written since reset; erase and program operations disabled for flash. 1 FCDIV has been written since reset; erase and program operations enabled for flash.
6 PRDIV8	Prescale (Divide) Flash Clock by 8 0 Clock input to the flash clock divider is the bus rate clock. 1 Clock input to the flash clock divider is the bus rate clock divided by 8.
5:0 DIV	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 DIV 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 μ s to 6.7 μ s. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See Equation 4-1 and Equation 4-2 .

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

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

[Table 4-7](#) shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

Table 4-7. Flash Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV (Decimal)	f_{CLK}	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

4.7.2 Flash Options Register (FOPT and NVOPT)

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

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	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 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 4.6, “Security.” 0 No backdoor key access allowed. 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 FNORED	Vector Redirection Disable — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 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. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash. For more detailed information about security, refer to Section 4.6, “Security.”

Table 4-9. Security States¹

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

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

4.7.3 Flash Configuration Register (FCNFG)

	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 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 4.6, “Security.” 0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a flash programming or erase command. 1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.

4.7.4 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.

	7	6	5	4	3	2	1	0
R	FPS ¹							FPDIS ¹
W								
Reset	This register is loaded from nonvolatile location NVPROT during reset.							

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

Figure 4-8. Flash Protection Register (FPROT)

Table 4-11. FPROT Register Field Descriptions

Field	Description
7:1 FPS	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 FPDIS	Flash Protection Disable 0 Flash block specified by FPS7:FPS1 is block protected (program or erase not allowed). 1 No flash block is protected.

4.7.5 Flash Status Register (FSTAT)

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

= Unimplemented or Reserved

Figure 4-9. Flash Status Register (FSTAT)

Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 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 1 to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command can be written to the command buffer.
6 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. 0 Command in progress 1 All commands complete
5 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. 0 No protection violation. 1 An attempt was made to erase or program a protected location.

Table 4-12. FSTAT Register Field Descriptions (continued)

Field	Description
4 FACCERR	Access Error Flag — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.5.5, “Access Errors.” FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect. 0 No access error. 1 An access error has occurred.
2 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. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the flash array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the flash array is completely erased (all 0xFF).

4.7.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in [Table 4-13](#). Refer to [Section 4.5.3, “Program and Erase Command Execution,”](#) for a detailed discussion of flash programming and erase operations.

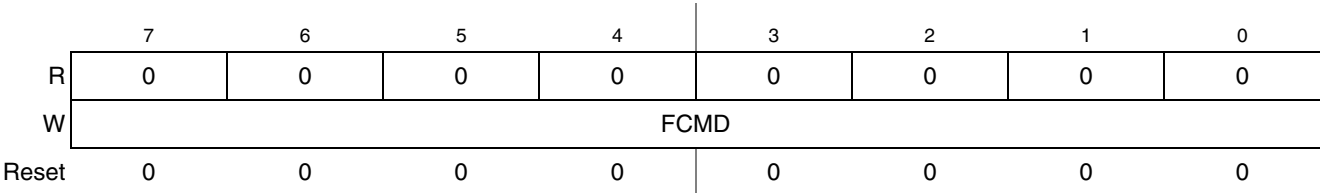


Figure 4-10. Flash Command Register (FCMD)

Table 4-13. 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. The blank check command is only required as part of the security unlocking mechanism.

Chapter 5

Resets, Interrupts, and General System Control

5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08QA4 series. Some interrupt sources from peripheral modules are discussed in greater detail within other chapters of this reference 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 chapters 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
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see [Table 5-2](#))

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 (0xFFFF: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 MC9S08QA4 series have the following sources for reset:

- External pin reset (PIN) — enabled using RSTPE in SOPT1
- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (IOP)
- Illegal address detect (ILAD)
- Background debug force reset

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

5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is used 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 SOPT1 enabling the COP watchdog (see [Section 5.8.4, “System Options Register 1 \(SOPT1\)”](#) for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing 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 5.8.5, “System Options Register 2 \(SOPT2\)”](#) for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1 kHz clock source. With each clock source, there is an associated short and long time-out controlled by COPT in SOPT1. [Table 5-1](#) summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz clock source and the associated long time-out (2^8 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 MC9S08QA4 Series *Data Sheet* for the tolerance of this value.

Even if the application uses the reset default settings of COPE, COPCLKS, and COPT, the user must write to the write-once SOPT1 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 SOPT1 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 was 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 $\overline{\text{IRQ}}$ 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 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:

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

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

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

NOTE

For compatibility with M68HC08 devices, the H register is not automatically saved and restored. 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 content 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.

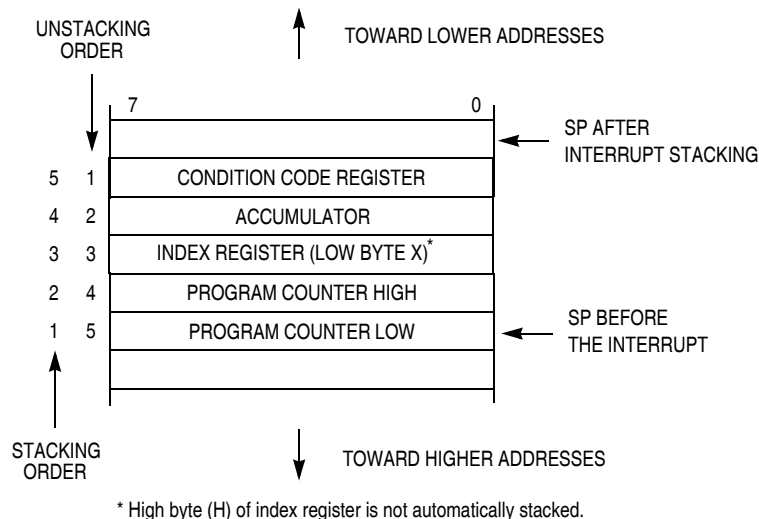


Figure 5-1. Interrupt Stack Frame

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 is cleared at the beginning of the ISR so that if another interrupt is generated by this same source, it will be registered so it can be serviced after completion of the current ISR.

5.5.2 External Interrupt Request Pin ($\overline{\text{IRQ}}$)

External interrupts are managed by the IRQ status and control register, IRQSC. When the $\overline{\text{IRQ}}$ 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 $\overline{\text{IRQ}}$ (if enabled) can wake the MCU.

5.5.2.1 Pin Configuration Options

The $\overline{\text{IRQ}}$ pin enable (IRQPE) control bit in IRQSC must be 1 for the $\overline{\text{IRQ}}$ pin to act as the interrupt request (IRQ) input. As an IRQ input, the user can choose 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 $\overline{\text{IRQ}}$ pin, when enabled, defaults to use an internal pullup device ($\text{IRQPDD} = 0$). If the user desires to use an external pullup, 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 $\overline{\text{IRQ}}$ 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 must not be driven above V_{DD} .

The voltage measured on the internally pulled-up $\overline{\text{IRQ}}$ pin will not be pulled to V_{DD} . The internal gates connected to this pin are pulled to V_{DD} . The $\overline{\text{IRQ}}$ pullup must not be used to pull up components external to the MCU. The internal gates connected to this pin are pulled all the way to V_{DD} .

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 this edge detection mode, the IRQF status flag becomes set when an edge is detected (when the $\overline{\text{IRQ}}$ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the $\overline{\text{IRQ}}$ 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 Number	Address (High:Low)	Vector Name	Module	Source	Enable	Description
Lower ↑ ↓ Higher	31 through 24	0xFFC0:FFC1 through 0xFFCE:FFCF	Unused Vector Space (available for user program)				
	23	0xFFD0:FFD1	Vrti	System control	RTIF	RTIE	Real-time interrupt
	22	0xFFD2:FFD3	—	—	—	—	—
	21	0xFFD4:FFD5	—	—	—	—	—
	20	0xFFD6:FFD7	Vacmp	ACMP	ACF	ACIE	ACMP
	19	0xFFD8:FFD9	Vadc	ADC	COCO	AIEN	ADC
	18	0xFFDA:FFDB	Vkeyboard	KBI	KBF	KBIE	Keyboard pins
	17	0xFFDC:FFDD	—	—	—	—	—
	16	0xFFDE:FFDF	—	—	—	—	—
	15	0xFFE0:FFE1	—	—	—	—	—
	14	0xFFE2:FFE3	—	—	—	—	—
	13	0xFFE4:FFE5	—	—	—	—	—
	12	0xFFE6:FFE7	Vmtim	MTIM	TOF	TOIE	MTIM
	11	0xFFE8:FFE9	—	—	—	—	—
	10	0xFFEA:FFEB	—	—	—	—	—
	9	0xFFEC:FFED	—	—	—	—	—
	8	0xFFEE:FFEF	—	—	—	—	—
	7	0xFFFF0:FFF1	Vtpmovf	TPM	TOF	TOIE	TPM overflow
	6	0xFFFF2:FFF3	—	—	—	—	—
	5	0xFFFF4:FFF5	Vtpmch0	TPM	CH0F	CH0IE	TPM channel 0
	4	0xFFFF6:FFF7	—	—	—	—	—
	3	0xFFFF8:FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect
	2	0xFFFFA:FFFB	Virq	IRQ	IRQF	IRQIE	$\overline{\text{IRQ}}$ pin
	1	0xFFFFC:FFFD	Vswi	CPU	SWI Instruction	—	Software interrupt
	0	0xFFFFE:FFFF	Vreset	System control	COP LVD $\overline{\text{RESET}}$ pin Illegal opcode Illegal address POR	COPE LVDRE RSTPE — — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address power-on-reset

5.6 Low-Voltage Detect (LVD) System

The MC9S08QA4 series include a system to protect against low voltage conditions. This system protects memory contents and controls MCU system states during supply voltage variations, and is comprised of a power-on reset (POR) circuit and a LVD circuit with a user selectable trip voltage, 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 SPMSC3. The LVD is disabled upon entering any of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU cannot enter stop1 or 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 MCU 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 using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF in SPMSC1 will be set and an LVD interrupt request 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 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 SPMSC3.

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. External clock input requires the XOSC module; consult [Table 1-1](#) to see if your MCU contains this module. 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 (EREFSTEN = 1) and configured for low frequency operation (RANGE = 0). Only the internal 1-kHz clock source can be selected to wake the MCU from stop1 or stop2 modes.

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS) used to 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 0s, and no interrupts will be generated. See [Section 5.8.7, “System Real-Time Interrupt Status and Control Register \(SRTISC\),”](#) for detailed information about this register.

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

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

Refer to the direct-page register summary in [Chapter 4, “Memory Map and Register Definition,”](#) for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1, SOPT2, and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in [Chapter 3, “Modes of Operation.”](#)

5.8.1 Interrupt Pin Request Status and Control Register (IRQSC)

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

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

 = Unimplemented or Reserved

¹ Bit 5 is a reserved bit that must always be written to 0.

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

Table 5-3. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	Interrupt Request ($\overline{\text{IRQ}}$) Pull Device Disable — This read/write control bit is used to disable the internal pullup device when the $\overline{\text{IRQ}}$ pin is enabled ($\text{IRQPE} = 1$) allowing for an external device to be used. 0 $\overline{\text{IRQ}}$ pull device enabled if $\text{IRQPE} = 1$. 1 $\overline{\text{IRQ}}$ pull device disabled if $\text{IRQPE} = 1$.
4 IRQPE	$\overline{\text{IRQ}}$ Pin Enable — This read/write control bit enables the $\overline{\text{IRQ}}$ pin function. When this bit is set, the $\overline{\text{IRQ}}$ pin can be used as an interrupt request. 0 $\overline{\text{IRQ}}$ pin function is disabled. 1 $\overline{\text{IRQ}}$ pin function is enabled.
3 IRQF	IRQ Flag — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 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 ($\text{IRQMOD} = 1$), IRQF cannot be cleared while the $\overline{\text{IRQ}}$ pin remains at its asserted level.
1 IRQIE	IRQ Interrupt Enable — This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever $\text{IRQF} = 1$.
0 IRQMOD	IRQ Detection Mode — This read/write control bit selects either edge-only detection or edge-and-level detection. See Section 5.5.2.2, “Edge and Level Sensitivity,” for more details. 0 IRQ event on falling edges only. 1 IRQ event on falling edges and low levels.

5.8.2 System Reset Status Register (SRS)

This high-page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDPR register, all of the status bits in SRS will be cleared. 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	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u ¹	0	0	0	0	0	1	0
Any other reset:	0	Note ²	Note ²	Note ²	Note ²	0	0	0

¹ u = unaffected

² Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

Figure 5-3. System Reset Status (SRS)

Table 5-4. SRS Register Field Descriptions

Field	Description
7 POR	Power-On Reset — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	Computer Operating Properly (COP) Watchdog — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by COPE = 0. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 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 SOPT1 register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.

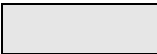
Table 5-4. SRS Register Field Descriptions (continued)

Field	Description
3 ILAD	Illegal Address — Reset was caused by an attempt to access data or an instruction at an unimplemented memory address. 0 Reset not caused by an illegal address. 1 Reset caused by an illegal address.
1 LVD	Low Voltage Detect — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

5.8.3 System Background Debug Force Reset Register (SBDFR)

This high-page register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write to 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 BDFR	Background Debug Force Reset — A serial background command such as WRITE_BYTE can be used to allow an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program. To enter user mode, PTA4/ACMPO/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTA4/ACMPO/BKGD/MS must be low immediately after issuing WRITE_BYTE command. See MC9S08QA4 Series <i>Data Sheet</i> for more information.

5.8.4 System Options Register 1 (SOPT1)

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

	7	6	5	4 ¹	3	2	1	0
R	COPE	COPT	STOPE		0	0	BKGDPE	RSTPE
W								
Reset:	1	1	0	1	0	0	1	u ²
POR:	1	1	0	1	0	0	1	0
LVD:	1	1	0	1	0	0	1	0

 = Unimplemented or Reserved

¹ Bit 4 is reserved; writes will change the value but will have no effect on this MCU.

² u = unaffected

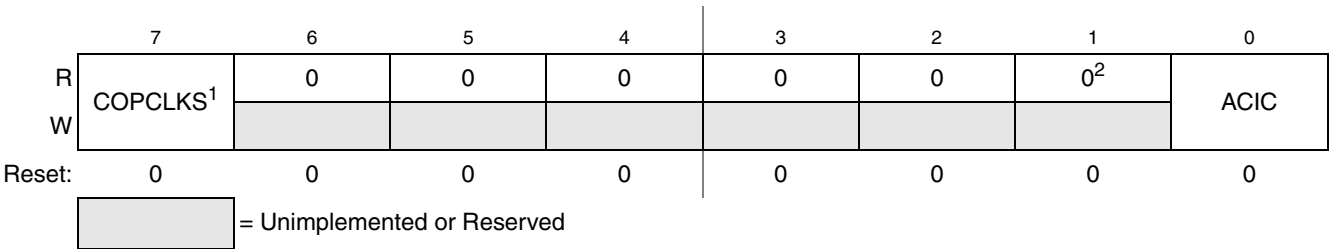
Figure 5-5. System Options Register 1 (SOPT1)

Table 5-6. SOPT1 Register Field Descriptions

Field	Description
7 COPE	COP Watchdog Enable — This write-once bit selects whether the COP watchdog is enabled. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	COP Watchdog Timeout — This write-once bit selects the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. 0 Short timeout period selected. 1 Long timeout period selected.
5 STOPE	Stop Mode Enable — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
1 BKGDPE	Background Debug Mode Pin Enable — This write-once bit when set enables the PTA4/ACMPO/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output-only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset. 0 PTA4/ACMPO/BKGD/MS pin functions as PTA4 or ACMPO. 1 PTA4/ACMPO/BKGD/MS pin functions as BKGD/MS.
0 RSTPE	RESET Pin Enable — This write-once bit when set enables the PTA5/ $\overline{\text{IRQ}}$ /TCLK/ $\overline{\text{RESET}}$ pin to function as $\overline{\text{RESET}}$. When clear, the pin functions as one of its input-only alternative functions. This pin defaults to its input-only port function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on $\overline{\text{RESET}}$. 0 PTA5/ $\overline{\text{IRQ}}$ /TCLK/ $\overline{\text{RESET}}$ pin functions as PTA5, $\overline{\text{IRQ}}$, or TCLK. 1 PTA5/ $\overline{\text{IRQ}}$ /TCLK/ $\overline{\text{RESET}}$ pin functions as $\overline{\text{RESET}}$.

5.8.5 System Options Register 2 (SOPT2)

This high-page register contains bits to configure MCU specific features on the MC9S08QA4 Series devices.



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

² This bit must not be written to 1

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

Table 5-7. SOPT2 Register Field Descriptions

Field	Description
7 COPCLKS	COP Watchdog Clock Select — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz clock is source to COP. 1 Bus clock is source to COP.
0 ACIC	Analog Comparator to Input Capture Enable — This bit connects the output of ACMP to TPM input channel 0. 0 ACMP output not connected to TPM input channel 0. 1 ACMP output connected to TPM input channel 0.

5.8.6 System Device Identification Register (SDIDH, SDIDL)

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

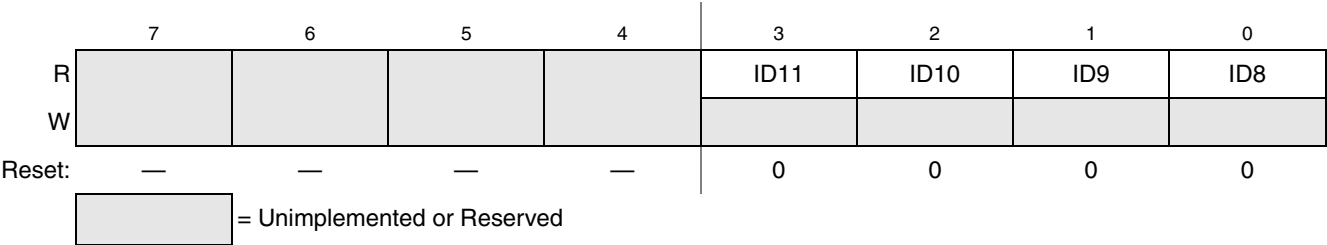


Figure 5-7. System Device Identification Register — High (SDIDH)

Table 5-8. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.
3:0 ID[11:8]	Part Identification Number — Each derivative in the HCS08 Family has a unique identification number. The MC9S08QA4 is hard coded to the value 0x009. See also ID bits in Table 5-9 .

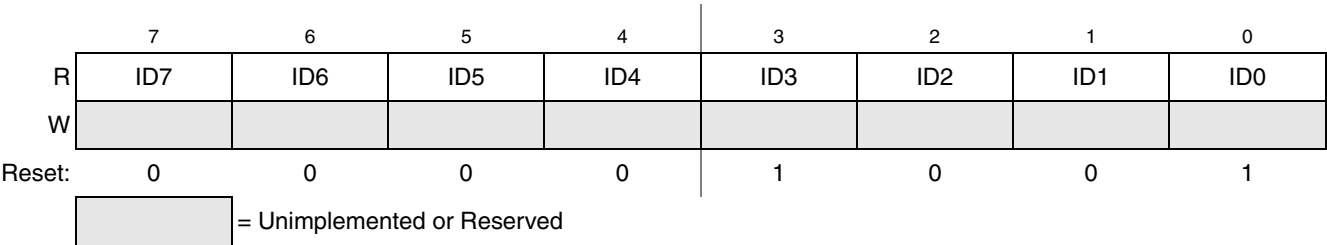


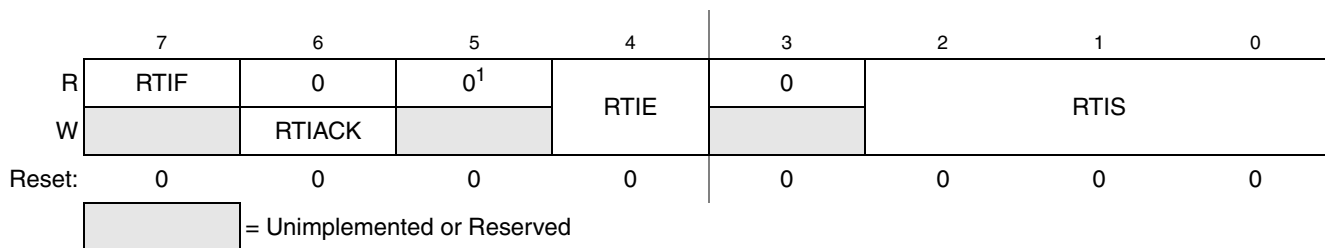
Figure 5-8. System Device Identification Register — Low (SDIDL)

Table 5-9. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	Part Identification Number — Each derivative in the HCS08 Family has a unique identification number. The MC9S08QA4 is hard coded to the value 0x009. See also ID bits in Table 5-8 .

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

This high-page register contains status and control bits for the RTI.



¹ Do not set this bit to 1

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

Table 5-10. SRTISC Register Field Descriptions

Field	Description
7 RTIF	Real-Time Interrupt Flag — This read-only status bit indicates the periodic wakeup timer has timed out. 0 Periodic wakeup timer not timed out. 1 Periodic wakeup timer timed out.
6 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 0.
4 RTIE	Real-Time Interrupt Enable — This read-write bit enables real-time interrupts. 0 Real-time interrupts disabled. 1 Real-time interrupts enabled.
2:0 RTIS	Real-Time Interrupt Delay Selects — These read/write bits select the period for the RTI. See Table 5-11 .

Table 5-11. Real-Time Interrupt Period

RTIS2:RTIS1:RTIS0	Using Internal 1 kHz Clock Source ^{1 2}	Using External Clock Source Period = t_{ext} ³
0:0:0	Disable RTI	Disable RTI
0:0:1	8 ms	$t_{ext} \times 256$
0:1:0	32 ms	$t_{ext} \times 1024$
0:1:1	64 ms	$t_{ext} \times 2048$
1:0:0	128 ms	$t_{ext} \times 4096$
1:0:1	256 ms	$t_{ext} \times 8192$
1:1:0	512 ms	$t_{ext} \times 16384$
1:1:1	1.024 s	$t_{ext} \times 32768$

¹ Values shown in this column are based on $t_{RTI} = 1$ ms. See t_{RTI} in MC9S08QA4 Series *Data Sheet* for the tolerance of this value.

² The initial RTI timeout period will be up to one 1 kHz clock period less than the time specified.

³ t_{ext} is the period of the external crystal frequency.

5.8.8 System Power Management Status and Control 1 Register (SPMSC1)

This high-page register contains status and control bits to support the low-voltage detect function and to enable the bandgap voltage reference for use by the ADC module. To configure the low-voltage detect trip voltage, see [Table 5-14](#) for the LVDV bit description in SPMSC3.

	7	6	5	4	3	2	1 ¹	0
R	LVDF	0	LVDIE	LVDRE ²	LVDSE	LVDE ²	0	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)

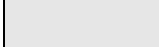
Table 5-12. SPMSC1 Register Field Descriptions

Field	Description
7 LVDF	Low-Voltage Detect Flag — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event.
6 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 LVDIE	Low-Voltage Detect Interrupt Enable — This bit enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF = 1.
4 LVDRE	Low-Voltage Detect Reset Enable — This write-once bit enables LVDF events to generate a hardware reset (provided LVDE = 1). 0 LVDF does not generate hardware resets. 1 Force an MCU reset when LVDF = 1.
3 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. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	Low-Voltage Detect Enable — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	Bandgap Buffer Enable — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels or as a voltage reference for ACMP module. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

5.8.9 System Power Management Status and Control 2 Register (SPMSC2)

This high-page register contains status and control bits to configure the stop mode behavior of the MCU. See [Section 3.6, “Stop Modes,”](#) for more information on stop modes.

	7	6	5	4	3	2	1	0
R	0	0	0	PDF	PPDF	0	PDC ¹	PPDC ¹
W						PPDACK		
Reset:	0	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 5-11. System Power Management Status and Control 2 Register (SPMSC2)

Table 5-13. SPMSC2 Register Field Descriptions

Field	Description
4 PDF	Power Down Flag — This read-only status bit indicates the MCU has recovered from stop1 mode. 0 MCU has not recovered from stop1 mode. 1 MCU recovered from stop1 mode.
3 PPDF	Partial Power Down Flag — This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	Partial Power Down Acknowledge — Writing a 1 to PPDACK clears the PPDF and the PDF bits.
1 PDC	Power Down Control — The PDC bit controls entry into the power down (stop2 and stop1) modes. 0 Power down modes are disabled. 1 Power down modes are enabled.
0 PPDC	Partial Power Down Control — The PPDC bit controls which power down mode is selected. 0 Stop1, full power down mode enabled if PDC set. 1 Stop2, partial power down mode enabled if PDC set.

5.8.10 System Power Management Status and Control 3 Register (SPMSC3)

This high-page register is used to report the status of the low-voltage warning function and to select the low-voltage detect trip voltage.

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	0	0	0	0
W		LVWACK						
POR:	0 ¹	0	0	0	0	0	0	0
LVD:	0 ¹	0	U	U	0	0	0	0
Any other reset:	0 ¹	0	U	U	0	0	0	0



= Unimplemented or Reserved

U= Unaffected by reset

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

Figure 5-12. System Power Management Status and Control 3 Register (SPMSC3)

Table 5-14. SPMSC3 Register Field Descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag — The LVWF bit indicates the low voltage warning status. 0 Low voltage warning not present. 1 Low voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge — The LVWF bit indicates the low voltage warning status. Writing a 1 to LVWACK clears LVWF to a 0, if a low voltage warning is not present.
5 LVDV	Low Voltage Detect Voltage Select — The LVDV bit selects the LVD trip point voltage (V_{LVD}). 0 Low trip point selected ($V_{\text{LVD}} = V_{\text{LVDL}}$) 1 High trip point selected ($V_{\text{LVD}} = V_{\text{LV DH}}$)
4 LVWV	Low Voltage Warning Voltage Select — The LVWV bit selects the LVW trip point voltage (V_{LVW}). 0 Low trip point selected ($V_{\text{LVW}} = V_{\text{LVWL}}$) 1 High trip point selected ($V_{\text{LVW}} = V_{\text{LVWH}}$)

Chapter 6

Parallel Input/Output Control

This chapter explains software controls related to parallel input/output (I/O) and pin control. The MC9S08QA4 has two parallel I/O ports which include a total of four I/O pins, one output-only pin and one input-only pin. See [Chapter 2, “External Signal Description,”](#) for more information about the pin assignments and external hardware considerations of these pins. Not all pins are available on all devices of the MC9S08QA4 series family. See [Table 1-1](#) for the number of general-purpose pins available on the device.

All of these I/O pins are shared with on-chip peripheral functions as shown in [Table 2-2](#). The peripheral modules have priority over the I/Os so that when a peripheral is enabled, the I/O functions associated with that shared pins are disabled. After reset, the shared peripheral functions are disabled so that the pins are controlled by the I/O. All of the I/Os are configured as inputs ($PTxDDn = 0$) with pullup devices disabled ($PTxPEN = 0$), except for output-only pin PTA4 which defaults to the BKGD/MS pin.

NOTE

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user reset initialization routine in the application program must either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

6.1 Port Data and Data Direction

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

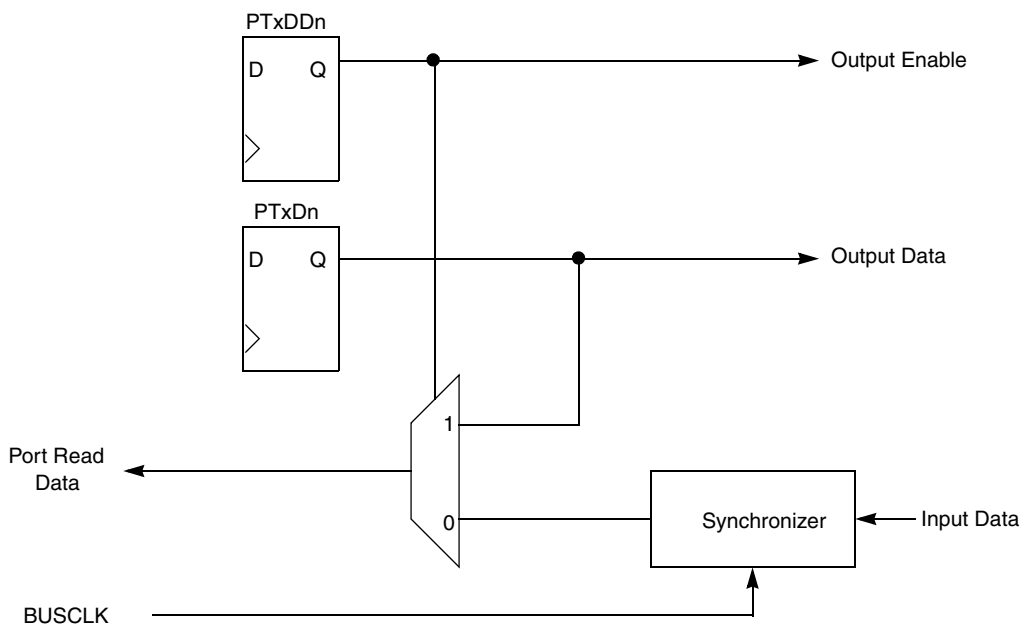


Figure 6-1. Parallel I/O Block Diagram

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

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

When a shared analog function is enabled for a pin, both the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PTxDDn = 0) and the input buffer is disabled. In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority. So if both the digital and analog functions are enabled, the analog function controls the pin.

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.

6.2 Pin Control — Pullup, Slew Rate, and Drive Strength

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

6.3 Pin Behavior in Stop Modes

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

- In stop1 mode, all internal registers including parallel I/O control and data registers are powered off. Each of the pins assumes its default reset state (output buffer and internal pullup disabled). Upon exit from stop1, all pins must be re-configured to the same status they would have if the MCU had been reset by POR.
- Stop2 mode is a partial power-down mode, whereby latches maintain the pin state as before the STOP instruction was executed. CPU register status and the state of I/O registers must be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode. Upon recovery from stop2 mode, the user must examine the state of the PPDF bit in the SPMSC2 register before accessing any I/O. 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, and in peripherals previously enabled, will need to be 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 pins is now permitted again in the user application program.
- In stop3 mode, all pin states are maintained because internal logic stays powered up. Upon recovery, all pin functions are the same as before entering stop3.

6.4 Parallel I/O Registers

6.4.1 Port A Registers

This section provides information about the registers associated with the parallel I/O ports.

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

6.4.1.1 Port A Data (PTAD)

	7	6	5	4	3	2	1	0
R	0	0	PTAD5 ¹	PTAD4 ²	PTAD3	PTAD2	PTAD1	PTAD0
W								
Reset:	0	0	0	0	0	0	0	0

¹ Reads of bit PTAD5 always return the pin value of PTA5, regardless of the value stored in bit PTADD5.

² Reads of bit PTAD4 always return the contents of PTAD4, regardless of the value stored in bit PTADD4.

Figure 6-2. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
5:0 PTAD[5:0]	Port A Data Register Bits — For port A pins that are inputs, reads return the logic level on the pin. For port A pins that are configured as outputs, reads return the last value written to this register. 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. 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.

6.4.1.2 Port A Data Direction (PTADD)

	7	6	5	4	3	2	1	0
R	0	0	PTADD5 ¹	PTADD4 ²	PTADD3	PTADD2	PTADD1	PTADD0
W								
Reset:	0	0	0	0	0	0	0	0

¹ PTADD5 has no effect on the input-only PTA5 pin.

² PTADD4 has no effect on the output-only PTA4 pin.

Figure 6-3. Port A Data Direction Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
5:0 PTADD[5:0]	Data Direction for Port A Bits — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn.

6.4.2 Port A Control Registers

The pins associated with port A are controlled by the registers in this section. These registers control the pin pullup, slew rate, and drive strength of the port A pins independent of the parallel I/O register.

6.4.2.1 Port A Internal Pullup Enable (PTAPE)

An internal pullup device can be enabled for each port pin by setting the corresponding bit in the pullup enable register (PTAPEn). 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.

	7	6	5	4	3	2	1	0
R	0	0	PTAPE5	PTAPE4 ¹	PTAPE3	PTAPE2	PTAPE1	PTAPE0
W								
Reset:	0	0	0	0	0	0	0	0

¹ PTAPE4 has no effect on the output-only PTA4 pin.

Figure 6-4. Internal Pullup Enable for Port A Register (PTAPE)

Table 6-3. PTAPE Register Field Descriptions

Field	Description
5:0 PTAPE[5:0]	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. 0 Internal pullup device disabled for port A bit n. 1 Internal pullup device enabled for port A bit n.

6.4.2.2 Port A Slew Rate Enable (PTASE)

Slew rate control can be enabled for each port pin by setting the corresponding bit in the slew rate control register (PTASEn). When enabled, slew control limits the rate at which an output can transition. This helps to reduce EMC emissions. Slew rate control has no effect on pins which are configured as [Figure 6-5](#) inputs.

	7	6	5	4	3	2	1	0
R	0	0	PTASE5 ¹	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
W								
Reset:	0	0	1	1	1	1	1	1

¹ PTASE5 has no effect on the input-only PTA5 pin.

Figure 6-5. Slew Rate Enable for Port A Register (PTASE)

Table 6-4. PTASE Register Field Descriptions

Field	Description
5:0 PTASE[5:0]	Output Slew Rate Enable for Port A Bits — Each of these control bits determines if the output slew rate control is enabled for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.

6.4.2.3 Port A Drive Strength Select (PTADS)

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PTADS). 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 affects 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.

6.4.2.4 Port A Drive Strength Select (PTADS)

	7	6	5	4	3	2	1	0
R	0	0	PTADS5 ¹	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
W								
Reset:	0	0	0	0	0	0	0	0

¹ PTADS5 has no effect on the input-only PTA5 pin.

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

Table 6-5. PTADS Register Field Descriptions

Field	Description
5:0 PTADS[5:0]	Output Drive Strength Selection for Port A Bits — Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port A bit n. 1 High output drive strength selected for port A 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*, Freescale Semiconductor document order number HCS08RMV1/D.

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

7.1.1 Features

Features of the HCS08 CPU include:

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

7.2 Programmer's Model and CPU Registers

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

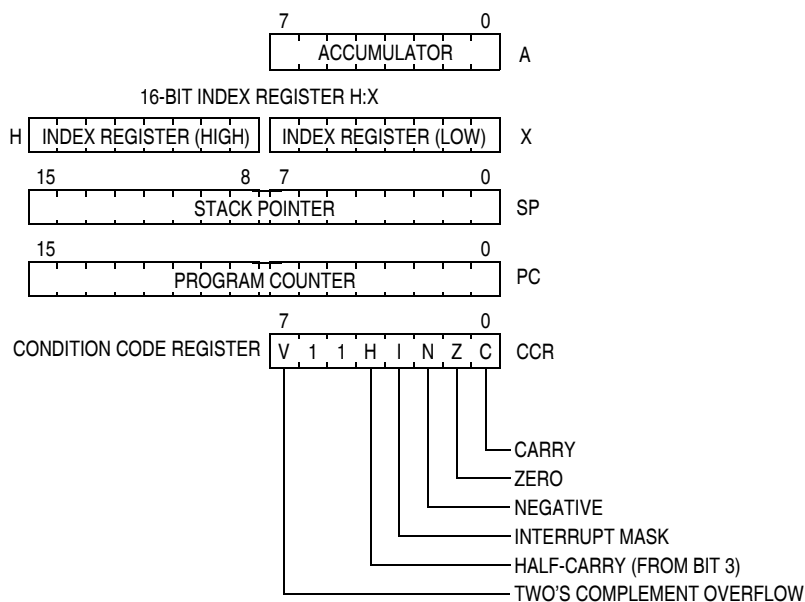


Figure 7-1. CPU Registers

7.2.1 Accumulator (A)

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

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

7.2.2 Index Register (H:X)

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

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

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

7.2.3 Stack Pointer (SP)

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

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

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

7.2.4 Program Counter (PC)

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

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

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

7.2.5 Condition Code Register (CCR)

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

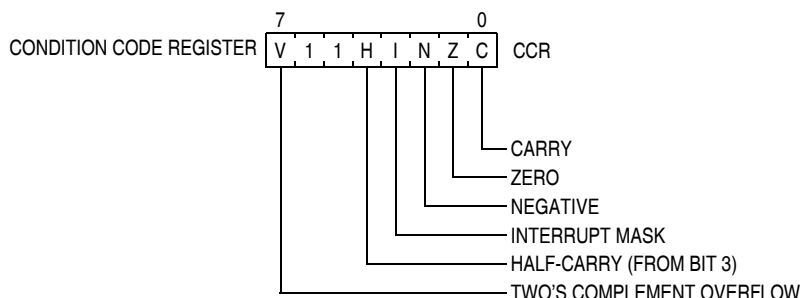


Figure 7-2. Condition Code Register

Table 7-1. CCR Register Field Descriptions

Field	Description
7 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. 0 No overflow 1 Overflow
4 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. 0 No carry between bits 3 and 4 1 Carry between bits 3 and 4
3 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. 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. 0 Interrupts enabled 1 Interrupts disabled
2 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. 0 Non-negative result 1 Negative result
1 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. 0 Non-zero result 1 Zero result
0 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. 0 No carry out of bit 7 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 0xFFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

7.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

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

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

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

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

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

7.4.3 Wait Mode Operation

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

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

7.4.4 Stop Mode Operation

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

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

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

7.4.5 BGND Instruction

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

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

7.5 HCS08 Instruction Set Summary

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

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

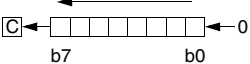
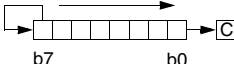
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry $A \leftarrow (A) + (M) + (C)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 ee ff E9 ff F9 9E D9 ee ff 9E E9 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↑↑	-	↑	↑	↑
ADD #opr8i ADD opr8a ADD opr16a ADD oprx16,X ADD oprx8,X ADD ,X ADD oprx16,SP ADD oprx8,SP	Add without Carry $A \leftarrow (A) + (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB ii BB dd CB hh ll DB ee ff EB ff FB 9E DB ee ff 9E EB ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↑↑	-	↑	↑	↑
AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer $SP \leftarrow (SP) + (M)$	IMM	A7 ii	2	pp	--	-	-	-	-
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X) $H:X \leftarrow (H:X) + (M)$	IMM	AF ii	2	pp	--	-	-	-	-
AND #opr8i AND opr8a AND opr16a AND oprx16,X AND oprx8,X AND ,X AND oprx16,SP AND oprx8,SP	Logical AND $A \leftarrow (A) \& (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 ii B4 dd C4 hh ll D4 ee ff E4 ff F4 9E D4 ee ff 9E E4 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0-	-	↑	↑	-
ASL opr8a ASLA ASLX ASL oprx8,X ASL ,X ASL oprx8,SP	Arithmetic Shift Left  (Same as LSL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rffwpp p p rffwpp rffw prffwpp	↑-	-	↑	↑	↑
ASR opr8a ASRA ASRX ASR oprx8,X ASR ,X ASR oprx8,SP	Arithmetic Shift Right 	DIR INH INH IX1 IX SP1	37 dd 47 57 67 ff 77 9E 67 ff	5 1 1 5 4 6	rffwpp p p rffwpp rffw prffwpp	↑-	-	↑	↑	↑
BCC rel	Branch if Carry Bit Clear (if C = 0)	REL	24 rr	3	ppp	--	-	-	-	-

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
BCLR <i>n,opr8a</i>	Clear Bit <i>n</i> in Memory ($M_n \leftarrow 0$)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 dd 13 dd 15 dd 17 dd 19 dd 1B dd 1D dd 1F dd	5 5 5 5 5 5 5 5	r fwpp r fwpp r fwpp r fwpp r fwpp r fwpp r fwpp r fwpp	--	--	--	--	--
BCS <i>rel</i>	Branch if Carry Bit Set (if $C = 1$) (Same as BLO)	REL	25 rr	3	ppp	--	--	--	--	--
BEQ <i>rel</i>	Branch if Equal (if $Z = 1$)	REL	27 rr	3	ppp	--	--	--	--	--
BGE <i>rel</i>	Branch if Greater Than or Equal To (if $N \oplus V = 0$) (Signed)	REL	90 rr	3	ppp	--	--	--	--	--
BGND	Enter active background if ENBDM=1 Waits for and processes BDM commands until GO, TRACE1, or TAGGO	INH	82	5+	fp...ppp	--	--	--	--	--
BGT <i>rel</i>	Branch if Greater Than (if $Z \mid (N \oplus V) = 0$) (Signed)	REL	92 rr	3	ppp	--	--	--	--	--
BHCC <i>rel</i>	Branch if Half Carry Bit Clear (if $H = 0$)	REL	28 rr	3	ppp	--	--	--	--	--
BHCS <i>rel</i>	Branch if Half Carry Bit Set (if $H = 1$)	REL	29 rr	3	ppp	--	--	--	--	--
BHI <i>rel</i>	Branch if Higher (if $C \mid Z = 0$)	REL	22 rr	3	ppp	--	--	--	--	--
BHS <i>rel</i>	Branch if Higher or Same (if $C = 0$) (Same as BCC)	REL	24 rr	3	ppp	--	--	--	--	--
BIH <i>rel</i>	Branch if IRQ Pin High (if IRQ pin = 1)	REL	2F rr	3	ppp	--	--	--	--	--
BIL <i>rel</i>	Branch if IRQ Pin Low (if IRQ pin = 0)	REL	2E rr	3	ppp	--	--	--	--	--
BIT # <i>opr8i</i> BIT <i>opr8a</i> BIT <i>opr16a</i> BIT <i>opr16,X</i> BIT <i>opr8,X</i> BIT <i>,X</i> BIT <i>opr16,SP</i> BIT <i>opr8,SP</i>	Bit Test (A) & (M) (CCR Updated but Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 ii B5 dd C5 hh ll D5 ee ff E5 ff F5 9E D5 ee ff 9E E5 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0-	-	↑	↓	-
BLE <i>rel</i>	Branch if Less Than or Equal To (if $Z \mid (N \oplus V) = 1$) (Signed)	REL	93 rr	3	ppp	--	--	--	--	--
BLO <i>rel</i>	Branch if Lower (if $C = 1$) (Same as BCS)	REL	25 rr	3	ppp	--	--	--	--	--
BLS <i>rel</i>	Branch if Lower or Same (if $C \mid Z = 1$)	REL	23 rr	3	ppp	--	--	--	--	--
BLT <i>rel</i>	Branch if Less Than (if $N \oplus V = 1$) (Signed)	REL	91 rr	3	ppp	--	--	--	--	--
BMC <i>rel</i>	Branch if Interrupt Mask Clear (if $I = 0$)	REL	2C rr	3	ppp	--	--	--	--	--
BMI <i>rel</i>	Branch if Minus (if $N = 1$)	REL	2B rr	3	ppp	--	--	--	--	--
BMS <i>rel</i>	Branch if Interrupt Mask Set (if $I = 1$)	REL	2D rr	3	ppp	--	--	--	--	--
BNE <i>rel</i>	Branch if Not Equal (if $Z = 0$)	REL	26 rr	3	ppp	--	--	--	--	--
BPL <i>rel</i>	Branch if Plus (if $N = 0$)	REL	2A rr	3	ppp	--	--	--	--	--

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
BRA <i>rel</i>	Branch Always (if I = 1)	REL	20 rr	3	ppp	--	--	--	--	--
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear (if (Mn) = 0)	DIR (b0)	01 dd rr	5	rpppp	--	--	--	--	↑
		DIR (b1)	03 dd rr	5	rpppp					
		DIR (b2)	05 dd rr	5	rpppp					
		DIR (b3)	07 dd rr	5	rpppp					
		DIR (b4)	09 dd rr	5	rpppp					
		DIR (b5)	0B dd rr	5	rpppp					
		DIR (b6)	0D dd rr	5	rpppp					
		DIR (b7)	0F dd rr	5	rpppp					
BRN <i>rel</i>	Branch Never (if I = 0)	REL	21 rr	3	ppp	--	--	--	--	--
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set (if (Mn) = 1)	DIR (b0)	00 dd rr	5	rpppp	--	--	--	--	↑
		DIR (b1)	02 dd rr	5	rpppp					
		DIR (b2)	04 dd rr	5	rpppp					
		DIR (b3)	06 dd rr	5	rpppp					
		DIR (b4)	08 dd rr	5	rpppp					
		DIR (b5)	0A dd rr	5	rpppp					
		DIR (b6)	0C dd rr	5	rpppp					
		DIR (b7)	0E dd rr	5	rpppp					
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory (Mn ← 1)	DIR (b0)	10 dd	5	rfwpp	--	--	--	--	--
		DIR (b1)	12 dd	5	rfwpp					
		DIR (b2)	14 dd	5	rfwpp					
		DIR (b3)	16 dd	5	rfwpp					
		DIR (b4)	18 dd	5	rfwpp					
		DIR (b5)	1A dd	5	rfwpp					
		DIR (b6)	1C dd	5	rfwpp					
		DIR (b7)	1E dd	5	rfwpp					
BSR <i>rel</i>	Branch to Subroutine PC ← (PC) + \$0002 push (PCL); SP ← (SP) – \$0001 push (PCH); SP ← (SP) – \$0001 PC ← (PC) + <i>rel</i>	REL	AD rr	5	ssppp	--	--	--	--	--
CBEQ <i>opr8a,rel</i>	Compare and... Branch if (A) = (M)	DIR	31 dd rr	5	rpppp	--	--	--	--	--
CBEQA # <i>opr8i,rel</i>	Branch if (A) = (M)	IMM	41 ii rr	4	pppp					
CBEQX # <i>opr8i,rel</i>	Branch if (X) = (M)	IMM	51 ii rr	4	pppp					
CBEQ <i>opr8,X+,rel</i>	Branch if (A) = (M)	IX1+	61 ff rr	5	rpppp					
CBEQ <i>,X+,rel</i>	Branch if (A) = (M)	IX+	71 rr	5	rfppp					
CBEQ <i>opr8,SP,rel</i>	Branch if (A) = (M)	SP1	9E 61 ff rr	6	prpppp					
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	--	--	--	--	0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	--	0	--	--	--
CLR <i>opr8a</i>	Clear M ← \$00	DIR	3F dd	5	rfwpp	0	--	0	1	--
CLRA	A ← \$00	INH	4F	1	p					
CLR X	X ← \$00	INH	5F	1	p					
CLR H	H ← \$00	INH	8C	1	p					
CLR <i>opr8,X</i>	M ← \$00	IX1	6F ff	5	rfwpp					
CLR <i>,X</i>	M ← \$00	IX	7F	4	rfwp					
CLR <i>opr8,SP</i>	M ← \$00	SP1	9E 6F ff	6	prfwpp					

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

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

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

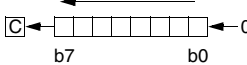
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
INC <i>opr8a</i> INCA INCX INC <i>opr8,X</i> INC ,X INC <i>opr8,SP</i>	Increment $M \leftarrow (M) + \$01$ $A \leftarrow (A) + \$01$ $X \leftarrow (X) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$	DIR INH INH IX1 IX SP1	3C dd 4C 5C 6C ff 7C 9E 6C ff	5 1 1 5 4 6	rwwpp p p rwwpp rwwp prwwpp					
JMP <i>opr8a</i> JMP <i>opr16a</i> JMP <i>opr16,X</i> JMP <i>opr8,X</i> JMP ,X	Jump $PC \leftarrow \text{Jump Address}$	DIR EXT IX2 IX1 IX	BC dd CC hh ll DC ee ff EC ff FC	3 4 4 3 3	ppp pppp pppp ppp ppp					
JSR <i>opr8a</i> JSR <i>opr16a</i> JSR <i>opr16,X</i> JSR <i>opr8,X</i> JSR ,X	Jump to Subroutine $PC \leftarrow (PC) + n$ ($n = 1, 2, \text{ or } 3$) Push (PCL); $SP \leftarrow (SP) - \$0001$ Push (PCH); $SP \leftarrow (SP) - \$0001$ $PC \leftarrow \text{Unconditional Address}$	DIR EXT IX2 IX1 IX	BD dd CD hh ll DD ee ff ED ff FD	5 6 6 5 5	ssppp psppp psppp ssppp ssppp					
LDA # <i>opr8i</i> LDA <i>opr8a</i> LDA <i>opr16a</i> LDA <i>opr16,X</i> LDA <i>opr8,X</i> LDA ,X LDA <i>opr16,SP</i> LDA <i>opr8,SP</i>	Load Accumulator from Memory $A \leftarrow (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 ii B6 dd C6 hh ll D6 ee ff E6 ff F6 9E D6 ee ff 9E E6 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rff pprpp prpp					
LDHX # <i>opr16i</i> LDHX <i>opr8a</i> LDHX <i>opr16a</i> LDHX ,X LDHX <i>opr16,X</i> LDHX <i>opr8,X</i> LDHX <i>opr8,SP</i>	Load Index Register (H:X) $H:X \leftarrow (M:M + \$0001)$	IMM DIR EXT IX IX2 IX1 SP1	45 jj kk 55 dd 32 hh ll 9E AE 9E BE ee ff 9E CE ff 9E FE ff	3 4 5 5 6 5 5	ppp rrpp prpp prffp pprrpp prppp prppp					
LDX # <i>opr8i</i> LDX <i>opr8a</i> LDX <i>opr16a</i> LDX <i>opr16,X</i> LDX <i>opr8,X</i> LDX ,X LDX <i>opr16,SP</i> LDX <i>opr8,SP</i>	Load X (Index Register Low) from Memory $X \leftarrow (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE ii BE dd CE hh ll DE ee ff EE ff FE 9E DE ee ff 9E EE ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rff pprpp prpp					
LSL <i>opr8a</i> LSLA LSLX LSL <i>opr8,X</i> LSL ,X LSL <i>opr8,SP</i>	Logical Shift Left  (Same as ASL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rwwpp p p rwwpp rwwp prwwpp					
LSR <i>opr8a</i> LSRA LSRX LSR <i>opr8,X</i> LSR ,X LSR <i>opr8,SP</i>	Logical Shift Right 	DIR INH INH IX1 IX SP1	34 dd 44 54 64 ff 74 9E 64 ff	5 1 1 5 4 6	rwwpp p p rwwpp rwwp prwwpp					

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

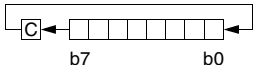
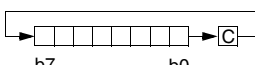
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR			
						VH	I	N	Z C
MOV <i>opr8a,opr8a</i> MOV <i>opr8a,X+</i> MOV <i>#opr8i,opr8a</i> MOV <i>,X+,opr8a</i>	Move $(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ In IX+/DIR and DIR/IX+ Modes, $H:X \leftarrow (H:X) + \$0001$	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E dd dd 5E dd 6E ii dd 7E dd	5 5 4 5	rpwpp rfwpp pwpp rfwpp	0-	-	↑	↓ -
MUL	Unsigned multiply $X:A \leftarrow (X) \times (A)$	INH	42	5	fffffp	-0	-	-	- 0
NEG <i>opr8a</i> NEGA NEGX NEG <i>opr8,X</i> NEG <i>,X</i> NEG <i>opr8,SP</i>	Negate $M \leftarrow -(M) = \$00 - (M)$ (Two's Complement) $A \leftarrow -(A) = \$00 - (A)$ $X \leftarrow -(X) = \$00 - (X)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$	DIR INH INH IX1 IX SP1	30 dd 40 50 60 ff 70 9E 60 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↑ ↓
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	--	-	-	- - -
NSA	Nibble Swap Accumulator $A \leftarrow (A[3:0]:A[7:4])$	INH	62	1	p	--	-	-	- - -
ORA <i>#opr8i</i> ORA <i>opr8a</i> ORA <i>opr16a</i> ORA <i>opr16,X</i> ORA <i>opr8,X</i> ORA <i>,X</i> ORA <i>opr16,SP</i> ORA <i>opr8,SP</i>	Inclusive OR Accumulator and Memory $A \leftarrow (A) \vee (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA ii BA dd CA hh ll DA ee ff EA ff FA 9E DA ee ff 9E EA ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0-	-	↑	↓ -
PSHA	Push Accumulator onto Stack $\text{Push } (A); SP \leftarrow (SP) - \0001	INH	87	2	sp	--	-	-	- - -
PSHH	Push H (Index Register High) onto Stack $\text{Push } (H); SP \leftarrow (SP) - \0001	INH	8B	2	sp	--	-	-	- - -
PSHX	Push X (Index Register Low) onto Stack $\text{Push } (X); SP \leftarrow (SP) - \0001	INH	89	2	sp	--	-	-	- - -
PULA	Pull Accumulator from Stack $SP \leftarrow (SP + \$0001); \text{Pull } (A)$	INH	86	3	ufp	--	-	-	- - -
PULH	Pull H (Index Register High) from Stack $SP \leftarrow (SP + \$0001); \text{Pull } (H)$	INH	8A	3	ufp	--	-	-	- - -
PULX	Pull X (Index Register Low) from Stack $SP \leftarrow (SP + \$0001); \text{Pull } (X)$	INH	88	3	ufp	--	-	-	- - -
ROL <i>opr8a</i> ROLA ROLX ROL <i>opr8,X</i> ROL <i>,X</i> ROL <i>opr8,SP</i>	Rotate Left through Carry 	DIR INH INH IX1 IX SP1	39 dd 49 59 69 ff 79 9E 69 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↑ ↓
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry 	DIR INH INH IX1 IX SP1	36 dd 46 56 66 ff 76 9E 66 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↑ ↓

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

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

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

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

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
TXS	Transfer Index Reg. to SP $SP \leftarrow (H:X) - \$0001$	INH	94	2	f p	--	--	--	--	--
WAIT	Enable Interrupts; Wait for Interrupt $I \text{ bit} \leftarrow 0$; Halt CPU	INH	8F	2+	f p . . .	--	0	--	--	--

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

- n* Any label or expression that evaluates to a single integer in the range 0-7.
- opr8i* Any label or expression that evaluates to an 8-bit immediate value.
- opr16i* Any label or expression that evaluates to a 16-bit immediate value.
- opr8a* Any label or expression that evaluates to an 8-bit direct-page address (\$00xx).
- opr16a* Any label or expression that evaluates to a 16-bit address.
- opr8* Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.
- opr16* Any label or expression that evaluates to a 16-bit value, used for indexed addressing.
- rel* Any label or expression that refers to an address that is within -128 to +127 locations from the start of the next instruction.

Operation Symbols:

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

CCR Bits:

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

Addressing Modes:

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

Cycle-by-Cycle Codes:

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

CCR Effects:

- ↑ Set or cleared
- Not affected
- U Undefined

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

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

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

Opcode in Hexadecimal F0 SUB 3
 Number of Bytes 1 SUB IX
 HCS08 Cycles Instruction Mnemonic Addressing Mode

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

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

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

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

Prebyte (9E) and Opcode in
 Hexadecimal 9E60 6
 3 SP1
 Number of Bytes 3 NEG HCS08 Cycles
 SP1 Instruction Mnemonic
 Addressing Mode

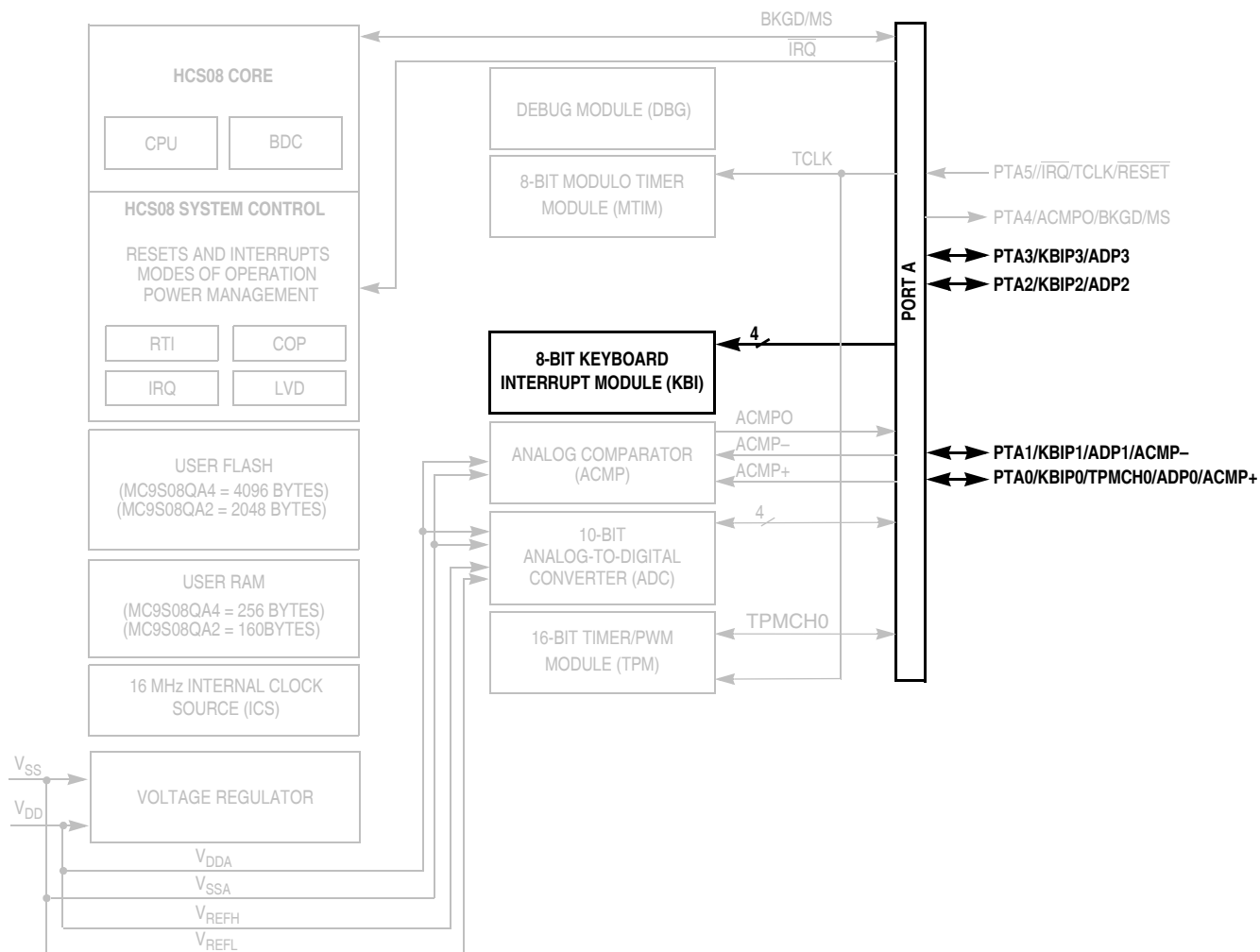
Chapter 8

Keyboard Interrupt (S08KBIV2)

8.1 Introduction

The keyboard interrupt KBI module provides up to four independently enabled external interrupt sources. KBIP4-KBIP7 are not available in the MC9S08QA4 series.

[Figure 8-1](#) shows the MC9S08QA4 series block diagram with the KBI highlighted.



NOTES:

- ¹ Port pins are software configurable with pullup device if input port.
- ² Port pins are software configurable for output drive strength.
- ³ Port pins are software configurable for output slew rate control.
- ⁴ $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- ⁵ $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- ⁶ PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
- ⁷ When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 8-1. MC9S08QA4 Series Block Diagram Highlighting KBI Block and Pins

8.1.1 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits.
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
- One software enabled keyboard interrupt.
- Exit from low-power modes.

8.1.2 Modes of Operation

This section defines the KBI operation in wait, stop, and background debug modes.

8.1.2.1 KBI in Wait Mode

The KBI continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, an enabled KBI pin ($KBPE_x = 1$) can be used to bring the MCU out of wait mode if the KBI interrupt is enabled ($KBIE = 1$).

8.1.2.2 KBI in Stop Modes

The KBI operates asynchronously in stop3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin ($KBPE_x = 1$) can be used to bring the MCU out of stop3 mode if the KBI interrupt is enabled ($KBIE = 1$).

During either stop1 or stop2 mode, the KBI is disabled. In some systems, the pins associated with the KBI may be sources of wakeup from stop1 or stop2, see the stop modes section in the [Modes of Operation](#) chapter. Upon wake-up from stop1 or stop2 mode, the KBI module will be in the reset state.

8.1.2.3 KBI in Active Background Mode

When the microcontroller is in active background mode, the KBI will continue to operate normally.

8.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown [Figure 8-2](#).

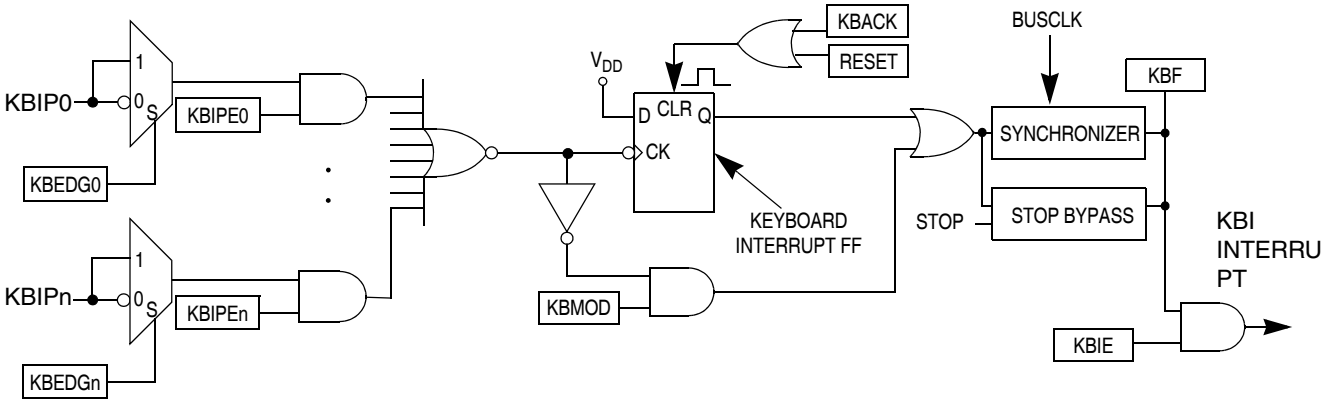


Figure 8-2. KBI Block Diagram

8.2 External Signal Description

The KBI input pins can be used to detect either falling edges, or both falling edge and low level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high level interrupt requests.

The signal properties of KBI are shown in [Table 8-1](#).

Table 8-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

8.3 Register Definition

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

Refer to the direct-page register summary in the [Memory](#) chapter for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

Some MCUs may have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced.

8.3.1 KBI Status and Control Register (KBISC)

KBISC contains the status flag and control bits, which are used to configure the KBI.

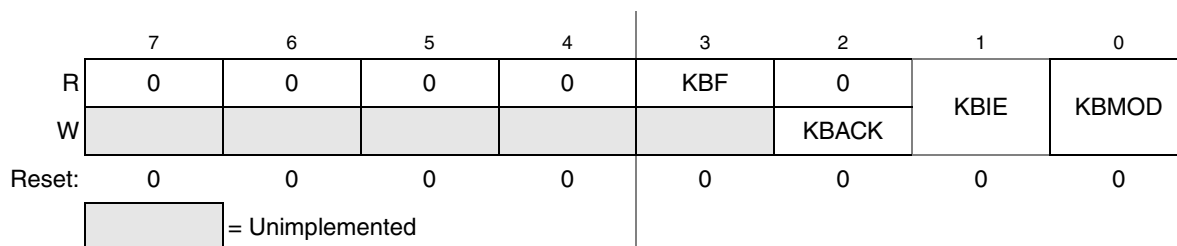


Figure 8-3. KBI Status and Control Register

Table 8-2. KBISC Register Field Descriptions

Field	Description
7:4	Unused register bits, always read 0.
3 KBF	Keyboard Interrupt Flag — KBF indicates when a keyboard interrupt is detected. Writes have no effect on KBF. 0 No keyboard interrupt detected. 1 Keyboard interrupt detected.
2 KBACK	Keyboard Acknowledge — Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	Keyboard Interrupt Enable — KBIE determines whether a keyboard interrupt is requested. 0 Keyboard interrupt request not enabled. 1 Keyboard interrupt request enabled.
0 KBMOD	Keyboard Detection Mode — KBMOD (along with the KBEDG bits) controls the detection mode of the keyboard interrupt pins. 0 Keyboard detects edges only. 1 Keyboard detects both edges and levels.

8.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.

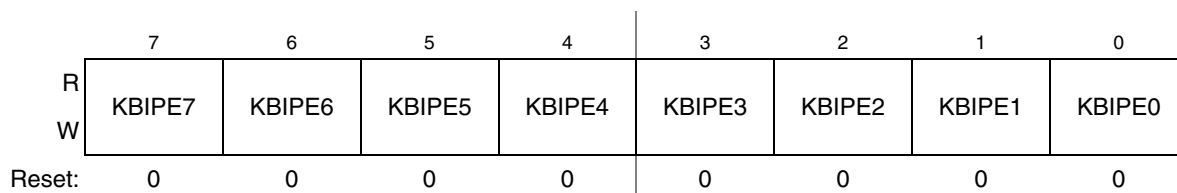


Figure 8-4. KBI Pin Enable Register

Table 8-3. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPE _n	Keyboard Pin Enables — Each of the KBIPE _n bits enable the corresponding keyboard interrupt pin. 0 Pin not enabled as keyboard interrupt. 1 Pin enabled as keyboard interrupt.

8.3.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

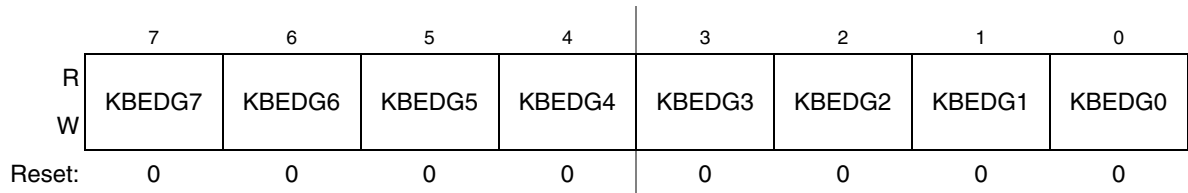


Figure 8-5. KBI Edge Select Register

Table 8-4. KBIES Register Field Descriptions

Field	Description
7:0 KBEDGn	Keyboard Edge Selects — Each of the KBEDGn bits selects the falling edge/low level or rising edge/high level function of the corresponding pin). 0 Falling edge/low level. 1 Rising edge/high level.

8.4 Functional Description

This on-chip peripheral module is called a keyboard interrupt (KBI) module because originally it was designed to simplify the connection and use of row-column matrices of keyboard switches. However, these inputs are also useful as extra external interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes.

The KBI module allows up to eight pins to act as additional interrupt sources. Writing to the KBIPEn bits in the keyboard interrupt pin enable register (KBIPE) independently enables or disables each KBI pin. Each KBI pin can be configured as edge sensitive or edge and level sensitive based on the KBMOD bit in the keyboard interrupt status and control register (KBISC). Edge sensitive can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the KBEDGn bits in the keyboard interrupt edge select register (KBIES).

8.4.1 Edge Only Sensitivity

Synchronous logic is used to detect edges. A falling edge is detected when an enabled keyboard interrupt (KBIPEn=1) input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 (the deasserted level) during one bus cycle and then a logic 1 (the asserted level) during the next cycle. Before the first edge is detected, all enabled keyboard interrupt input signals must be at the deasserted logic levels. After any edge is detected, all enabled keyboard interrupt input signals must return to the deasserted level before any new edge can be detected.

A valid edge on an enabled KBI pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBISC.

8.4.2 Edge and Level Sensitivity

A valid edge or level on an enabled KBI pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in

KBISC provided all enabled keyboard inputs are at their deasserted levels. KBF will remain set if any enabled KBI pin is asserted while attempting to clear by writing a 1 to KBACK.

8.4.3 KBI Pullup/Pulldown Resistors

The KBI pins can be configured to use an internal pullup/pulldown resistor using the associated I/O port pullup enable register. If an internal resistor is enabled, the KBIES register is used to select whether the resistor is a pullup ($KBEDGn = 0$) or a pulldown ($KBEDGn = 1$).

8.4.4 KBI Initialization

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

1. Mask keyboard interrupts by clearing KBIE in KBISC.
2. Enable the KBI polarity by setting the appropriate KBEDGn bits in KBIES.
3. If using internal pullup/pulldown device, configure the associated pullup enable bits in PTxPE.
4. Enable the KBI pins by setting the appropriate KBIPEn bits in KBIPE.
5. Write to KBACK in KBISC to clear any false interrupts.
6. Set KBIE in KBISC to enable interrupts.



Chapter 9

Analog Comparator (S08ACMPV2)

9.1 Introduction

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages or for comparing one analog input voltage to an internal reference voltage. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

[Figure 9-1](#) shows the MC9S08QA4 series block diagram with the ACMP highlighted.

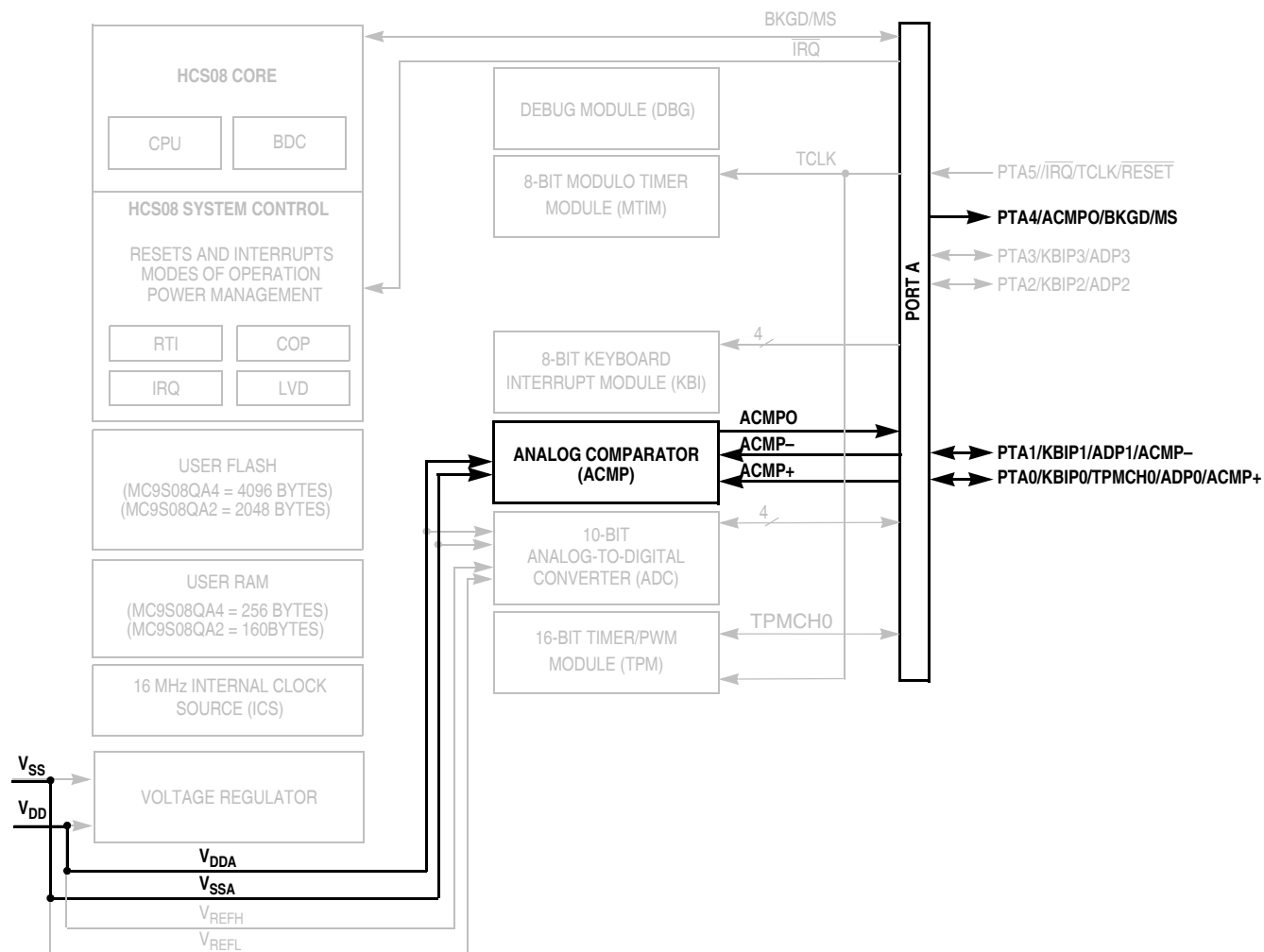
9.1.1 ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP+, the user must enable the bandgap buffer by setting BGBE =1 in SPMSC1; see [Section 5.8.8, “System Power Management Status and Control 1 Register \(SPMSC1\)”](#). For the value of the bandgap voltage reference see MC9S08QA4 Series *Data Sheet*.

To use ACMPO, the BKGDPE bit in SOPT1 must be cleared. This will disable the background debug mode and on-chip ICE.

9.1.2 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.



NOTES:

- 1 Port pins are software configurable with pullup device if input port.
- 2 Port pins are software configurable for output drive strength.
- 3 Port pins are software configurable for output slew rate control.
- 4 $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- 5 $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- 6 PTA4 contains integrated pullup device if BKGD enabled (BKGDPPE = 1).
- 7 When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 9-1. MC9S08QA4 Series Block Diagram Highlighting ACMP Block and Pins

9.1.3 Features

The ACMP has the following features:

- Full rail-to-rail supply operation.
- Less than 40 mV of input offset.
- Less than 15 mV of hysteresis.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Option to compare to fixed internal bandgap reference voltage.
- Option to allow comparator output to be visible on a pin, ACMPO.

9.1.4 Modes of Operation

This section defines the ACMP operation in wait, stop, and background debug modes.

9.1.4.1 ACMP in Wait Mode

The ACMP continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the ACMP can be used to bring the MCU out of wait mode if the ACMP interrupt, ACIE, is enabled. For lowest possible current consumption, the ACMP should be disabled by software if not required as an interrupt source during wait mode.

9.1.4.2 ACMP in Stop Modes

The ACMP is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the ACMP cannot be used as a wake up source from stop modes.

During either stop1 or stop2 mode, the ACMP module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the ACMP module will be in the reset state.

During stop3 mode, clocks to the ACMP module are halted. No registers are affected. In addition, the ACMP comparator circuit will enter a low power state. No compare operation will occur while in stop3.

If stop3 is exited with a reset, the ACMP will be put into its reset state. If stop3 is exited with an interrupt, the ACMP continues from the state it was in when stop3 was entered.

9.1.4.3 ACMP in Active Background Mode

When the microcontroller is in active background mode, the ACMP will continue to operate normally.

9.1.5 Block Diagram

The block diagram for the analog comparator module is shown [Figure 9-2](#).

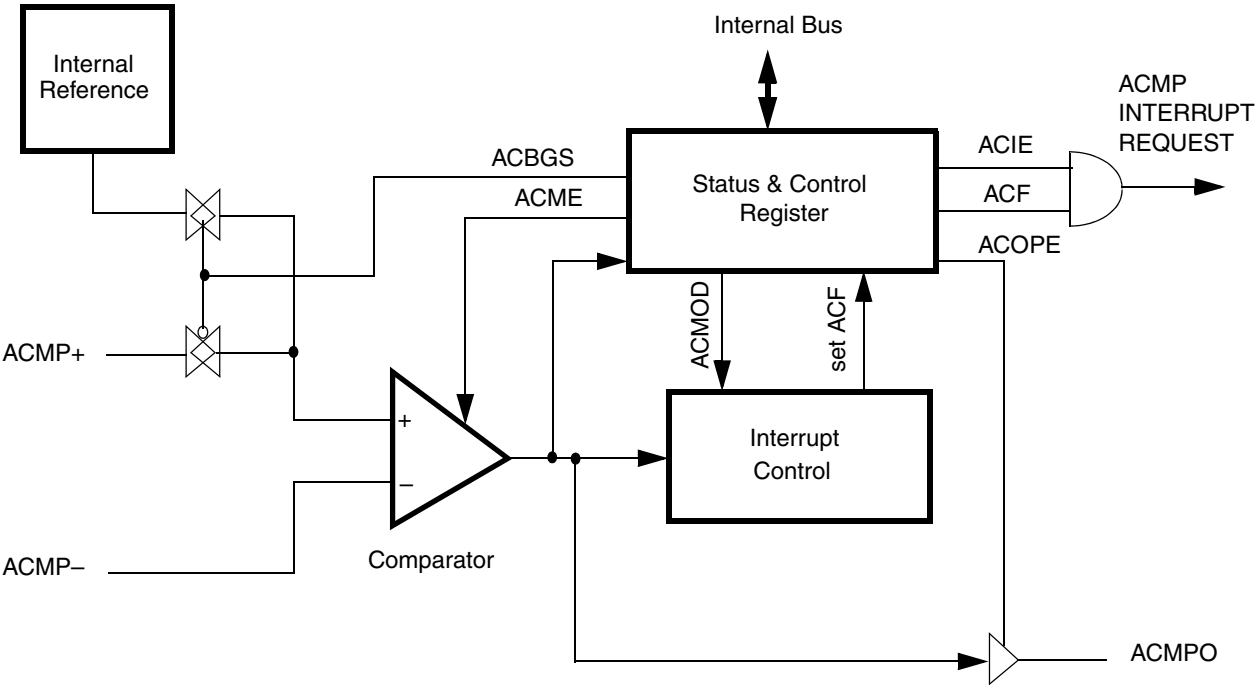


Figure 9-2. Analog Comparator (ACMP) Block Diagram

9.2 External Signal Description

The ACMP has two analog input pins, ACMP+ and ACMP– and one digital output pin ACMPO. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. As shown in [Figure 9-2](#), the ACMP– pin is connected to the inverting input of the comparator, and the ACMP+ pin is connected to the comparator non-inverting input if ACBGS is a 0. As shown in [Figure 9-2](#), the ACMPO pin can be enabled to drive an external pin.

The signal properties of ACMP are shown in [Table 9-1](#).

Table 9-1. Signal Properties

Signal	Function	I/O
ACMP–	Inverting analog input to the ACMP. (Minus input)	I
ACMP+	Non-inverting analog input to the ACMP. (Positive input)	I
ACMPO	Digital output of the ACMP.	O

9.3 Register Definition

The ACMP includes one register:

- An 8-bit status and control register

Refer to the direct-page register summary in the memory section of this data sheet for the absolute address assignments for all ACMP registers. This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one ACMP, so register names include placeholder characters to identify which ACMP is being referenced.

9.3.1 ACMP Status and Control Register (ACMPSC)

ACMPSC contains the status flag and control bits which are used to enable and configure the ACMP.

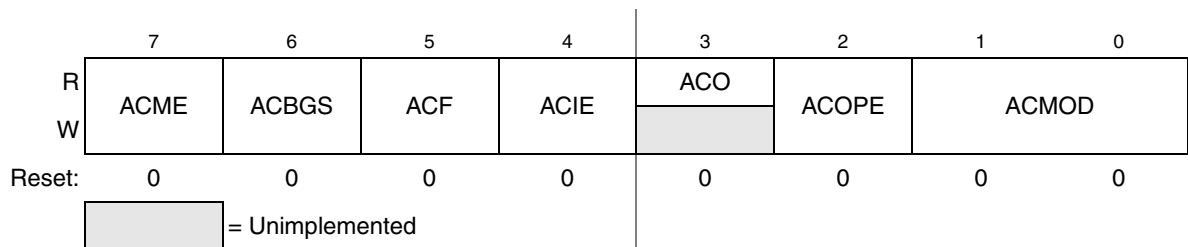


Figure 9-3. ACMP Status and Control Register

Table 9-2. ACMP Status and Control Register Field Descriptions

Field	Description
7 ACME	Analog Comparator Module Enable — ACME enables the ACMP module. 0 ACMP not enabled 1 ACMP is enabled
6 ACBGS	Analog Comparator Bandgap Select — ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input to the non-inverting input of the analog comparatorr. 0 External pin ACMP+ selected as non-inverting input to comparator 1 Internal reference select as non-inverting input to comparator
5 ACF	Analog Comparator Flag — ACF is set when a compare event occurs. Compare events are defined by ACMOD. ACF is cleared by writing a one to ACF. 0 Compare event has not occurred 1 Compare event has occurred
4 ACIE	Analog Comparator Interrupt Enable — ACIE enables the interrupt from the ACMP. When ACIE is set, an interrupt will be asserted when ACF is set. 0 Interrupt disabled 1 Interrupt enabled
3 ACO	Analog Comparator Output — Reading ACO will return the current value of the analog comparator output. ACO is reset to a 0 and will read as a 0 when the ACMP is disabled (ACME = 0).
2 ACOPE	Analog Comparator Output Pin Enable — ACOPE is used to enable the comparator output to be placed onto the external pin, ACMPO. 0 Analog comparator output not available on ACMPO 1 Analog comparator output is driven out on ACMPO
1:0 ACMOD	Analog Comparator Mode — ACMOD selects the type of compare event which sets ACF. 00 Encoding 0 — Comparator output falling edge 01 Encoding 1 — Comparator output rising edge 10 Encoding 2 — Comparator output falling edge 11 Encoding 3 — Comparator output rising or falling edge

9.4 Functional Description

The analog comparator can be used to compare two analog input voltages applied to ACMP+ and ACMP–; or it can be used to compare an analog input voltage applied to ACMP– with an internal bandgap reference voltage. ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input to the non-inverting input of the analog comparator. The comparator output is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. ACMOD is used to select the condition which will cause ACF to be set. ACF can be set on a rising edge of the comparator output, a falling edge of the comparator output, or either a rising or a falling edge (toggle). The comparator output can be read directly through ACO. The comparator output can be driven onto the ACMPO pin using ACOPE.



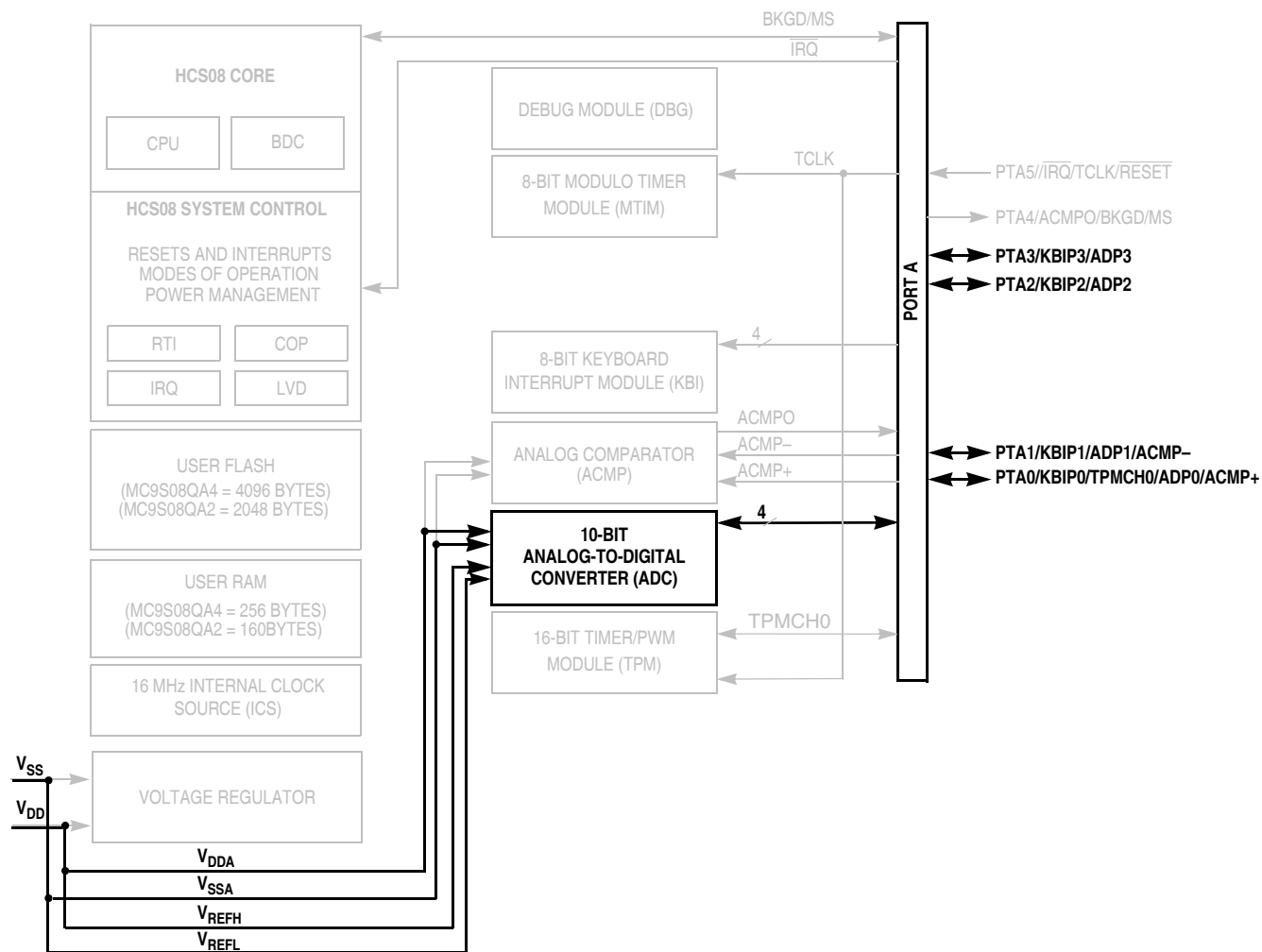
Chapter 10

Analog-to-Digital Converter (S08ADC10V1)

10.1 Introduction

The 10-bit analog-to-digital converter (ADC) is a successive-approximation ADC designed for operation within an integrated microcontroller system-on-chip.

[Figure 10-1](#) shows the MC9S08QA4 series block diagram with the ADC module and pins highlighted.



NOTES:

- 1 Port pins are software configurable with pullup device if input port.
- 2 Port pins are software configurable for output drive strength.
- 3 Port pins are software configurable for output slew rate control.
- 4 $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- 5 $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- 6 PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
- 7 When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 10-1. MC9S08QA4 Series Block Diagram Highlighting ADC Block and Pins

10.1.1 Module Configurations

This section provides device-specific information for configuring the ADC on the MC9S08QA4 series of devices.

10.1.1.1 Analog Supply and Voltage Reference Connections

The V_{DDAD} and V_{REFH} sources for the ADC are internally connected to the V_{DD} pin. The V_{SSAD} and V_{REFL} sources for the ADC are internally connected to the V_{SS} pin.

10.1.1.2 Channel Assignments

The ADC channel assignments for the MC9S08QA4 series devices are shown in [Table 10-1](#). Reserved channels convert to an unknown value.

Table 10-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	AD0	PTA0/ADP0	ADPC0	10000	AD16	V_{SS}	N/A
00001	AD1	PTA1/ADP1	ADPC1	10001	AD17	V_{SS}	N/A
00010	AD2	PTA2/ADP2	ADPC2	10010	AD18	V_{SS}	N/A
00011	AD3	PTA3/ADP3	ADPC3	10011	AD19	V_{SS}	N/A
00100	AD4	Reserved	ADPC4	10100	AD20	V_{SS}	N/A
00101	AD5	Reserved	ADPC5	10101	AD21	V_{SS}	N/A
00110	AD6	Reserved	ADPC6	10110	AD22	Reserved	N/A
00111	AD7	Reserved	ADPC7	10111	AD23	Reserved	N/A
01000	AD8	V_{SS}	N/A	11000	AD24	Reserved	N/A
01001	AD9	V_{SS}	N/A	11001	AD25	Reserved	N/A
01010	AD10	V_{SS}	N/A	11010	AD26	Temperature Sensor ¹	N/A
01011	AD11	V_{SS}	N/A	11011	AD27	Internal Bandgap	N/A
01100	AD12	V_{SS}	N/A	11100	—	Reserved	N/A
01101	AD13	V_{SS}	N/A	11101	V_{REFH}	V_{DD}	N/A
01110	AD14	V_{SS}	N/A	11110	V_{REFL}	V_{SS}	N/A
01111	AD15	V_{SS}	N/A	11111	Module Disabled	None	N/A

¹ For information, see [Section 10.1.1.6](#), “Temperature Sensor.”

NOTE

Selecting the internal bandgap channel requires $BGBE = 1$ in $SPMSC1$; see [Section 5.8.8](#), “System Power Management Status and Control 1 Register ($SPMSC1$).” For the value of the bandgap voltage reference see *MC9S08QA4 Series Data Sheet*.

10.1.1.3 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, or the local asynchronous clock (ADACK) within the module. The alternate clock (ALTCLK) input for the MC9S08QA4 series MCU devices is not implemented.

10.1.1.4 Hardware Trigger

The ADC hardware trigger, ADHWT, is output from the real-time interrupt (RTI) counter. The RTI counter can be clocked by ICSERCLK or a nominal 1 kHz clock source within the RTI block.

The period of the RTI is determined by the input clock frequency and the RTIS bits. The RTI counter is a free running counter that generates an overflow at the RTI rate determined by the RTIS bits. When the ADC hardware trigger is enabled, a conversion is initiated upon an RTI counter overflow.

The RTI can be configured to cause a hardware trigger in MCU run, wait, and stop3.

10.1.1.5 Analog Pin Enables

The ADC on MC9S08QA4 series devices contains only one analog pin enable register, APCTL1.

10.1.1.6 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 10-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{\text{TEMP}} - V_{\text{TEMP25}}) \div m) \quad \text{Eqn. 10-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values from MC9S08QA4 Series *Data Sheet*.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP} , and compares to V_{TEMP25} . If V_{TEMP} is greater than V_{TEMP25} , the cold slope value is applied in Equation 10-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 10-1.

10.1.1.7 Low-Power Mode Operation

The ADC is capable of running in stop3 mode but requires LVDSE and LVDE in SPMSC1 to be set.

10.1.2 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 10 bits resolution.
- Up to 28 analog inputs.
- Output formatted in 10- or 8-bit right-justified format.
- Single or continuous conversion (automatic return to idle after single conversion).
- Configurable sample time and conversion speed/power.
- Conversion complete flag and interrupt.
- Input clock selectable from up to four sources.
- Operation in wait or stop3 modes for lower noise operation.
- Asynchronous clock source for lower noise operation.
- Selectable asynchronous hardware conversion trigger.
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value.

10.1.3 Block Diagram

Figure 10-2 provides a block diagram of the ADC module

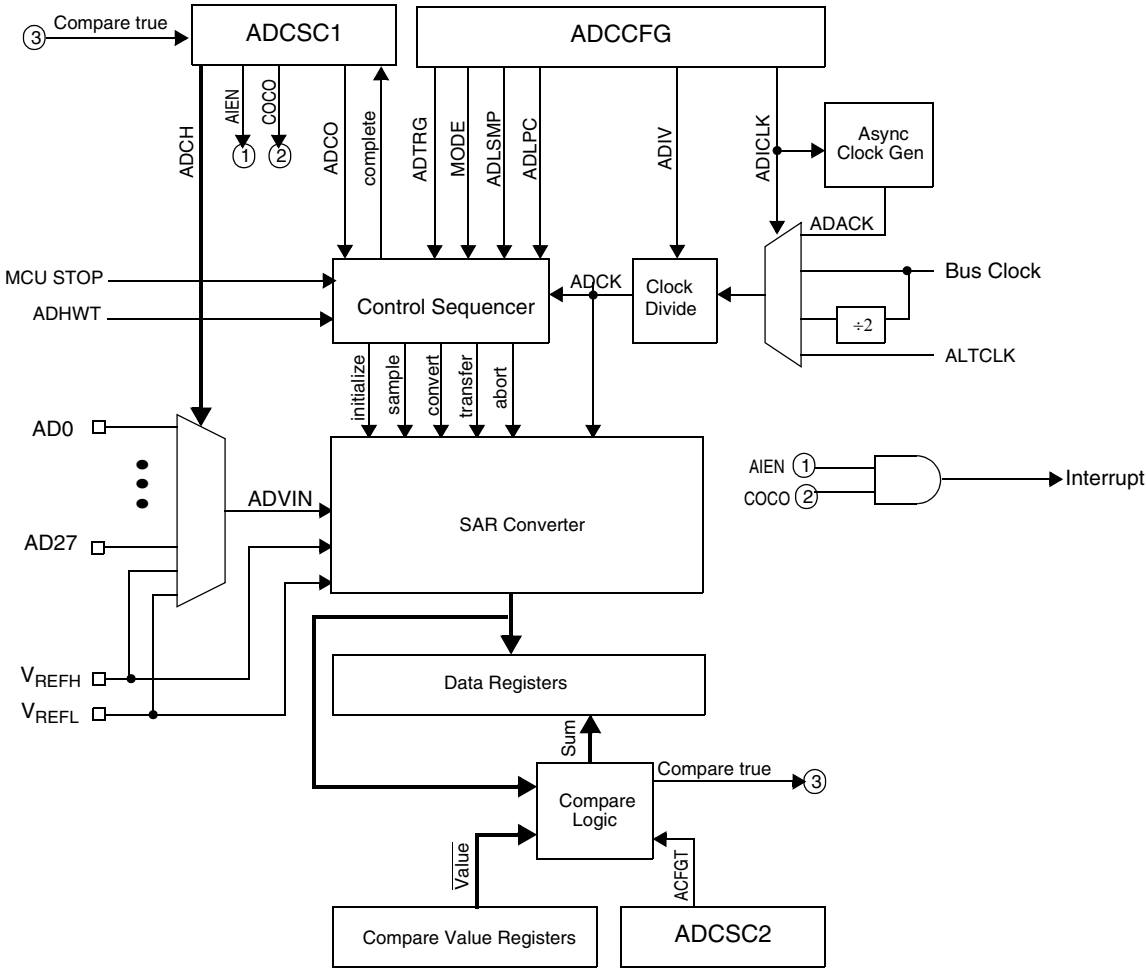


Figure 10-2. ADC Block Diagram

10.2 External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

Table 10-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V _{REFH}	High reference voltage
V _{REFL}	Low reference voltage
V _{DDAD}	Analog power supply
V _{SSAD}	Analog ground

10.2.1 Analog Power (V_{DDAD})

The ADC analog portion uses V_{DDAD} as its power connection. In some packages, V_{DDAD} is connected internally to V_{DD} . If externally available, connect the V_{DDAD} pin to the same voltage potential as V_{DD} . External filtering may be necessary to ensure clean V_{DDAD} for good results.

10.2.2 Analog Ground (V_{SSAD})

The ADC analog portion uses V_{SSAD} as its ground connection. In some packages, V_{SSAD} is connected internally to V_{SS} . If externally available, connect the V_{SSAD} pin to the same voltage potential as V_{SS} .

10.2.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDAD} . If externally available, V_{REFH} may be connected to the same potential as V_{DDAD} , or may be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}).

10.2.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSAD} . If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSAD} .

10.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

10.3 Register Definition

These memory mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin enable registers, APCTL1, APCTL2, APCTL3

10.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

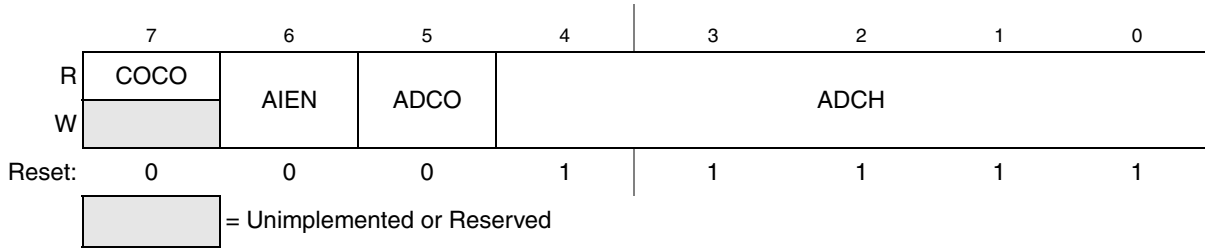


Figure 10-3. Status and Control Register (ADCSC1)

Table 10-3. ADCSC1 Register Field Descriptions

Field	Description
7 COCO	Conversion Complete Flag — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADCSC1 is written or whenever ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	Interrupt Enable — AIEN is used to enable conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	Continuous Conversion Enable — ADCO is used to enable continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	Input Channel Select — The ADCH bits form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Figure 10-4 . The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Figure 10-4. Input Channel Select

ADCH	Input Select	ADCH	Input Select
00000	AD0	10000	AD16
00001	AD1	10001	AD17
00010	AD2	10010	AD18
00011	AD3	10011	AD19
00100	AD4	10100	AD20
00101	AD5	10101	AD21
00110	AD6	10110	AD22

Figure 10-4. Input Channel Select (continued)

ADCH	Input Select	ADCH	Input Select
00111	AD7	10111	AD23
01000	AD8	11000	AD24
01001	AD9	11001	AD25
01010	AD10	11010	AD26
01011	AD11	11011	AD27
01100	AD12	11100	Reserved
01101	AD13	11101	V _{REFH}
01110	AD14	11110	V _{REFL}
01111	AD15	11111	Module disabled

10.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register is used to control the compare function, conversion trigger and conversion active of the ADC module.



¹ Bits 1 and 0 are reserved bits that must always be written to 0.

Figure 10-5. Status and Control Register 2 (ADCSC2)

Table 10-4. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	Conversion Active — ADACT indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	Conversion Trigger Select — ADTRG is used to select the type of trigger to be used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected

Table 10-4. ADCSC2 Register Field Descriptions (continued)

Field	Description
5 ACFE	Compare Function Enable — ACFE is used to enable the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	Compare Function Greater Than Enable — ACFGT is used to configure the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare level 1 Compare triggers when input is greater than or equal to compare level

10.3.3 Data Result High Register (ADCRH)

ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 8-bit conversions both ADR8 and ADR9 are equal to zero. ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 10-bit MODE, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode there is no interlocking with ADCRL. In the case that the MODE bits are changed, any data in ADCRH becomes invalid.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	ADR9	ADR8
W								
Reset:	0	0	0	0	0	0	0	0
	= Unimplemented or Reserved							

Figure 10-6. Data Result High Register (ADCRH)

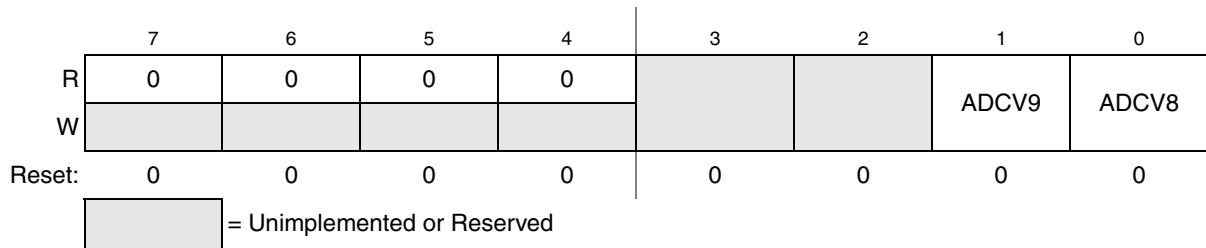
10.3.4 Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of the result of a 10-bit conversion, and all eight bits of an 8-bit conversion. This register is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until the after next conversion is completed, then the intermediate conversion results will be lost. In 8-bit mode, there is no interlocking with ADCRH. In the case that the MODE bits are changed, any data in ADCRL becomes invalid.


Figure 10-7. Data Result Low Register (ADCRL)

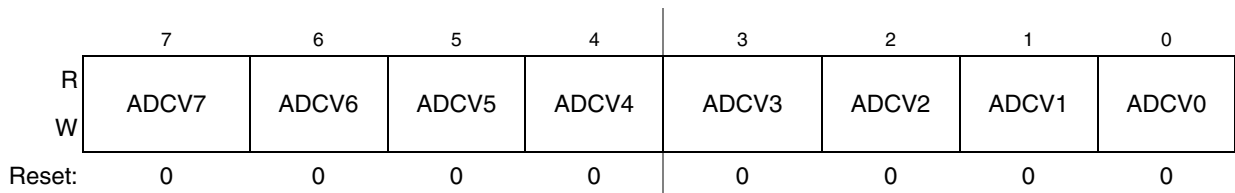
10.3.5 Compare Value High Register (ADCCVH)

This register holds the upper two bits of the 10-bit compare value. These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled. In 8-bit operation, ADCCVH is not used during compare.


Figure 10-8. Compare Value High Register (ADCCVH)

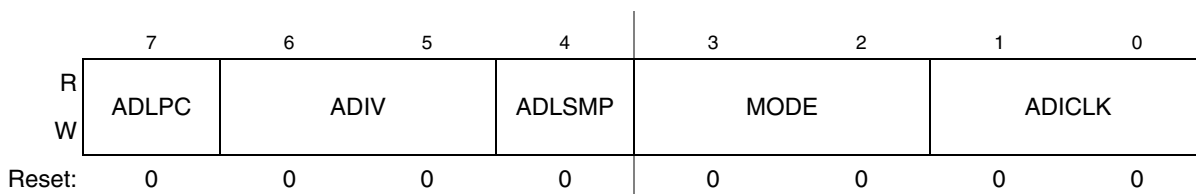
10.3.6 Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 10-bit compare value, or all 8 bits of the 8-bit compare value. Bits ADCV7:ADCV0 are compared to the lower 8 bits of the result following a conversion in either 10-bit or 8-bit mode.


Figure 10-9. Compare Value Low Register(ADCCVL)

10.3.7 Configuration Register (ADCCFG)

ADCCFG is used to select the mode of operation, clock source, clock divide, and configure for low power or long sample time.


Figure 10-10. Configuration Register (ADCCFG)
Table 10-5. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	Low Power Configuration — ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: {FC31}The power is reduced at the expense of maximum clock speed.
6:5 ADIV	Clock Divide Select — ADIV select the divide ratio used by the ADC to generate the internal clock ADCK. Table 10-6 shows the available clock configurations.
4 ADLSMP	Long Sample Time Configuration — ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	Conversion Mode Selection — MODE bits are used to select between 10- or 8-bit operation. See Table 10-7 .
1:0 ADICLK	Input Clock Select — ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 10-8 .

Table 10-6. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 10-7. Conversion Modes

MODE	Mode Description
00	8-bit conversion (N=8)
01	Reserved
10	10-bit conversion (N=10)
11	Reserved

Table 10-8. Input Clock Select

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

10.3.8 Pin Control 1 Register (APCTL1)

The pin control registers are used to disable the I/O port control of MCU pins used as analog inputs. APCTL1 is used to control the pins associated with channels 0–7 of the ADC module.

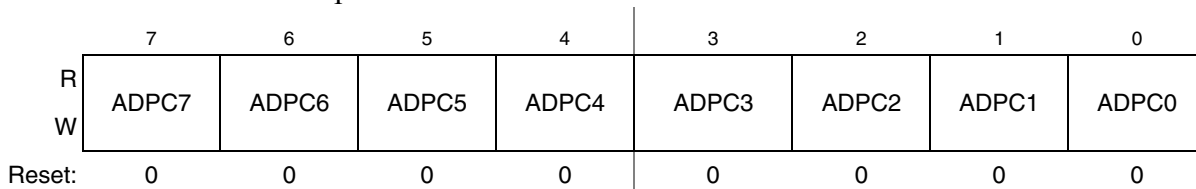


Figure 10-11. Pin Control 1 Register (APCTL1)

Table 10-9. APCTL1 Register Field Descriptions

Field	Description
7 ADPC7	ADC Pin Control 7 — ADPC7 is used to control the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	ADC Pin Control 6 — ADPC6 is used to control the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	ADC Pin Control 5 — ADPC5 is used to control the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	ADC Pin Control 4 — ADPC4 is used to control the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	ADC Pin Control 3 — ADPC3 is used to control the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	ADC Pin Control 2 — ADPC2 is used to control the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled

Table 10-9. APCTL1 Register Field Descriptions (continued)

Field	Description
1 ADPC1	ADC Pin Control 1 — ADPC1 is used to control the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	ADC Pin Control 0 — ADPC0 is used to control the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

10.3.9 Pin Control 2 Register (APCTL2)

APCTL2 is used to control channels 8–15 of the ADC module.

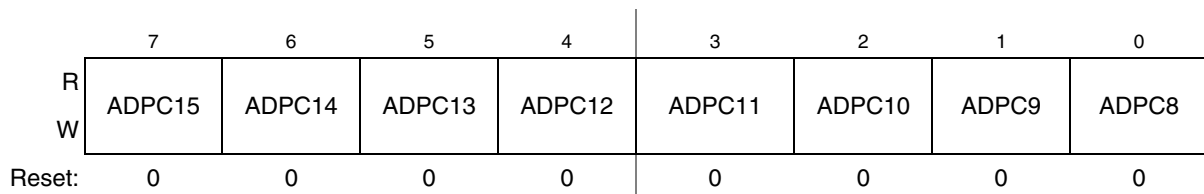


Figure 10-12. Pin Control 2 Register (APCTL2)

Table 10-10. APCTL2 Register Field Descriptions

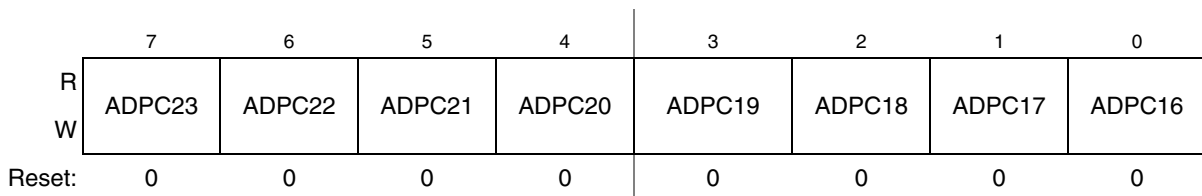
Field	Description
7 ADPC15	ADC Pin Control 15 — ADPC15 is used to control the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	ADC Pin Control 14 — ADPC14 is used to control the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	ADC Pin Control 13 — ADPC13 is used to control the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	ADC Pin Control 12 — ADPC12 is used to control the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	ADC Pin Control 11 — ADPC11 is used to control the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	ADC Pin Control 10 — ADPC10 is used to control the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled

Table 10-10. APCTL2 Register Field Descriptions (continued)

Field	Description
1 ADPC9	ADC Pin Control 9 — ADPC9 is used to control the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	ADC Pin Control 8 — ADPC8 is used to control the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

10.3.10 Pin Control 3 Register (APCTL3)

APCTL3 is used to control channels 16–23 of the ADC module.


Figure 10-13. Pin Control 3 Register (APCTL3)
Table 10-11. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	ADC Pin Control 23 — ADPC23 is used to control the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	ADC Pin Control 22 — ADPC22 is used to control the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	ADC Pin Control 21 — ADPC21 is used to control the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	ADC Pin Control 20 — ADPC20 is used to control the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	ADC Pin Control 19 — ADPC19 is used to control the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	ADC Pin Control 18 — ADPC18 is used to control the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled

Table 10-11. APCTL3 Register Field Descriptions (continued)

Field	Description
1 ADPC17	ADC Pin Control 17 — ADPC17 is used to control the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	ADC Pin Control 16 — ADPC16 is used to control the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

10.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. The selected channel voltage is converted by a successive approximation algorithm into an 11-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). In 10-bit mode, the result is rounded to 10 bits and placed in ADCRH and ADCRL. In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ACFE bit and operates in conjunction with any of the conversion modes and configurations.

10.4.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by 2. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK) – This clock is generated from a clock source within the ADC module. When selected as the clock source this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC will not perform according to specifications. If the available clocks

are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

10.4.2 Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) are used to disable the I/O port control of the pins used as analog inputs. When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

10.4.3 Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

10.4.4 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

10.4.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

10.4.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRH and ADCRL. This is indicated by the setting of COCO. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 10-bit MODE (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

10.4.4.3 Aborting Conversions

Any conversion in progress will be aborted when:

- A write to ADCSC1 occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCSC2, ADCCFG, ADCCVH, or ADCCVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL, are not altered but continue to be the values transferred after the completion of the last successful conversion. In the case that the conversion was aborted by a reset, ADCRH and ADCRL return to their reset states.

10.4.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADLPC. This results in a lower maximum value for f_{ADCK} (see the electrical specifications).

10.4.4.5 Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit or 10-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP is used to select between short and long sample times. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The

result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in [Table 10-12](#).

Table 10-12. Total Conversion Time vs. Control Conditions

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μ s + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	0	5 μ s + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μ s + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	1	5 μ s + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. For example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion is:

$$\text{Conversion time} = \frac{23 \text{ ADCK cyc}}{8 \text{ MHz}/1} + \frac{5 \text{ bus cyc}}{8 \text{ MHz}} = 3.5 \mu\text{s}$$

$$\text{Number of bus cycles} = 3.5 \mu\text{s} \times 8 \text{ MHz} = 28 \text{ cycles}$$

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

10.4.5 Automatic Compare Function

The compare function can be configured to check for either an upper limit or lower limit. After the input is sampled and converted, the result is added to the two's complement of the compare value (ADCCVH and ADCCVL). When comparing to an upper limit (ACFGT = 1), if the result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. The value generated by the addition of the conversion result and the two's complement of the compare value is transferred to ADCRH and ADCRL.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

NOTE

The compare function can be used to monitor the voltage on a channel while the MCU is in either wait or stop3 mode. The ADC interrupt will wake the MCU when the compare condition is met.

10.4.6 MCU Wait Mode Operation

The WAIT instruction puts the MCU in a lower power-consumption standby mode from which recovery is very fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (AIEN = 1).

10.4.7 MCU Stop3 Mode Operation

The STOP instruction is used to put the MCU in a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

10.4.7.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a STOP instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

10.4.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in [Section 10.4.4.2, "Completing Conversions"](#)) is cleared when entering stop3 and continuing ADC conversions.

10.4.8 MCU Stop1 and Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters either stop1 or stop2 mode. All module registers contain their reset values following exit from stop1 or stop2. Therefore the module must be re-enabled and re-configured following exit from stop1 or stop2.

10.5 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to [Table 10-6](#), [Table 10-7](#), and [Table 10-8](#) for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

10.5.1 ADC Module Initialization Example

10.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.

2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

10.5.1.2 Pseudo — Code Example

In this example, the ADC module will be set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock will be derived from the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

Bit 7	ADLPC	1	Configures for low power (lowers maximum clock speed)
Bit 6:5	ADIV	00	Sets the ADCK to the input clock ÷ 1
Bit 4	ADLSMP	1	Configures for long sample time
Bit 3:2	MODE	10	Sets mode at 10-bit conversions
Bit 1:0	ADICLK	00	Selects bus clock as input clock source

ADCSC2 = 0x00 (%00000000)

Bit 7	ADACT	0	Flag indicates if a conversion is in progress
Bit 6	ADTRG	0	Software trigger selected
Bit 5	ACFE	0	Compare function disabled
Bit 4	ACFGT	0	Not used in this example
Bit 3:2		00	Unimplemented or reserved, always reads zero
Bit 1:0		00	Reserved for Freescale's internal use; always write zero

ADCSC1 = 0x41 (%01000001)

Bit 7	COCO	0	Read-only flag which is set when a conversion completes
Bit 6	AIEN	1	Conversion complete interrupt enabled
Bit 5	ADCO	0	One conversion only (continuous conversions disabled)
Bit 4:0	ADCH	00001	Input channel 1 selected as ADC input channel

ADCRH/L = 0xxx

Holds results of conversion. Read high byte (ADCRH) before low byte (ADCRL) so that conversion data cannot be overwritten with data from the next conversion.

ADCCVH/L = 0xxx

Holds compare value when compare function enabled

APCTL1=0x02

AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins

APCTL2=0x00

All other AD pins remain general purpose I/O pins

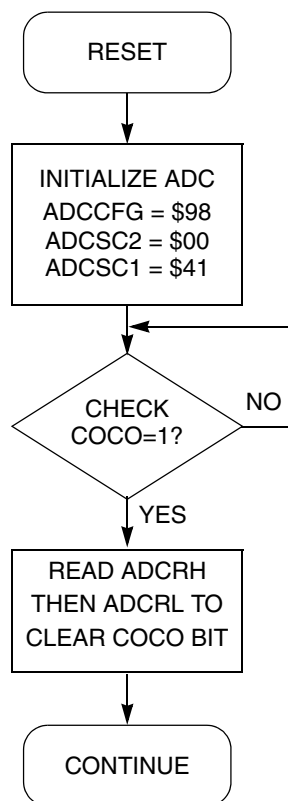


Figure 10-14. Initialization Flowchart for Example

10.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

10.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

10.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDAD} and V_{SSAD}) which are available as separate pins on some devices. On other devices, V_{SSAD} is shared on the same pin as the MCU digital V_{SS} , and on others, both V_{SSAD} and V_{DDAD} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies which are bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDAD} and V_{SSAD} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

In cases where separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSAD} pin. This should be the only ground connection between these supplies if possible. The V_{SSAD} pin makes a good single point ground location.

10.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH} , which may be shared on the same pin as V_{DDAD} on some devices. The low reference is V_{REFL} , which may be shared on the same pin as V_{SSAD} on some devices.

When available on a separate pin, V_{REFH} may be connected to the same potential as V_{DDAD} , or may be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSAD} . Both V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current will cause a voltage drop which could result in conversion errors. Inductance in this path must be minimum (parasitic only).

10.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer will be in its high impedance state and the pullup is disabled. Also, the input buffer draws dc current when its input is not at either V_{DD} or V_{SS} . Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA} .

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL} . If the input is equal to or exceeds V_{REFH} , the converter circuit converts the signal to \$3FF (full scale 10-bit representation) or \$FF (full scale 8-bit representation). If the input is equal to or less than V_{REFL} , the converter circuit converts it to \$000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There will be a brief current associated with V_{REFL} when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

10.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

10.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately $7\text{k}\Omega$ and input capacitance of approximately 5.5 pF , sampling to within $1/4\text{LSB}$ (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below $5\text{ k}\Omega$.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

10.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDAD} / (2^N \cdot I_{LEAK})$ for less than $1/4\text{LSB}$ leakage error ($N = 8$ in 8-bit mode or 10 in 10-bit mode).

10.6.2.3 Noise-Induced Errors

System noise which occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a $0.1\text{ }\mu\text{F}$ low-ESR capacitor from V_{REFH} to V_{REFL} .
- There is a $0.1\text{ }\mu\text{F}$ low-ESR capacitor from V_{DDAD} to V_{SSAD} .
- If inductive isolation is used from the primary supply, an additional $1\text{ }\mu\text{F}$ capacitor is placed from V_{DDAD} to V_{SSAD} .
- V_{SSAD} (and V_{REFL} , if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
 - For software triggered conversions, immediately follow the write to the ADCSC1 with a WAIT instruction or STOP instruction.
 - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a $0.01\text{ }\mu\text{F}$ capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this will improve noise issues but will affect sample rate based on the external analog source resistance).

- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

10.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1\text{LSB} = (V_{\text{REFH}} - V_{\text{REFL}}) / 2^N \quad \text{Eqn. 10-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code will transition when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be $\pm 1/2\text{LSB}$ in 8- or 10-bit mode. As a consequence, however, the code width of the first (\$000) conversion is only $1/2\text{LSB}$ and the code width of the last (\$FF or \$3FF) is 1.5LSB .

10.6.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width ($1/2\text{LSB}$). Note, if the first conversion is \$001, then the difference between the actual \$001 code width and its ideal (1LSB) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5LSB). Note, if the last conversion is \$3FE, then the difference between the actual \$3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

10.6.2.6 Code Jitter, Non-Monotonicity and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around $\pm 1/2$ LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 10.6.2.3](#) will reduce this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values which are never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and to have no missing codes.



Chapter 11

Internal Clock Source (S08ICSV1)

11.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by an internal reference clock. The module can provide either this FLL clock or else the internal reference clock as a source for the MCU system clock. Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

The bus frequency will be one-half of the ICSOUT frequency.

NOTE

The external reference clock is not available for the MC9S08QA4 series.

11.1.1 Module Configuration

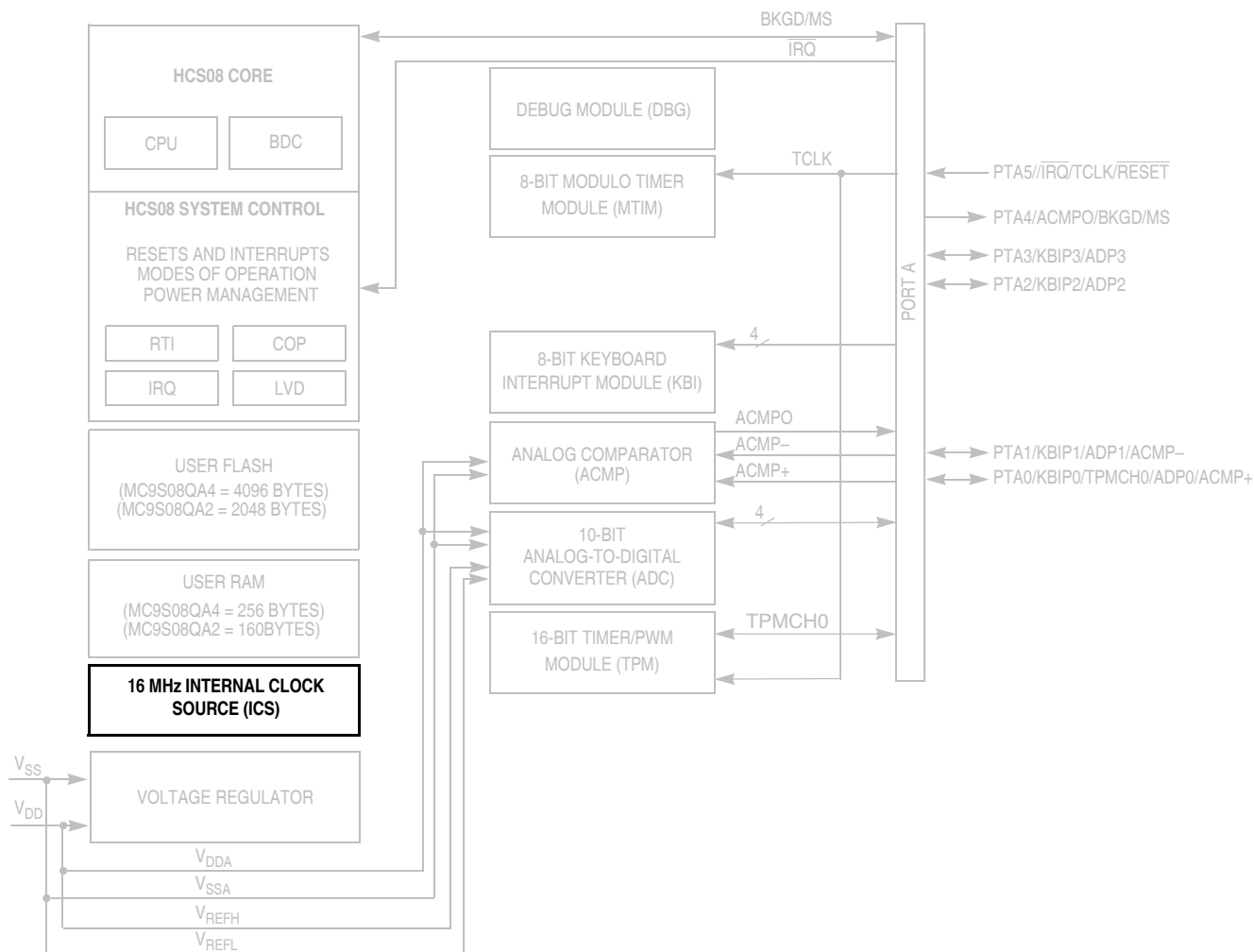
When the internal reference is enabled in stop mode (IREFSTEN = 1), the voltage regulator must also be enabled in stop mode by setting the LVDE and LVDSE bits in the SPMSC1 register.

On this MCU, the internal reference is not connected to any module that is operational in stop mode. Therefore, the IREFSTEN bit in the ICSC1 register must always be cleared.

[Figure 11-1](#) shows the MC9S08QA4 series block diagram with the ICS highlighted.

11.1.2 Factory Trim Value

A factory trim value is stored in flash during production testing. This trim value is calibrated to 31.25 kHz at nominal V_{DD} and temperature. To be used, this value must be copied from flash memory to the ICSTRM register. A factory value for this FTRIM bit is also stored in flash and must be copied into the FTRIM bit in the ICSSC register. See [Table 4-4](#) for the flash locations of the factory ICSTRM and FTRIM values.



NOTES:

- ¹ Port pins are software configurable with pullup device if input port.
- ² Port pins are software configurable for output drive strength.
- ³ Port pins are software configurable for output slew rate control.
- ⁴ $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- ⁵ $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- ⁶ PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
- ⁷ When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 11-1. MC9S08QA4 Series Block Diagram Highlighting ICS Block

11.1.3 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
 - 0.2% resolution using internal 32 kHz reference
 - 2% deviation over voltage and temperature using internal 32 kHz reference
- External reference clock up to 5 MHz can be used to control the FLL
 - 3 bit select for reference divider is provided
- Internal reference clock has 9 trim bits available
- Internal or external reference clock can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
 - 2 bit select for clock divider is provided
 - Allowable dividers are: 1, 2, 4, 8
 - BDC clock is provided as a constant divide by 2 of the DCO output
- Control signals for a low power oscillator as the external reference clock are provided
 - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL engaged internal mode is automatically selected out of reset

11.1.4 Modes of Operation

The ICS features the following modes of operation: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

11.1.4.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock. The BDC clock is supplied from the FLL.

11.1.4.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock. The BDC clock is supplied from the FLL.

11.1.4.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

11.1.4.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock. The BDC clock is not available.

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is supplied from the FLL.

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is not available.

In stop mode, the FLL is disabled and the internal or external reference clock can be selected to be enabled or disabled. The BDC clock is not available. ICS does not provide an MCU clock source.

This section contains the ICS block diagram.



11.2 External Signal Description

No ICS signal connects off chip.

11.3 Register Definition

11.3.1 ICS Control Register 1 (ICSC1)

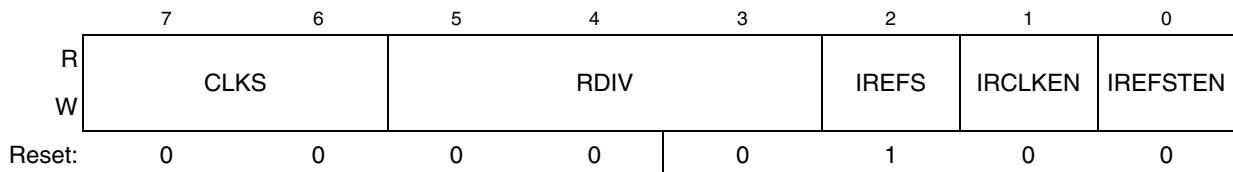


Figure 11-3. ICS Control Register 1 (ICSC1)

Table 11-1. ICS Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	Clock Source Select — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	Reference Divider — Selects the amount to divide down the FLL reference clock selected by the IREFS bits. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. 000 Encoding 0 — Divides reference clock by 1 (reset default) 001 Encoding 1 — Divides reference clock by 2 010 Encoding 2 — Divides reference clock by 4 011 Encoding 3 — Divides reference clock by 8 100 Encoding 4 — Divides reference clock by 16 101 Encoding 5 — Divides reference clock by 32 110 Encoding 6 — Divides reference clock by 64 111 Encoding 7 — Divides reference clock by 128
2 IREFS	Internal Reference Select — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected 0 External reference clock selected
1 IRCLKEN	Internal Reference Clock Enable — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active 0 ICSIRCLK inactive
0 IREFSTEN	Internal Reference Stop Enable — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set or if ICS is in FEI, FBI, or FBILP mode before entering stop 0 Internal reference clock is disabled in stop

11.3.2 ICS Control Register 2 (ICSC2)

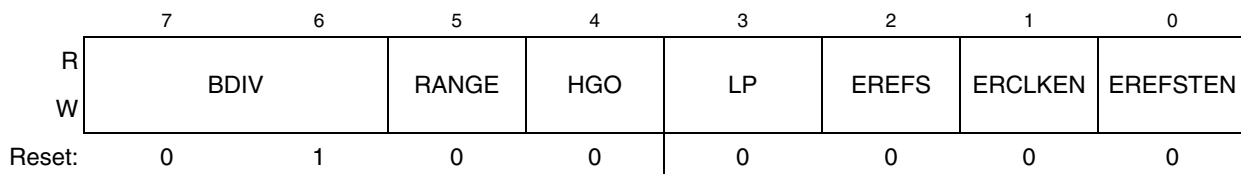


Figure 11-4. ICS Control Register 2 (ICSC2)

Table 11-2. ICS Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	Bus Frequency Divider — Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1 01 Encoding 1 — Divides selected clock by 2 (reset default) 10 Encoding 2 — Divides selected clock by 4 11 Encoding 3 — Divides selected clock by 8
5 RANGE	Frequency Range Select — Selects the frequency range for the external oscillator. 1 High frequency range selected for the external oscillator 0 Low frequency range selected for the external oscillator
4 HGO	High Gain Oscillator Select — The HGO bit controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation 0 Configure external oscillator for low power operation
3 LP	Low Power Select — The LP bit controls whether the FLL is disabled in FLL bypassed modes. 1 FLL is disabled in bypass modes unless BDM is active 0 FLL is not disabled in bypass mode
2 EREFS	External Reference Select — The EREFS bit selects the source for the external reference clock. 1 Oscillator requested 0 External Clock Source requested
1 ERCLKEN	External Reference Enable — The ERCLKEN bit enables the external reference clock for use as ICSECLK. 1 ICSECLK active 0 ICSECLK inactive
0 EREFSTEN	External Reference Stop Enable — The EREFSTEN bit controls whether or not the external reference clock remains enabled when the ICS enters stop mode. 1 External reference clock stays enabled in stop if ERCLKEN is set or if ICS is in FEE, FBE, or FBELP mode before entering stop 0 External reference clock is disabled in stop

11.3.3 ICS Trim Register (ICSTRM)

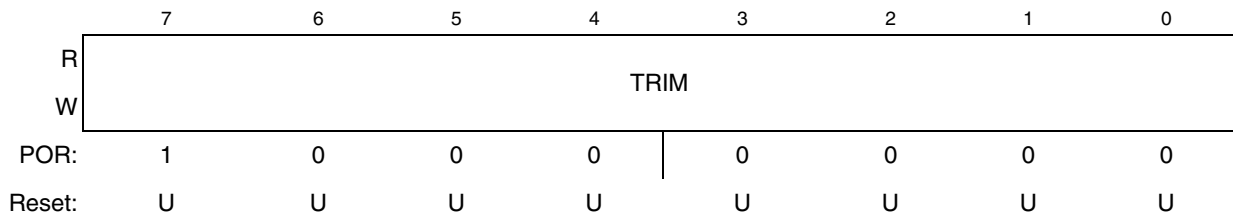


Figure 11-5. ICS Trim Register (ICSTRM)

Table 11-3. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	ICS Trim Setting — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (i.e., bit 1 will adjust twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period. An additional fine trim bit is available in ICSSC as the FTRIM bit.

11.3.4 ICS Status and Control (ICSSC)

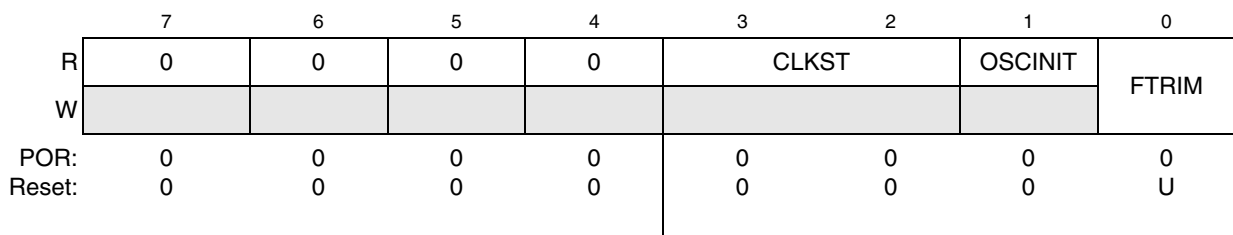


Figure 11-6. ICS Status and Control Register (ICSSC)

Table 11-4. ICS Status and Control Register Field Descriptions

Field	Description
3:2 CLKST	Clock Mode Status — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, Internal reference clock is selected. 10 FLL Bypassed, External reference clock is selected. 11 Reserved.
1	OSC Initialization — If the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared.
FTRIM 0	ICS Fine Trim — The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible.

11.4 Functional Description

11.4.1 Operational Modes

The states of the ICS are shown as a state diagram and are described in the following sections. The arrows indicate the allowed movements between the states.

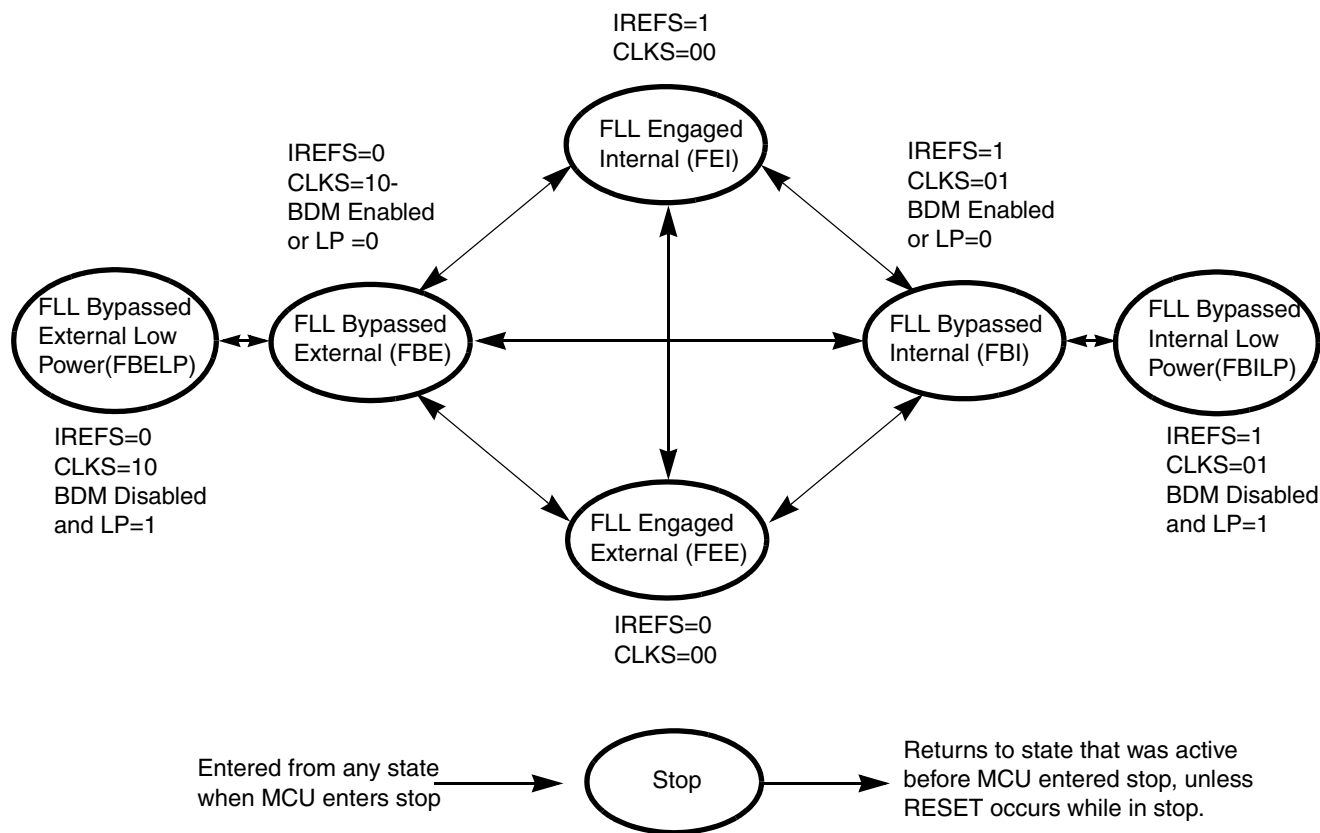


Figure 11-7. Clock Switching Modes

11.4.1.1 FLL Engaged Internal (FEI)

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

- CLKS bits are written to 00
- IREFS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop will lock the frequency to 512 times the filter frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

11.4.1.2 FLL Engaged External (FEE)

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

- CLKS bits are written to 00
- IREFS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock. The FLL loop will lock the frequency to 512 times the filter frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

11.4.1.3 FLL Bypassed Internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- BDM mode is active or LP bit is written to 0

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop will lock the FLL frequency to 512 times the filter frequency, as selected by the RDIV bits. The ICSLCLK will be available for BDC communications, and the internal reference clock is enabled.

11.4.1.4 FLL Bypassed Internal Low Power (FBILP)

The FLL bypassed internal low power (FBILP) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1.
- BDM mode is not active and LP bit is written to 1

In FLL bypassed internal low power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The ICSLCLK will be not be available for BDC communications, and the internal reference clock is enabled.

11.4.1.5 FLL Bypassed External (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock. The FLL clock is controlled by the external reference clock, and the FLL loop will lock the FLL frequency to 512

times the filter frequency, as selected by the RDIV bits, so that the ICSLCLK will be available for BDC communications, and the external reference clock is enabled.

11.4.1.6 FLL Bypassed External Low Power (FBELP)

The FLL bypassed external low power (FBELP) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock and the FLL is disabled. The ICSLCLK will be not be available for BDC communications. The external reference clock is enabled.

11.4.1.7 Stop

ICS stop mode is entered whenever the MCU enters stop. In this mode, all ICS clock signals are stopped except in the following cases:

ICSIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1
- IREFSTEN bit is written to 1

ICSERCLK will be active in stop mode when all the following conditions occur:

- ERCLKEN bit is written to 1
- EREFSTEN bit is written to 1

11.4.2 Mode Switching

When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes the IREFS bit can be changed at anytime, but the RDIV bits must be changed simultaneously so that the resulting frequency stays in the range of 31.25 kHz to 39.0625 kHz. After a change in the IREFS value the FLL will begin locking again after a few full cycles of the resulting divided reference frequency.

The CLKS bits can also be changed at anytime, but the RDIV bits must be changed simultaneously so that the resulting frequency stays in the range of 31.25 kHz to 39.0625 kHz. The actual switch to the newly selected clock will not occur until after a few full cycles of the new clock. If the newly selected clock is not available, the previous clock will remain selected.

11.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency will occur immediately.

11.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. However, in some applications it may be desirable to enable the FLL and allow it to lock for maximum accuracy before switching to an FLL engaged mode. Do this by writing the LP bit to 0.

11.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as ICSIRCLK, which can be used as an additional clock source. The ICSIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the ICSTRM register. Writing a larger value will slow down the ICSIRCLK frequency, and writing a smaller value to the ICSTRM register will speed up the ICSIRCLK frequency. The TRIM bits will effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode. The TRIM and FTRIM value will not be affected by a reset.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value can be copied to the ICSTRM register during reset initialization. The factory trim value does not include the FTRIM bit. For finer precision, the user can trim the internal oscillator in the application and set the FTRIM bit accordingly.

11.4.6 Optional External Reference Clock

The ICS module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in all modes. When the ERCLKEN is set, the external reference clock signal will be presented as ICSECLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL and will only be used as ICSECLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the [Device Overview](#) chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

11.4.7 Fixed Frequency Clock

The ICS provides the divided FLL reference clock as ICSFFCLK for use as an additional clock source for peripheral modules. The ICS provides an output signal (ICSFFE) which indicates when the ICS is providing ICSOUT frequencies four times or greater than the divided FLL reference clock (ICSFFCLK). In FLL engaged mode (FEI and FEE), this is always true and ICSFFE is always high. In ICS Bypass modes, ICSFFE will get asserted for the following combinations of BDIV and RDIV values:

Internal Clock Source (S08ICSV1)

- BDIV=00 (divide by 1), RDIV $\geq$ 010
- BDIV=01 (divide by 2), RDIV $\geq$ 011
- BDIV=10 (divide by 4), RDIV $\geq$ 100
- BDIV=11 (divide by 8), RDIV $\geq$ 101

Chapter 12

Modulo Timer (S08MTIMV1)

12.1 Introduction

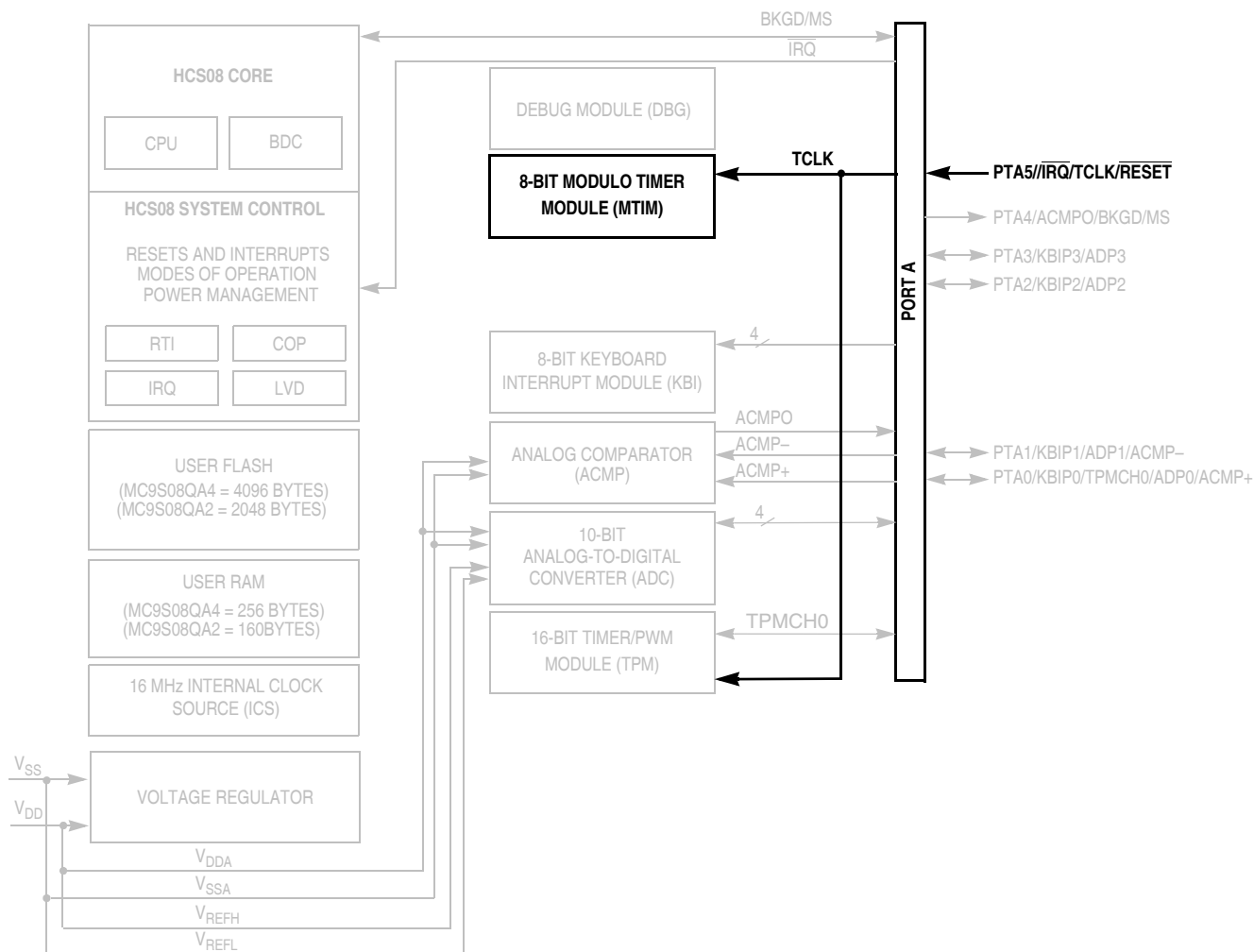
The MTIM is a simple 8-bit timer with several software selectable clock sources and a programmable interrupt.

The central component of the MTIM is the 8-bit counter, which can operate as a free-running counter or a modulo counter. A timer overflow interrupt can be enabled to generate periodic interrupts for time-based software loops.

Figure 12-1 shows the MC9S08QA4 series block diagram with the MTIM highlighted.

12.1.1 MTIM/TPM Configuration Information

The external clock for the MTIM module, TCLK, is selected by setting $CLKS = 1:1$ or $1:0$ in MTIMCLK, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.



NOTES:

- ¹ Port pins are software configurable with pullup device if input port.
- ² Port pins are software configurable for output drive strength.
- ³ Port pins are software configurable for output slew rate control.
- ⁴ $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- ⁵ $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- ⁶ PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
- ⁷ When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 12-1. MC9S08QA4 Series Block Diagram Highlighting MTIM Block and Pins

12.1.2 Features

Timer system features include:

- 8-bit up-counter
 - Free-running or 8-bit modulo limit
 - Software controllable interrupt on overflow
 - Counter reset bit (TRST)
 - Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
 - System bus clock — rising edge
 - Fixed frequency clock (XCLK) — rising edge
 - External clock source on the TCLK pin — rising edge
 - External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
 - Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

12.1.3 Modes of Operation

This section defines the MTIM's operation in stop, wait and background debug modes.

12.1.3.1 MTIM in Wait Mode

The MTIM continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the MTIM can be used to bring the MCU out of wait mode if the timer overflow interrupt is enabled. For lowest possible current consumption, the MTIM should be stopped by software if not needed as an interrupt source during wait mode.

12.1.3.2 MTIM in Stop Modes

The MTIM is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM cannot be used as a wake up source from stop modes.

Waking from stop1 and stop2 modes, the MTIM will be put into its reset state. If stop3 is exited with a reset, the MTIM will be put into its reset state. If stop3 is exited with an interrupt, the MTIM continues from the state it was in when stop3 was entered. If the counter was active upon entering stop3, the count will resume from the current value.

12.1.3.3 MTIM in Active Background Mode

The MTIM suspends all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM reset did not occur (TRST written to a 1 or MTIM1MOD written).

12.1.4 Block Diagram

The block diagram for the modulo timer module is shown [Figure 12-2](#).

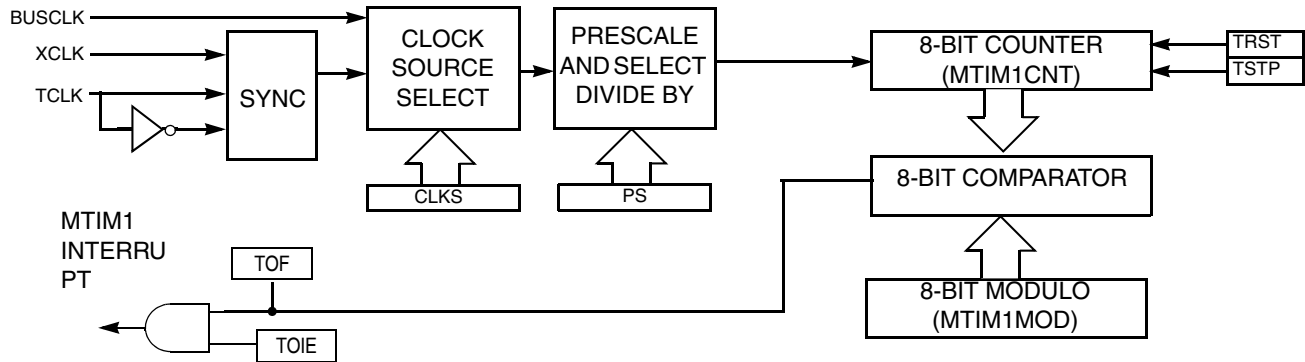


Figure 12-2. Modulo Timer (MTIM) Block Diagram

12.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in [Table 12-1](#).

Table 12-1. Signal Properties

Signal	Function	I/O
TCLK	External clock source input into MTIM	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. Therefore, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. See the [Pins and Connections](#) chapter for the pin location and priority of this function.

12.3 Register Definition

[Figure 12-3](#) is a summary of MTIM registers.

Figure 12-3. MTIM1 Register Summary

Name		7	6	5	4	3	2	1	0
MTIM1SC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIM1CLK	R	0	0	CLKS		PS			
	W								
MTIM1CNT	R	COUNT							
	W								
MTIM1MOD	R	MOD							
	W								

Each MTIM includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all MTIM registers. This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM, so register names include placeholder characters to identify which MTIM is being referenced.

12.3.1 MTIM1 Status and Control Register (MTIM1SC)

MTIM1SC contains the overflow status flag and control bits which are used to configure the interrupt enable, reset the counter, and stop the counter.

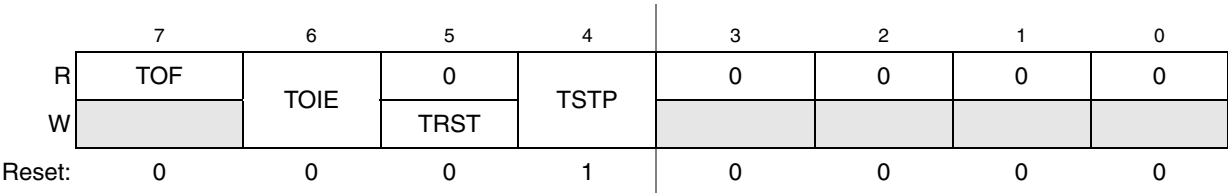


Figure 12-4. MTIM1 Status and Control Register

Table 12-2. MTIM1 Status and Control Register Field Descriptions

Field	Description
7 TOF	MTIM Overflow Flag — This read-only bit is set when the MTIM counter register overflows to \$00 after reaching the value in the MTIM modulo register. Clear TOF by reading the MTIM1SC register while TOF is set, then writing a 0 to TOF. TOF is also cleared when TRST is written to a 1 or when any value is written to the MTIM1MOD register. 0 MTIM counter has not reached the overflow value in the MTIM modulo register. 1 MTIM counter has reached the overflow value in the MTIM modulo register.
6 TOIE	MTIM Overflow Interrupt Enable — This read/write bit enables MTIM overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1. Clear TOF first, then set TOIE. 0 TOF interrupts are disabled. Use software polling. 1 TOF interrupts are enabled.
5 TRST	MTIM Counter Reset — When a 1 is written to this write-only bit, the MTIM counter register resets to \$00 and TOF is cleared. Reading this bit always returns 0. 0 No effect. MTIM counter remains at current state. 1 MTIM counter is reset to \$00.
4 TSTP	MTIM Counter Stop — When set, this read/write bit stops the MTIM counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM from counting. 0 MTIM counter is active. 1 MTIM counter is stopped.
3:0	Unused register bits, always read 0.

12.3.2 MTIM1 Clock Configuration Register (MTIM1CLK)

MTIM1CLK contains the clock select bits (CLKS) and the prescaler select bits (PS).

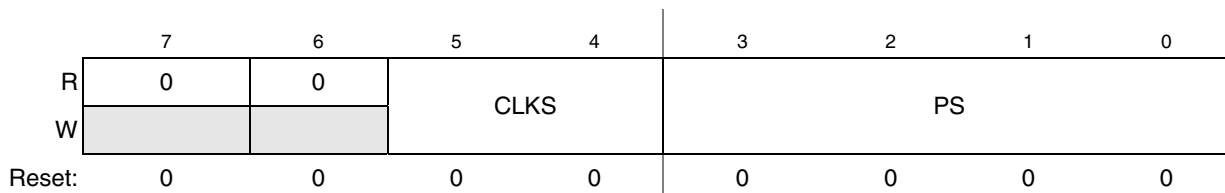


Figure 12-5. MTIM1 Clock Configuration Register

Table 12-3. MTIM1 Clock Configuration Register Field Description

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	Clock Source Select — These two read/write bits select one of four different clock sources as the input to the MTIM prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 000. 00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge All other encodings default to the bus clock (BUSCLK).
3:0 PS	Clock Source Prescaler — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM clock source ÷ 1 0001 Encoding 1. MTIM clock source ÷ 2 0010 Encoding 2. MTIM clock source ÷ 4 0011 Encoding 3. MTIM clock source ÷ 8 0100 Encoding 4. MTIM clock source ÷ 16 0101 Encoding 5. MTIM clock source ÷ 32 0110 Encoding 6. MTIM clock source ÷ 64 0111 Encoding 7. MTIM clock source ÷ 128 1000 Encoding 8. MTIM clock source ÷ 256 All other encodings default to MTIM clock source ÷ 256.

12.3.3 MTIM1 Counter Register (MTIM1CNT)

MTIM1CNT is the read-only value of the current MTIM count of the 8-bit counter.

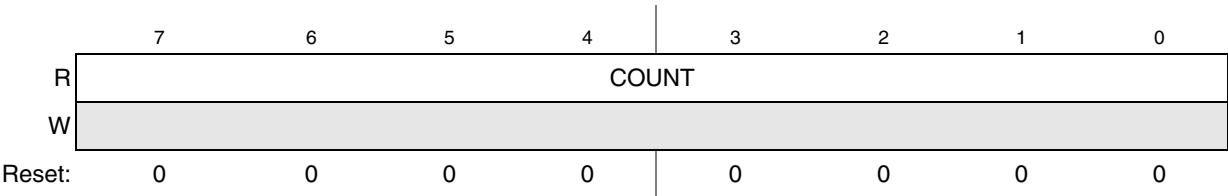


Figure 12-6. MTIM1 Counter Register

Table 12-4. MTIM1 Counter Register Field Description

Field	Description
7:0 COUNT	MTIM Count — These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset clears the count to \$00.

12.3.4 MTIM1 Modulo Register (MTIM1MOD)

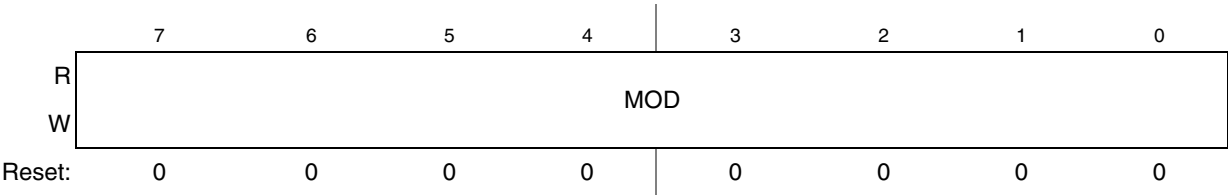


Figure 12-7. MTIM1 Modulo Register

Table 12-5. MTIM1 Modulo Register Field Descriptions

Field	Description
7:0 MOD	MTIM Modulo — These eight read/write bits contain the modulo value used to reset the count and set TOF. A value of \$00 puts the MTIM in free-running mode. Writing to MTIM1MOD resets the COUNT to \$00 and clears TOF. Reset sets the modulo to \$00.

12.4 Functional Description

The MTIM is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM counter (MTIM1CNT) has three modes of operation: stopped, free-running, and modulo. Out of reset, the counter is stopped. If the counter is started without writing a new value to the modulo register, then the counter will be in free-running mode. The counter is in modulo mode when a value other than \$00 is in the modulo register while the counter is running.

After any MCU reset, the counter is stopped and reset to \$00, and the modulus is set to \$00. The bus clock is selected as the default clock source and the prescale value is divide by 1. To start the MTIM in free-running mode, simply write to the MTIM status and control register (MTIM1SC) and clear the MTIM stop bit (TSTP).

Four clock sources are software selectable: the internal bus clock, the fixed frequency clock (XCLK), and an external clock on the TCLK pin, selectable as incrementing on either rising or falling edges. The MTIM clock select bits (CLKS1:CLKS0) in MTIM1SC are used to select the desired clock source. If the counter is active (TSTP = 0) when a new clock source is selected, the counter will continue counting from the previous value using the new clock source.

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in MTIM1SC select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter will continue counting from the previous value using the new prescaler value.

The MTIM modulo register (MTIM1MOD) allows the overflow compare value to be set to any value from \$01 to \$FF. Reset clears the modulo value to \$00, which results in a free running counter.

When the counter is active (TSTP = 0), the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter overflows to \$00 and continues counting. The MTIM overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to \$00. Writing to MTIM1MOD while the counter is active resets the counter to \$00 and clears TOF.

Clearing TOF is a two-step process. The first step is to read the MTIM1SC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF will remain set after the second step is performed. This will prevent the second occurrence from being missed. TOF is also cleared when a 1 is written to TRST or when any value is written to the MTIM1MOD register.

The MTIM allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM overflow interrupt, set the MTIM overflow interrupt enable bit (TOIE) in MTIM1SC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

12.4.1 MTIM Operation Example

This section shows an example of the MTIM operation as the counter reaches a matching value from the modulo register.

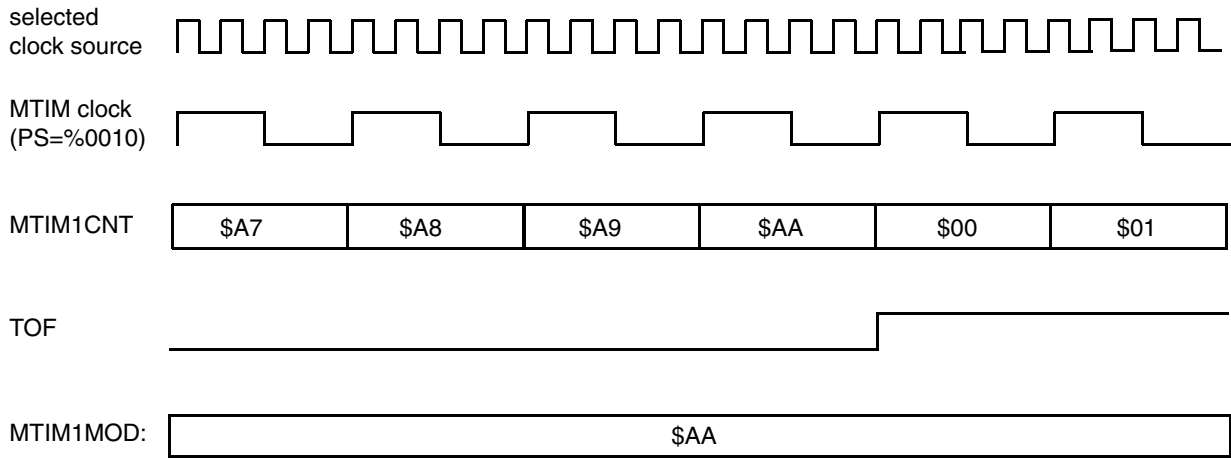


Figure 12-8. MTIM counter overflow example

In the example of [Figure 12-8](#), the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIM1MOD register is set to \$AA. When the counter, MTIM1CNT, reaches the modulo value of \$AA, the counter overflows to \$00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from \$AA to \$00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

Chapter 13

Timer/Pulse-Width Modulator (S08TPMV2)

13.1 Introduction

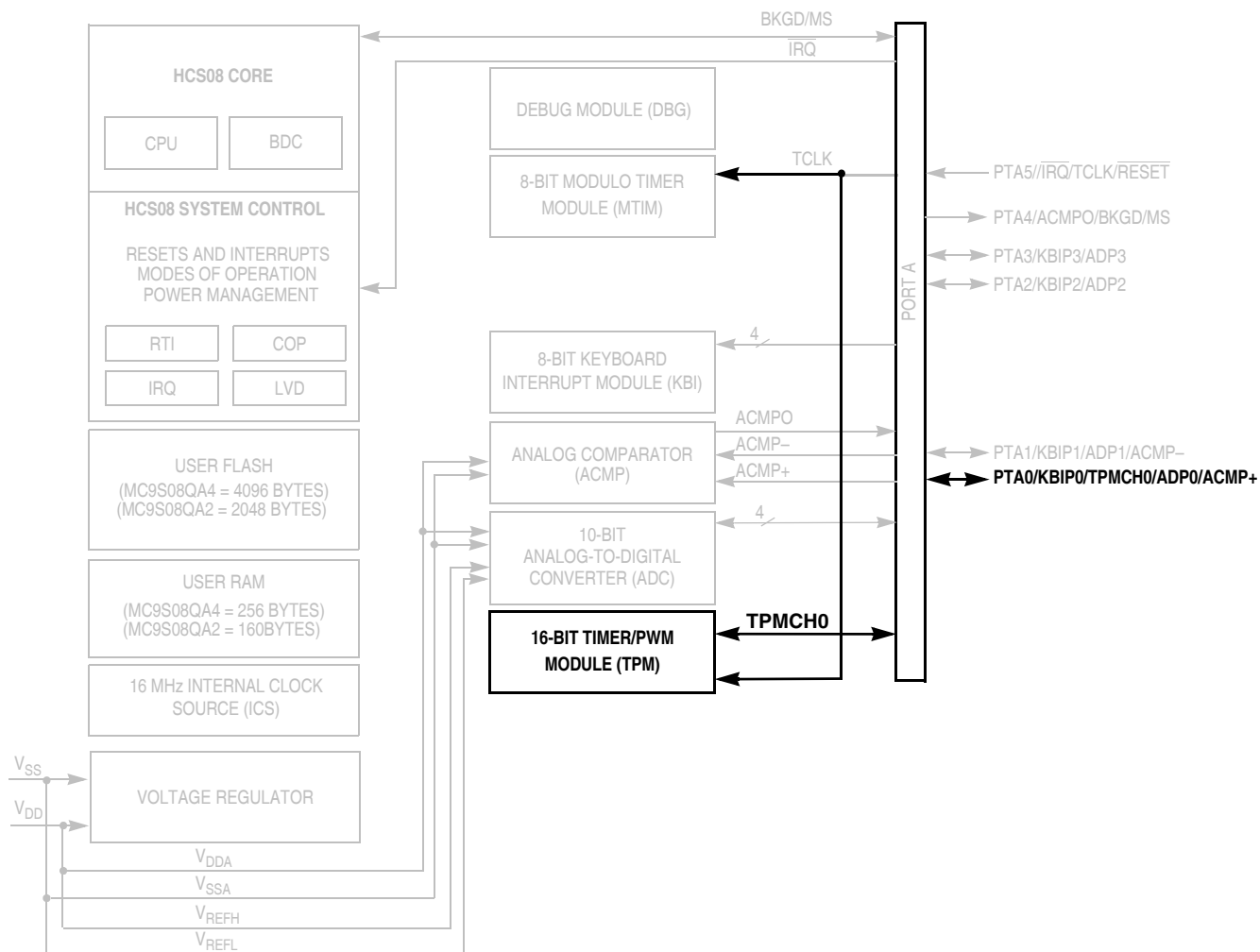
Figure 13-1 shows the MC9S08QA4 series block diagram with the TPM highlighted.

13.1.1 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.

13.1.2 MTIM/TPM Configuration Information

The external clock for the TPM module, TPMCLK, is selected by setting $CLKS[B:A] = 1:1$ in TPMSC, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.



NOTES:

- ¹ Port pins are software configurable with pullup device if input port.
- ² Port pins are software configurable for output drive strength.
- ³ Port pins are software configurable for output slew rate control.
- ⁴ $\overline{\text{IRQ}}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{\text{IRQ}}$ pin function (IRQPE = 1).
- ⁵ $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
- ⁶ PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
- ⁷ When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 13-1. MC9S08QA4 Series Block Diagram Highlighting TPM Block and Pins

13.1.3 Features

The TPM has the following features:

- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock sources independently selectable per TPM (multiple TPMs device)
- Selectable clock sources (device dependent): bus clock, fixed system clock, external pin
- Clock prescaler taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Timer system enable
- One interrupt per channel plus a terminal count interrupt for each TPM module (multiple TPMs device)
- Channel features:
 - Each channel may be input capture, output compare, or buffered edge-aligned PWM
 - Rising-edge, falling-edge, or any-edge input capture trigger
 - Set, clear, or toggle output compare action
 - Selectable polarity on PWM outputs

13.1.4 Block Diagram

Figure 13-2 shows the structure of a TPM. Some MCUs include more than one TPM, with various numbers of channels.

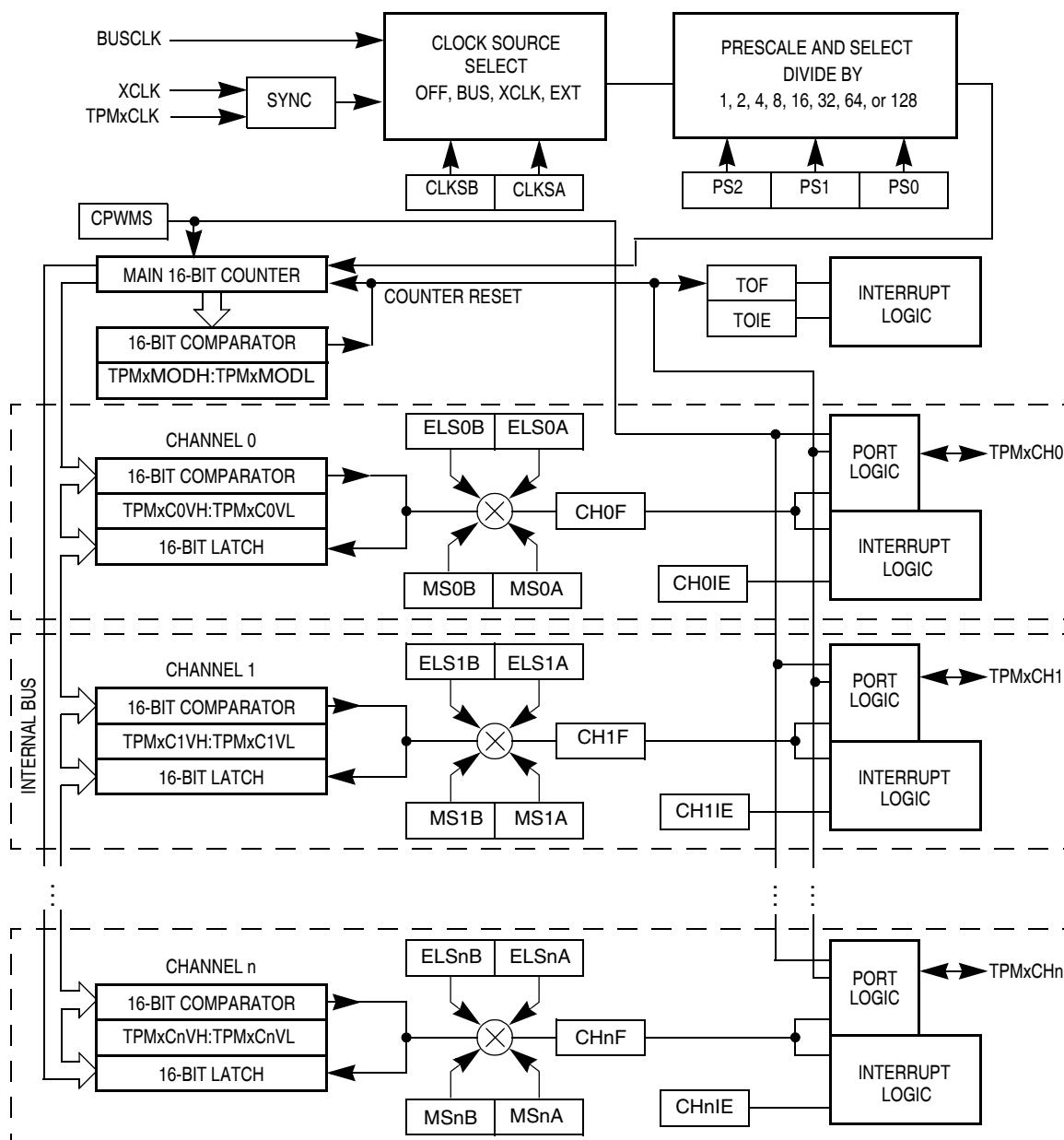


Figure 13-2. TPM Block Diagram

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter. (The values 0x0000 or 0xFFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the TPMxCNT counter resets the counter regardless of the data value written.

All TPM channels are programmable independently as input capture, output compare, or buffered edge-aligned PWM channels.

13.2 External Signal Description

When any pin associated with the timer is configured as a timer input, a passive pullup can be enabled. After reset, the TPM modules are disabled and all pins default to general-purpose inputs with the passive pullups disabled.

13.2.1 External TPM Clock Sources

When control bits CLKSB:CLKSA in the timer status and control register are set to 1:1, the prescaler and consequently the 16-bit counter for TPMx are driven by an external clock source, TPMxCLK, connected to an I/O pin. A synchronizer is needed between the external clock and the rest of the TPM. This synchronizer is clocked by the bus clock so the frequency of the external source must be less than one-half the frequency of the bus rate clock. The upper frequency limit for this external clock source is specified to be one-fourth the bus frequency to conservatively accommodate duty cycle and phase-locked loop (PLL) or frequency-locked loop (FLL) frequency jitter effects.

On some devices the external clock input is shared with one of the TPM channels. When a TPM channel is shared as the external clock input, the associated TPM channel cannot use the pin. (The channel can still be used in output compare mode as a software timer.) Also, if one of the TPM channels is used as the external clock input, the corresponding ELSnB:ELSnA control bits must be set to 0:0 so the channel is not trying to use the same pin.

13.2.2 TPMxCHn — TPMx Channel n I/O Pins

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the configuration of the channel. In some cases, no pin function is needed so the pin reverts to being controlled by general-purpose I/O controls. When a timer has control of a port pin, the port data and data direction registers do not affect the related pin(s). See the [Pins and Connections](#) chapter for additional information about shared pin functions.

13.3 Register Definition

The TPM includes:

- An 8-bit status and control register (TPMxSC)
- A 16-bit counter (TPMxCNTH:TPMxCNTL)
- A 16-bit modulo register (TPMxMODH:TPMxMODL)

Each timer channel has:

- An 8-bit status and control register (TPMxCnSC)
- A 16-bit channel value register (TPMxCnVH:TPMxCnVL)

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all TPM registers. This section refers to registers and control bits only by their names. A

Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

13.3.1 Timer Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.

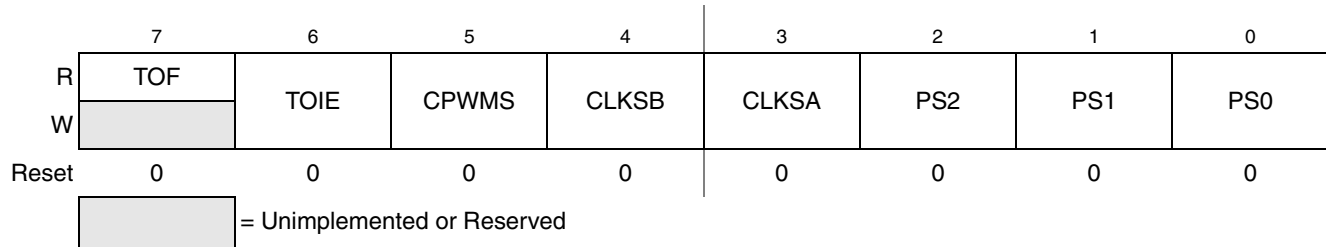


Figure 13-3. Timer Status and Control Register (TPMxSC)

Table 13-1. TPMxSC Register Field Descriptions

Field	Description
7 TOF	Timer Overflow Flag — This flag is set when the TPM counter changes to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value. Clear TOF by reading the TPM status and control register when TOF is set and then writing a 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. Reset clears TOF. Writing a 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer Overflow Interrupt Enable — This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals 1. Reset clears TOIE. 0 TOF interrupts inhibited (use software polling) 1 TOF interrupts enabled
5 CPWMS	Center-Aligned PWM Select — This read/write bit selects CPWM operating mode. Reset clears this bit so the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up-/down-counting mode for CPWM functions. Reset clears CPWMS. 0 All TPMx channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register 1 All TPMx channels operate in center-aligned PWM mode
4:3 CLKS[B:A]	Clock Source Select — As shown in Table 13-2 , this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The external source and the XCLK are synchronized to the bus clock by an on-chip synchronization circuit.
2:0 PS[2:0]	Prescale Divisor Select — This 3-bit field selects one of eight divisors for the TPM clock input as shown in Table 13-3 . This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system.

Table 13-2. TPM Clock Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
0:0	No clock selected (TPMx disabled)
0:1	Bus rate clock (BUSCLK)
1:0	Fixed system clock (XCLK)
1:1	External source (TPMxCLK) ^{1,2}

¹ The maximum frequency that is allowed as an external clock is one-fourth of the bus frequency.

² If the external clock input is shared with channel n and is selected as the TPM clock source, the corresponding ELSnB:ELSnA control bits should be set to 0:0 so channel n does not try to use the same pin for a conflicting function.

Table 13-3. Prescale Divisor Selection

PS2:PS1:PS0	TPM Clock Source Divided-By
0:0:0	1
0:0:1	2
0:1:0	4
0:1:1	8
1:0:0	16
1:0:1	32
1:1:0	64
1:1:1	128

13.3.2 Timer Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This allows coherent 16-bit reads in either order. The coherency mechanism is automatically restarted by an MCU reset, a write of any value to TPMxCNTH or TPMxCNTL, or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers.

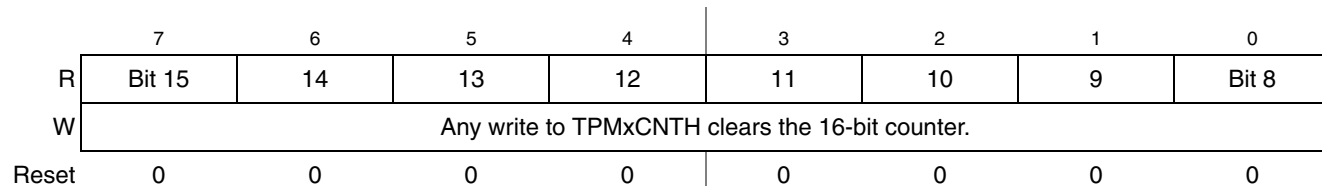


Figure 13-4. Timer Counter Register High (TPMxCNTH)

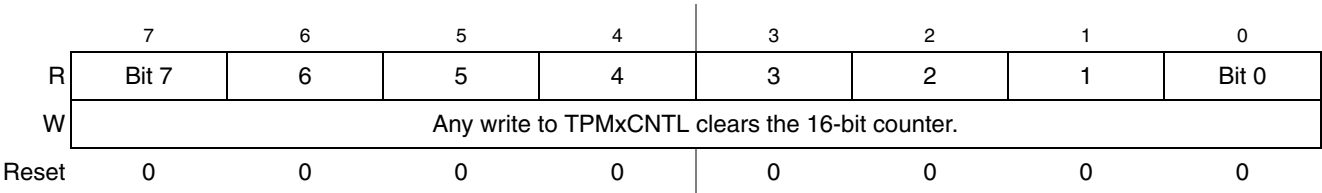


Figure 13-5. Timer Counter Register Low (TPMxCNTL)

When background mode is active, the timer counter and the coherency mechanism are frozen such that the buffer latches remain in the state they were in when the background mode became active even if one or both bytes of the counter are read while background mode is active.

13.3.3 Timer Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits TOF and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000, which results in a free-running timer counter (modulo disabled).

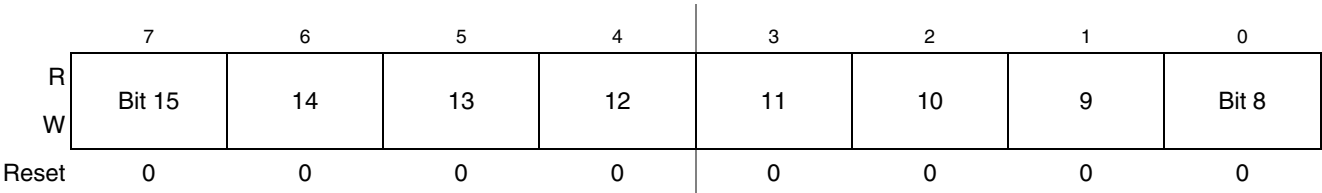


Figure 13-6. Timer Counter Modulo Register High (TPMxMODH)

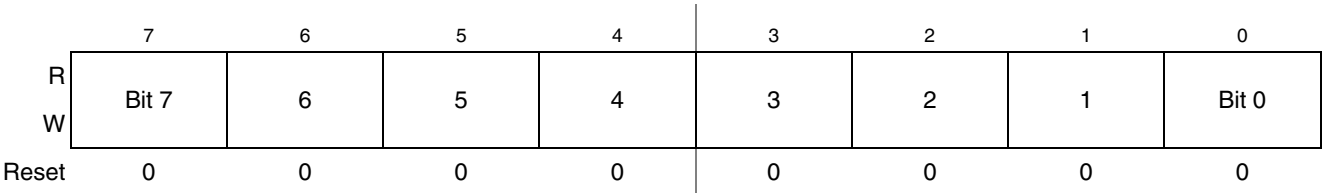


Figure 13-7. Timer Counter Modulo Register Low (TPMxMODL)

It is good practice to wait for an overflow interrupt so both bytes of the modulo register can be written well before a new overflow. An alternative approach is to reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

13.3.4 Timer Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel interrupt status flag and control bits that are used to configure the interrupt enable, channel configuration, and pin function.

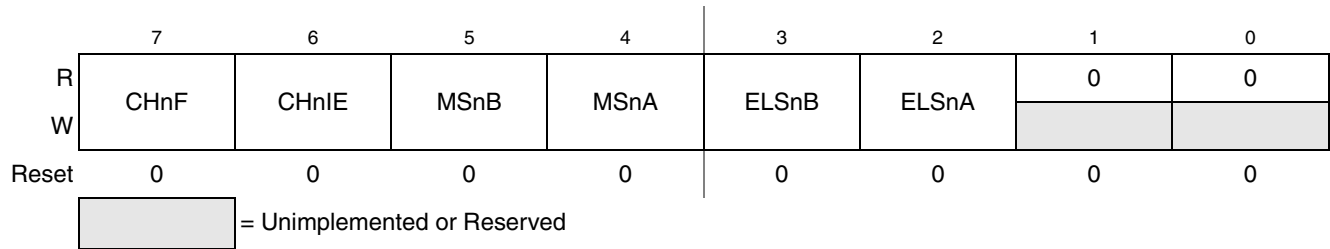


Figure 13-8. Timer Channel n Status and Control Register (TPMxCnSC)

Table 13-4. TPMxCnSC Register Field Descriptions

Field	Description
7 CHnF	Channel n Flag — When channel n is configured for input capture, this flag bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. This flag is seldom used with center-aligned PWMs because it is set every time the counter matches the channel value register, which correspond to both edges of the active duty cycle period. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while CHnF is set and then writing a 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF would remain set after the clear sequence was completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost by clearing a previous CHnF. Reset clears CHnF. Writing a 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event occurred on channel n
6 CHnIE	Channel n Interrupt Enable — This read/write bit enables interrupts from channel n. Reset clears CHnIE. 0 Channel n interrupt requests disabled (use software polling) 1 Channel n interrupt requests enabled
5 MSnB	Mode Select B for TPM Channel n — When CPWMS = 0, MSnB = 1 configures TPM channel n for edge-aligned PWM mode. For a summary of channel mode and setup controls, refer to Table 13-5 .
4 MSnA	Mode Select A for TPM Channel n — When CPWMS = 0 and MSnB = 0, MSnA configures TPM channel n for input capture mode or output compare mode. Refer to Table 13-5 for a summary of channel mode and setup controls.
3:2 ELSn[B:A]	Edge/Level Select Bits — Depending on the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 13-5 , these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output. Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general-purpose I/O pin unrelated to any timer channel functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general-purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 13-5. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM channel; use as an external clock for the TPM or revert to general-purpose I/O	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts or using the status flags to avoid any unexpected behavior.

13.3.5 Timer Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel value registers are cleared by reset.

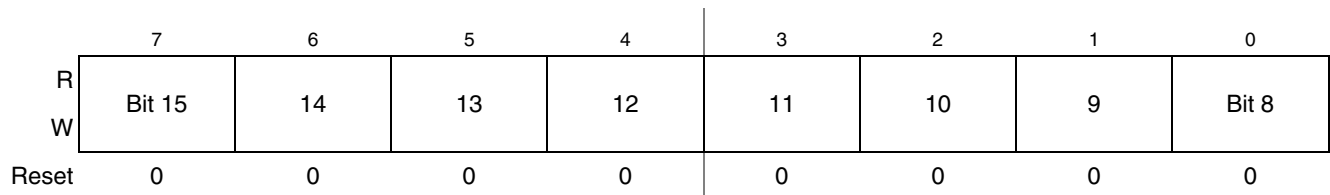


Figure 13-9. Timer Channel Value Register High (TPMxCnVH)

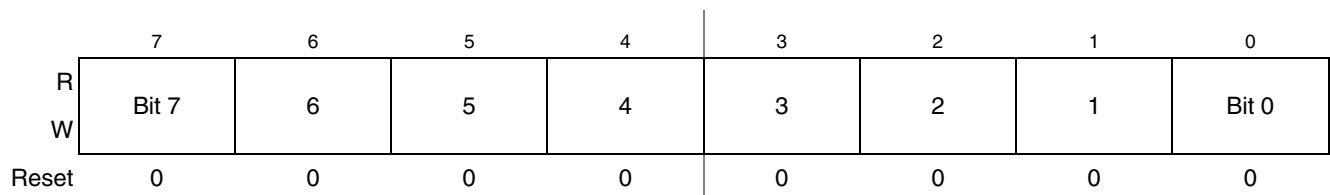


Figure 13-10. Timer Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the timer channel value registers. This latching mechanism may be manually reset by writing to the TPMxCnSC register.

This latching mechanism allows coherent 16-bit writes in either order, which is friendly to various compiler implementations.

13.4 Functional Description

All TPM functions are associated with a main 16-bit counter that allows flexible selection of the clock source and prescale divisor. A 16-bit modulo register also is associated with the main 16-bit counter in the TPM. Each TPM channel is optionally associated with an MCU pin and a maskable interrupt function.

The TPM has center-aligned PWM capabilities controlled by the CPWMS control bit in TPMxSC. When CPWMS is set to 1, timer counter TPMxCNT changes to an up-/down-counter and all channels in the associated TPM act as center-aligned PWM channels. When CPWMS = 0, each channel can independently be configured to operate in input capture, output compare, or buffered edge-aligned PWM mode.

The following sections describe the main 16-bit counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend on the operating mode, these topics are covered in the associated mode sections.

13.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, up-counting vs. up-/down-counting, end-of-count overflow, and manual counter reset.

After any MCU reset, CLKS_B:CLKS_A = 0:0 so no clock source is selected and the TPM is inactive. Normally, CLKS_B:CLKS_A would be set to 0:1 so the bus clock drives the timer counter. The clock source for the TPM can be selected to be off, the bus clock (BUSCLK), the fixed system clock (XCLK), or an external input. The maximum frequency allowed for the external clock option is one-fourth the bus rate. Refer to [Section 13.3.1, “Timer Status and Control Register \(TPMxSC\)”](#) and [Table 13-2](#) for more information about clock source selection.

When the microcontroller is in active background mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all TPM clocks are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally.

The main 16-bit counter has two counting modes. When center-aligned PWM is selected (CPWMS = 1), the counter operates in up-/down-counting mode. Otherwise, the counter operates as a simple up-counter. As an up-counter, the main 16-bit counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts upward from 0x0000 through its terminal count and then counts downward to 0x0000 where it returns to up-counting. Both 0x0000 and the terminal count value (value in TPMxMODH:TPMxMODL) are normal length counts (one timer clock period long).

An interrupt flag and enable are associated with the main 16-bit counter. The timer overflow flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE = 0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE = 1) where a static hardware interrupt is automatically generated whenever the TOF flag is 1.

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the main 16-bit counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the main 16-bit counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

Because the HCS08 MCU is an 8-bit architecture, a coherency mechanism is built into the timer counter for read operations. Whenever either byte of the counter is read (TPMxCNTH or TPMxCNTL), both bytes are captured into a buffer so when the other byte is read, the value will represent the other byte of the count at the time the first byte was read. The counter continues to count normally, but no new value can be read from either byte until both bytes of the old count have been read.

The main timer counter can be reset manually at any time by writing any value to either byte of the timer count TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only one byte of the counter was read before resetting the count.

13.4.2 Channel Mode Selection

Provided CPWMS = 0 (center-aligned PWM operation is not specified), the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and buffered edge-aligned PWM.

13.4.2.1 Input Capture Mode

With the input capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TPM latches the contents of the TPM counter into the channel value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either byte of the 16-bit capture register is read, both bytes are latched into a buffer to support coherent 16-bit accesses regardless of order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

13.4.2.2 Output Compare Mode

With the output compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel value registers of an output compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel value registers only after both 8-bit bytes of a 16-bit register have been written. This coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

13.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS = 0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the setting in the modulus register (TPMxMODH:TPMxMODL). The duty cycle is determined by the setting in the timer channel value register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As Figure 13-11 shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. The time between the modulus overflow and the output compare is the pulse width. If ELSnA = 0, the counter overflow forces the PWM signal high and the output compare forces the PWM signal low. If ELSnA = 1, the counter overflow forces the PWM signal low and the output compare forces the PWM signal high.

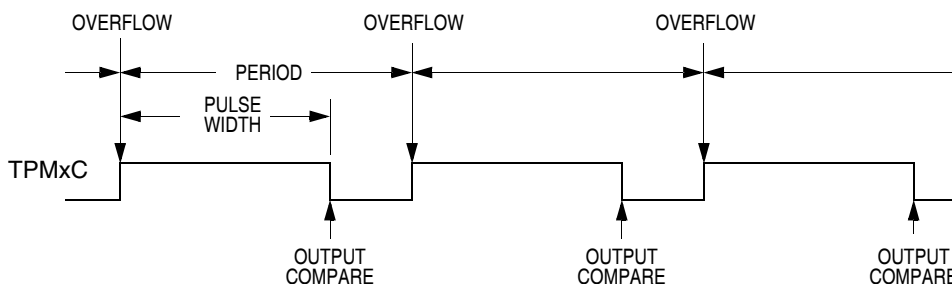


Figure 13-11. PWM Period and Pulse Width (ELSnA = 0)

When the channel value register is set to 0x0000, the duty cycle is 0 percent. By setting the timer channel value register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting, 100% duty cycle can be achieved. This implies that the modulus setting must be less than 0xFFFF to get 100% duty cycle.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to either register, TPMxCnVH or TPMxCnVL, write to buffer registers. In edge-PWM mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the value in the TPMxCNTH:TPMxCNTL counter is 0x0000. (The new duty cycle does not take effect until the next full period.)

13.4.3 Center-Aligned PWM Mode

This type of PWM output uses the up-/down-counting mode of the timer counter (CPWMS = 1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPMxMODH:TPMxMODL.

TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPMxCnVH:TPMxCnVL}) \quad \text{Eqn. 13-1}$$

$$\begin{aligned} \text{period} &= 2 \times (\text{TPMxMODH:TPMxMODL}); \\ &\text{for TPMxMODH:TPMxMODL} = 0x0001\text{--}0x7FFF \end{aligned} \quad \text{Eqn. 13-2}$$

If the channel value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if generation of 100% duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

TPMxMODH:TPMxMODL = 0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS = 0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS = 1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

Figure 13-12 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If ELSnA = 0, the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

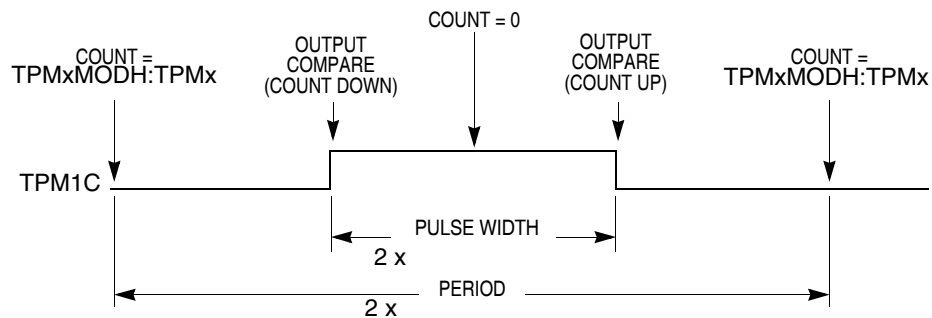


Figure 13-12. CPWM Period and Pulse Width (ELSnA = 0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers, TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers. Values are

transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the timer counter overflows (reverses direction from up-counting to down-counting at the end of the terminal count in the modulus register). This TPMxCNT overflow requirement only applies to PWM channels, not output compares.

Optionally, when $\text{TPMxCNTH}:\text{TPMxCNTL} = \text{TPMxMODH}:\text{TPMxMODL}$, the TPM can generate a TOF interrupt at the end of this count. The user can choose to reload any number of the PWM buffers, and they will all update simultaneously at the start of a new period.

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

13.5 TPM Interrupts

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on the mode of operation for each channel. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register. See the [Resets, Interrupts, and System Configuration](#) chapter for absolute interrupt vector addresses, priority, and local interrupt mask control bits.

For each interrupt source in the TPM, a flag bit is set on recognition of the interrupt condition such as timer overflow, channel input capture, or output compare events. This flag may be read (polled) by software to verify that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will be generated whenever the associated interrupt flag equals 1. It is the responsibility of user software to perform a sequence of steps to clear the interrupt flag before returning from the interrupt service routine.

13.5.1 Clearing Timer Interrupt Flags

TPM interrupt flags are cleared by a 2-step process that includes a read of the flag bit while it is set (1) followed by a write of 0 to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

13.5.2 Timer Overflow Interrupt Description

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

13.5.3 Channel Event Interrupt Description

The meaning of channel interrupts depends on the current mode of the channel (input capture, output compare, edge-aligned PWM, or center-aligned PWM).

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select rising edges, falling edges, any edge, or no edge (off) as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the 2-step sequence described in [Section 13.5.1, “Clearing Timer Interrupt Flags.”](#)

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the 2-step sequence described in [Section 13.5.1, “Clearing Timer Interrupt Flags.”](#)

13.5.4 PWM End-of-Duty-Cycle Events

For channels that are configured for PWM operation, there are two possibilities:

- When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period.
- When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle, which are the times when the timer counter matches the channel value register.

The flag is cleared by the 2-step sequence described in [Section 13.5.1, “Clearing Timer Interrupt Flags.”](#)

Chapter 14 Development Support

14.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip FLASH and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 Family, address and data bus signals are not available on external pins. Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

14.1.1 Module Configuration

The alternate BDC clock source is the ICSLCLK. This clock source is selected by clearing the CLKSW bit in the BDCSCR register. For details on ICSLCLK, see [Section 11.4, “Functional Description”](#) of the ICS chapter.

14.1.2 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system include:

- Two trigger comparators: Two address + read/write (R/W) or one full address + data + R/W
- Flexible 8-word by 16-bit FIFO (first-in, first-out) buffer for capture information:
 - Change-of-flow addresses or
 - Event-only data
- Two types of breakpoints:
 - Tag breakpoints for instruction opcodes
 - Force breakpoints for any address access
- Nine trigger modes:
 - Basic: A-only, A OR B
 - Sequence: A then B
 - Full: A AND B data, A AND NOT B data
 - Event (store data): Event-only B, A then event-only B
 - Range: Inside range ($A \leq \text{address} \leq B$), outside range ($\text{address} < A$ or $\text{address} > B$)

14.2 Background Debug Controller (BDC)

All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be

read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD} . An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

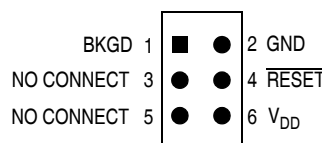


Figure 14-1. BDM Tool Connector

14.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to [Section 14.2.2, "Communication Details."](#)

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively

driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 14.2.2, “Communication Details,”](#) for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

14.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

[Figure 14-2](#) shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

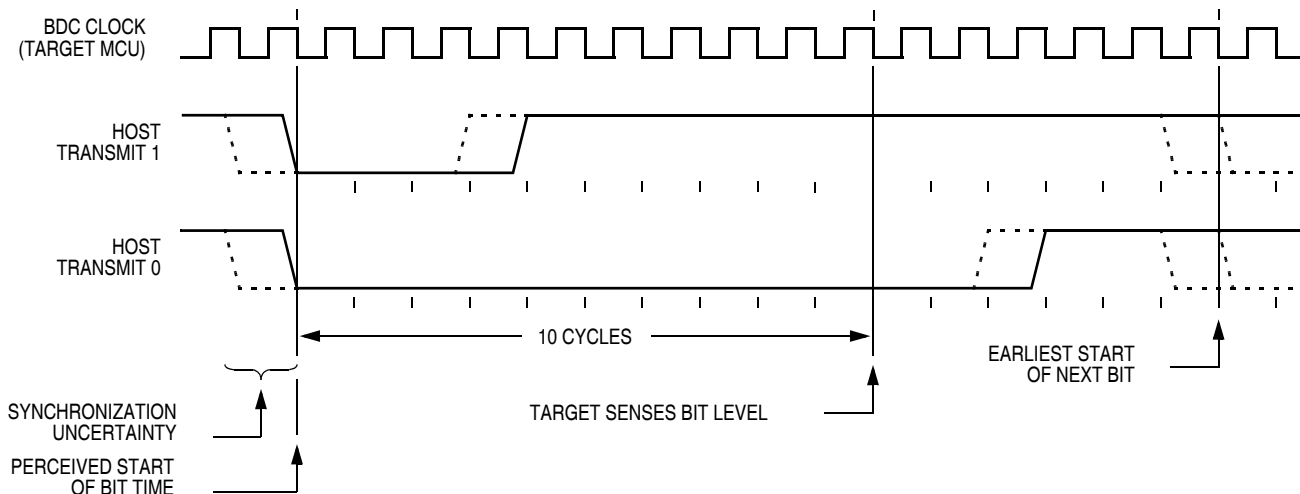


Figure 14-2. BDC Host-to-Target Serial Bit Timing

Figure 14-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

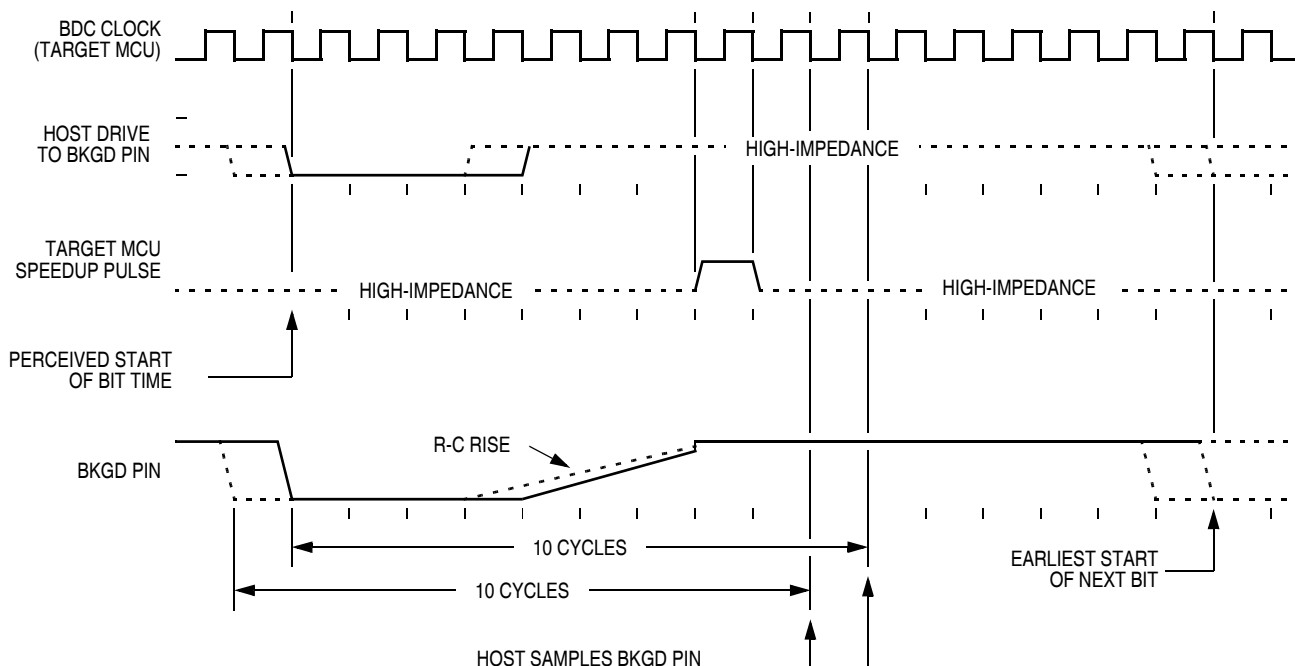


Figure 14-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 14-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time. The host samples the bit level about 10 cycles after starting the bit time.

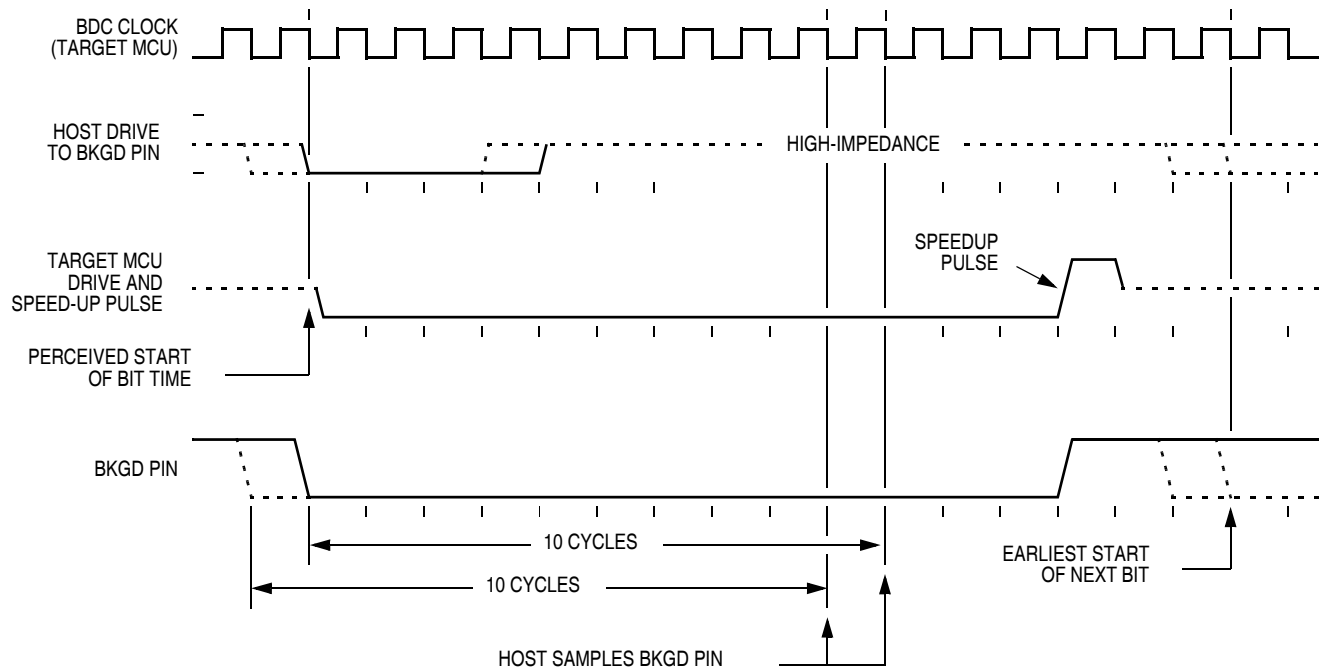


Figure 14-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

14.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Active background mode commands require that the target MCU is currently in the active background mode while non-intrusive commands may be issued at any time whether the target MCU is in active background mode or running a user application program.

Table 14-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

Coding Structure Nomenclature

This nomenclature is used in Table 14-1 to describe the coding structure of the BDC commands.

Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

- / = separates parts of the command
- d = delay 16 target BDC clock cycles
- AAAA = a 16-bit address in the host-to-target direction
- RD = 8 bits of read data in the target-to-host direction
- WD = 8 bits of write data in the host-to-target direction
- RD16 = 16 bits of read data in the target-to-host direction
- WD16 = 16 bits of write data in the host-to-target direction
- SS = the contents of BDCSCR in the target-to-host direction (STATUS)
- CC = 8 bits of write data for BDCSCR in the host-to-target direction (CONTROL)
- RBKP = 16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register)
- WBKP = 16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register)

Table 14-1. BDC Command Summary

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a ¹	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	Non-intrusive	D5/d	Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
ACK_DISABLE	Non-intrusive	D6/d	Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
BACKGROUND	Non-intrusive	90/d	Enter active background mode if enabled (ignore if ENBDM bit equals 0)
READ_STATUS	Non-intrusive	E4/SS	Read BDC status from BDCSCR
WRITE_CONTROL	Non-intrusive	C4/CC	Write BDC controls in BDCSCR
READ_BYTE	Non-intrusive	E0/AAAA/d/RD	Read a byte from target memory
READ_BYTE_WS	Non-intrusive	E1/AAAA/d/SS/RD	Read a byte and report status
READ_LAST	Non-intrusive	E8/SS/RD	Re-read byte from address just read and report status
WRITE_BYTE	Non-intrusive	C0/AAAA/WD/d	Write a byte to target memory
WRITE_BYTE_WS	Non-intrusive	C1/AAAA/WD/d/SS	Write a byte and report status
READ_BKPT	Non-intrusive	E2/RBKP	Read BDCBKPT breakpoint register
WRITE_BKPT	Non-intrusive	C2/WBKP	Write BDCBKPT breakpoint register
GO	Active BDM	08/d	Go to execute the user application program starting at the address currently in the PC
TRACE1	Active BDM	10/d	Trace 1 user instruction at the address in the PC, then return to active background mode
TAGGO	Active BDM	18/d	Same as GO but enable external tagging (HCS08 devices have no external tagging pin)
READ_A	Active BDM	68/d/RD	Read accumulator (A)
READ_CCR	Active BDM	69/d/RD	Read condition code register (CCR)
READ_PC	Active BDM	6B/d/RD16	Read program counter (PC)
READ_HX	Active BDM	6C/d/RD16	Read H and X register pair (H:X)
READ_SP	Active BDM	6F/d/RD16	Read stack pointer (SP)
READ_NEXT	Active BDM	70/d/RD	Increment H:X by one then read memory byte located at H:X
READ_NEXT_WS	Active BDM	71/d/SS/RD	Increment H:X by one then read memory byte located at H:X. Report status and data.
WRITE_A	Active BDM	48/WD/d	Write accumulator (A)
WRITE_CCR	Active BDM	49/WD/d	Write condition code register (CCR)
WRITE_PC	Active BDM	4B/WD16/d	Write program counter (PC)
WRITE_HX	Active BDM	4C/WD16/d	Write H and X register pair (H:X)
WRITE_SP	Active BDM	4F/WD16/d	Write stack pointer (SP)
WRITE_NEXT	Active BDM	50/WD/d	Increment H:X by one, then write memory byte located at H:X
WRITE_NEXT_WS	Active BDM	51/WD/d/SS	Increment H:X by one, then write memory byte located at H:X. Also report status.

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

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

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

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

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

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

14.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

14.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user's memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in [Section 14.3.6, "Hardware Breakpoints."](#)

14.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

14.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

the host must perform $((8 - \text{CNT}) - 1)$ dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see [Section 14.3.5, “Trigger Modes”](#)), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When $\text{ARM} = 0$, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

14.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

14.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGTR register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

14.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTR register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTR register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTR chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGCR register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGSR. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGCR.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

A-Only — Trigger when the address matches the value in comparator A

A OR B — Trigger when the address matches either the value in comparator A or the value in comparator B

A Then B — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

A AND B Data (Full Mode) — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

A AND NOT B Data (Full Mode) — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

Event-Only B (Store Data) — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

A Then Event-Only B (Store Data) — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

Inside Range ($A \leq \text{Address} \leq B$) — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

Outside Range ($\text{Address} < A$ or $\text{Address} > B$) — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

14.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGCR register may be set to 1 to allow any of the trigger conditions described in [Section 14.3.5, “Trigger Modes,”](#) to be used to generate a hardware breakpoint request to the CPU. TAG in DBGCR controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

14.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

14.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

14.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

 = Unimplemented or Reserved

Figure 14-5. BDC Status and Control Register (BDCSCR)

Table 14-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	Enable BDM (Permit Active Background Mode) — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	Background Mode Active Status — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	BDC Breakpoint Enable — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	Force/Tag Select — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	Select Source for BDC Communications Clock — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

Table 14-2. BDCSCR Register Field Descriptions (continued)

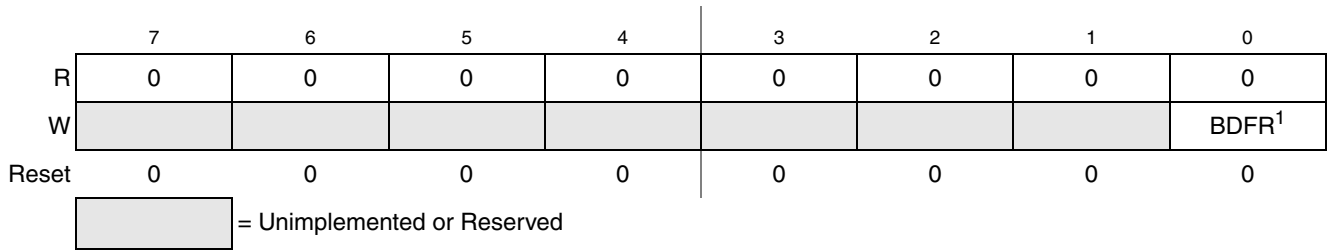
Field	Description
2 WS	Wait or Stop Status — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands. 0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active) 1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode
1 WSF	Wait or Stop Failure Status — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.) 0 Memory access did not conflict with a wait or stop instruction 1 Memory access command failed because the CPU entered wait or stop mode
0 DVF	Data Valid Failure Status — This status bit is not used in the MC9S08QA4 Series because it does not have any slow access memory. 0 Memory access did not conflict with a slow memory access 1 Memory access command failed because CPU was not finished with a slow memory access

14.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to [Section 14.2.4, “BDC Hardware Breakpoint.”](#)

14.4.2 System Background Debug Force Reset Register (SBDFR)

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



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

Figure 14-6. System Background Debug Force Reset Register (SBDFR)

Table 14-3. SBDFR Register Field Description

Field	Description
0 BDFR	Background Debug Force Reset — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program.

14.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

14.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

14.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

14.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
W								
Reset	0	0	0	0	0	0	0	0

Figure 14-7. Debug Control Register (DBGC)

Table 14-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	Debug Module Enable — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure. 0 DBG disabled 1 DBG enabled
6 ARM	Arm Control — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN. 0 Debugger not armed 1 Debugger armed
5 TAG	Tag/Force Select — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect. 0 CPU breaks requested as force type requests 1 CPU breaks requested as tag type requests
4 BRKEN	Break Enable — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests. 0 CPU break requests not enabled 1 Triggers cause a break request to the CPU
3 RWA	R/W Comparison Value for Comparator A — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A. 0 Comparator A can only match on a write cycle 1 Comparator A can only match on a read cycle
2 RWAEN	Enable R/W for Comparator A — Controls whether the level of R/W is considered for a comparator A match. 0 R/W is not used in comparison A 1 R/W is used in comparison A
1 RWB	R/W Comparison Value for Comparator B — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B. 0 Comparator B can match only on a write cycle 1 Comparator B can match only on a read cycle
0 RWBEN	Enable R/W for Comparator B — Controls whether the level of R/W is considered for a comparator B match. 0 R/W is not used in comparison B 1 R/W is used in comparison B

14.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

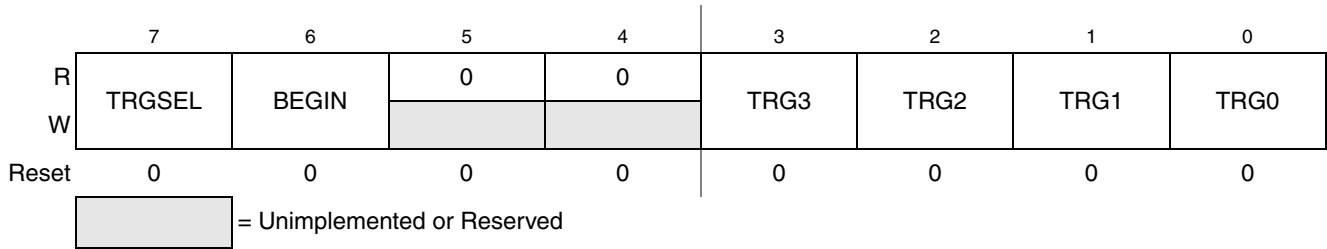


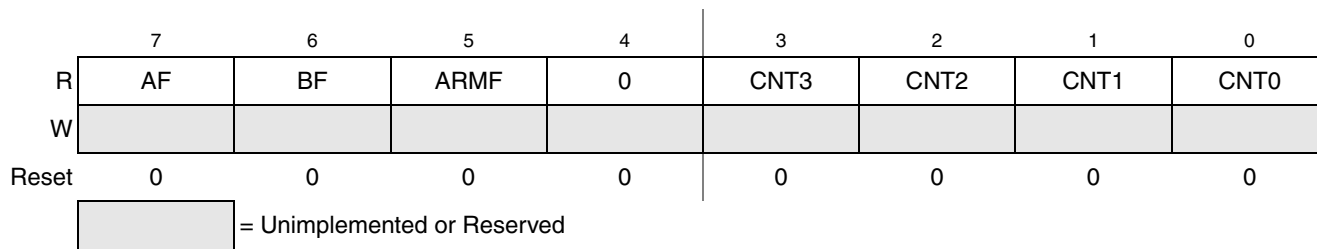
Figure 14-8. Debug Trigger Register (DBGT)

Table 14-5. DBGT Register Field Descriptions

Field	Description
7 TRGSEL	Trigger Type — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed. 0 Trigger on access to compare address (force) 1 Trigger if opcode at compare address is executed (tag)
6 BEGIN	Begin/End Trigger Select — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces. 0 Data stored in FIFO until trigger (end trace) 1 Trigger initiates data storage (begin trace)
3:0 TRG[3:0]	Select Trigger Mode — Selects one of nine triggering modes, as described below. 0000 A-only 0001 A OR B 0010 A Then B 0011 Event-only B (store data) 0100 A then event-only B (store data) 0101 A AND B data (full mode) 0110 A AND NOT B data (full mode) 0111 Inside range: $A \leq \text{address} \leq B$ 1000 Outside range: $\text{address} < A$ or $\text{address} > B$ 1001 – 1111 (No trigger)

14.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.


Figure 14-9. Debug Status Register (DBGS)
Table 14-6. DBGS Register Field Descriptions

Field	Description
7 AF	Trigger Match A Flag — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	Trigger Match B Flag — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	Arm Flag — While DBGEN = 1, this status bit is a read-only image of ARM in DBG. This bit is set by writing 1 to the ARM control bit in DBG (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBG. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	FIFO Valid Count — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8



How to Reach Us:

Home Page:

www.freescale.com

E-mail:

support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

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