



# MC9RS08LA8

## Reference Manual

### ***RS08 Microcontrollers***

#### **Related Documentation:**

---

- **MC9RS08LA8 (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>

MC9RS08LA8RM  
Rev. 1  
10/2008

[freescale.com](http://www.freescale.com)





# MC9RS08LA8 Features

## 8-Bit RS08 Central Processor Unit (CPU)

---

- Up to 20 MHz CPU at 2.7 V to 5.5 V across temperature range of  $-40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$
- Subset of HC08 instruction set with added BGND instruction

## On-Chip Memory

---

- 8 KB flash read/program/erase over full operating voltage and temperature
- 256-byte random-access memory (RAM)
- Security circuitry to prevent unauthorized access to flash contents

## Power-Saving Modes

---

- Wait and stop

## Clock Source Options

---

- Oscillator (XOSC) — Loop-control Pierce oscillator; crystal or ceramic resonator range of 31.25 kHz to 39.0625 kHz or 1 MHz to 16 MHz
- Internal Clock Source (ICS) — Internal clock source module containing a frequency-locked-loop (FLL) controlled by internal or external reference; supports bus frequencies up 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

---

- **LCD** — Up to  $8 \times 21$  or  $4 \times 25$  segments; Compatible with 5 V or 3 V LCD glass displays using on-chip charge pump; functional in wait, stop modes for very low power LCD operation; frontplane and backplane pins multiplexed with GPIO functions; selectable frontplane and backplane configurations
- **ADC** — 6-channel, 10-bit resolution; 2.5  $\mu\text{s}$  conversion time; automatic compare function; 1.7 mV/ $^{\circ}\text{C}$  temperature sensor; internal bandgap reference channel; operation in stop; fully functional from 2.7 V to 5.5 V.
- **TPM** — One 2-channel 16-bit timer/pulse-width modulator (TPM) module;
- **SCI** — One 2-channel serial communications interface module with optional 13-bit break; LIN extensions
- **SPI** — One serial peripheral interface module in 8-bit data length mode with a receive data buffer hardware match function
- **ACMP** — Analog comparator with option to compare to internal reference
- **MTIM** — One 8-bit modulo timer
- **KBI** — 8-pin keyboard interrupt module

## Input/Output

---

- 33 GPIOs including one output-only pin and one input-only pin.
- Hysteresis and configurable pullup device on all input pins; configurable slew rate and drive strength on all output pins.

## Package Options

---

- 48-pin QFN
- 48-pin LQFP



---

# MC9RS08LA8 MCU Series Reference Manual

Covers: MC9RS08LA8

MC9RS08LA8  
Rev. 1  
10/2008

# 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.

<b>Revision Number</b>	<b>Revision Date</b>	<b>Description of Changes</b>
1	10/6/2008	Initial public released

This product incorporates SuperFlash<sup>®</sup> technology licensed from SST.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc.  
© Freescale Semiconductor, Inc., 2008. All rights reserved.

## List of Chapters

<b>Chapter Number</b>	<b>Title</b>	<b>Page</b>
<b>Chapter 1</b>	<b>Device Overview</b> .....	<b>19</b>
<b>Chapter 2</b>	<b>Pins and Connections</b> .....	<b>23</b>
<b>Chapter 3</b>	<b>Modes of Operation</b> .....	<b>31</b>
<b>Chapter 4</b>	<b>Memory</b> .....	<b>37</b>
<b>Chapter 5</b>	<b>Resets, Interrupts, and System Configuration</b> .....	<b>53</b>
<b>Chapter 6</b>	<b>Parallel Input/Output</b> .....	<b>65</b>
<b>Chapter 7</b>	<b>Keyboard Interrupt (RS08KBIV1)</b> .....	<b>81</b>
<b>Chapter 8</b>	<b>Central Processor Unit (RS08CPUV1)</b> .....	<b>89</b>
<b>Chapter 9</b>	<b>Internal Clock Source (RS08ICSV1)</b> .....	<b>105</b>
<b>Chapter 10</b>	<b>Liquid Crystal Display Module (S08LCDV2)</b> .....	<b>119</b>
<b>Chapter 11</b>	<b>Analog Comparator (RS08ACMPV1)</b> .....	<b>161</b>
<b>Chapter 12</b>	<b>10-Bit Analog-to-Digital Converter (RS08ADC10V1)</b> .....	<b>167</b>
<b>Chapter 13</b>	<b>16-Bit Timer/PWM (RS08TPMV2)</b> .....	<b>193</b>
<b>Chapter 14</b>	<b>Modulo Timer (RS08MTIMV1)</b> .....	<b>217</b>
<b>Chapter 15</b>	<b>Serial Communications Interface (S08SCIV4)</b> .....	<b>227</b>
<b>Chapter 16</b>	<b>Serial Peripheral Interface (S08SPIV3)</b> .....	<b>249</b>
<b>Chapter 17</b>	<b>Development Support</b> .....	<b>265</b>





# Contents

Section Number	Title	Page
<b>Chapter 1</b>		
<b>Device Overview</b>		
1.1	Introduction .....	19
1.2	MCU Block Diagram .....	20
1.3	System Clock Distribution .....	21
<b>Chapter 2</b>		
<b>Pins and Connections</b>		
2.1	Introduction .....	23
2.2	Device Pin Assignment .....	23
2.3	Recommended System Connections .....	24
2.3.1	Power ( $V_{DD}$ , $V_{SS}$ , $V_{DDAD}$ , $V_{SSAD}$ ) .....	26
2.3.2	Oscillator (XTAL, EXTAL) .....	26
2.3.3	PTB2/RESET/ $V_{PP}$ Pin .....	26
2.3.4	PTC6/ACMPO/BKGD/MS .....	27
2.3.5	LCD Pins .....	27
2.3.6	General-Purpose I/O and Peripheral Ports .....	28
<b>Chapter 3</b>		
<b>Modes of Operation</b>		
3.1	Introduction .....	31
3.2	Features .....	31
3.3	Run Mode .....	31
3.4	Active Background Mode .....	31
3.5	Wait Mode .....	32
3.6	Stop Modes .....	33
3.6.1	Active BDM Enabled in Stop Mode .....	34
3.6.2	LVD Enabled in Stop Mode .....	34
<b>Chapter 4</b>		
<b>Memory</b>		
4.1	MC9RS08LA8 Memory Map .....	37
4.2	Unimplemented Memory .....	38
4.3	Indexed/Indirect Addressing .....	39
4.3.1	Accessing High-Page RAM .....	39
4.3.2	Accessing High-Page Register .....	40
4.4	Register Addresses and Bit Assignments .....	41
4.5	RAM (System RAM) .....	46
4.6	Flash .....	46

4.6.1	Features .....	47
4.6.2	Flash Programming Procedure .....	47
4.6.3	Flash Mass Erase Operation .....	48
4.6.4	Flash Security .....	48
4.6.5	Flash Registers and Control Bits .....	49
4.6.6	Flash Options Register (FOPT and NVOPT) .....	49
4.6.7	Flash Control Register (FLCR) .....	50
4.6.8	Page Select Register (PAGESEL) .....	51

## Chapter 5 Resets, Interrupts, and System Configuration

5.1	Introduction .....	53
5.2	Features .....	53
5.3	MCU Reset .....	53
5.4	Computer Operating Properly (COP) Watchdog .....	54
5.5	Interrupts .....	55
5.6	Low-Voltage Detect (LVD) System .....	55
5.6.1	Power-On Reset Operation .....	55
5.6.2	LVD Reset Operation .....	55
5.6.3	LVD Interrupt Operation .....	55
5.7	Real-Time Interrupt (RTI) .....	56
5.8	Reset, Interrupt, and System Control Registers and Control Bits .....	56
5.8.1	System Reset Status Register (SRS) .....	56
5.8.2	System Options Register (SOPT) .....	57
5.8.3	System Device Identification Register (SDIDH, SDIDL) .....	58
5.8.4	System PMC Real-Time Interrupt Status and Control (SRTISC) .....	60
5.8.5	System Power Management Status and Control 1 Register (SPMSC1) .....	61
5.8.6	System Interrupt Pending Register 1 (SIP1) .....	61
5.8.7	System Interrupt Pending Register 2 (SIP2) .....	62

## Chapter 6 Parallel Input/Output

6.1	Introduction .....	65
6.2	Pin Behavior in Low-Power Modes .....	66
6.3	Parallel I/O and Pin Control Registers .....	66
6.3.1	Port A I/O Registers (PTAD and PTADD) .....	66
6.3.2	Port A Pin Control Registers (PTAPE, PTAPUD, PTADS, PTASE) .....	67
6.3.3	Port B I/O Registers (PTBD and PTBDD) .....	69
6.3.4	Port B Pin Control Registers (PTBPE, PTBPUD, PTBDS, PTBSE) .....	70
6.3.5	Port C I/O Registers (PTCD and PTCDD) .....	71
6.3.6	Port C Pin Control Registers (PTCPE, PTCPU, PTCDS, PTCSE) .....	72
6.3.7	Port D I/O Registers (PTDD and PTDDD) .....	74
6.3.8	Port D Pin Control Registers (PTDPE, PTDPUD, PTDDS, PTDSE) .....	75
6.3.9	Port E I/O Registers (PTED and PTEDD) .....	76
6.3.10	Port E Pin Control Registers (PTEPE, PTEPUD, PTEDS, PTESE) .....	77

## Chapter 7 Keyboard Interrupt (RS08KBIV1)

7.1	Introduction .....	81
7.1.1	Features .....	82
7.1.2	Modes of Operation .....	83
7.1.3	Block Diagram .....	83
7.2	External Signal Description .....	84
7.3	Register Definition .....	84
7.4	Functional Description .....	86
7.4.1	Edge Only Sensitivity .....	86
7.4.2	Edge and Level Sensitivity .....	87
7.4.3	KBI Pullup/Pulldown Device .....	87
7.4.4	KBI Initialization .....	87

## Chapter 8 Central Processor Unit (RS08CPUV1)

8.1	Introduction .....	89
8.2	Programmer's Model and CPU Registers .....	89
8.2.1	Accumulator (A) .....	90
8.2.2	Program Counter (PC) .....	91
8.2.3	Shadow Program Counter (SPC) .....	91
8.2.4	Condition Code Register (CCR) .....	91
8.2.5	Indexed Data Register (D[X]) .....	92
8.2.6	Index Register (X) .....	92
8.2.7	Page Select Register (PAGESEL) .....	93
8.3	Addressing Modes .....	93
8.3.1	Inherent Addressing Mode (INH) .....	93
8.3.2	Relative Addressing Mode (REL) .....	93
8.3.3	Immediate Addressing Mode (IMM) .....	94
8.3.4	Tiny Addressing Mode (TNY) .....	94
8.3.5	Short Addressing Mode (SRT) .....	95
8.3.6	Direct Addressing Mode (DIR) .....	95
8.3.7	Extended Addressing Mode (EXT) .....	95
8.3.8	Indexed Addressing Mode (IX, Implemented by Pseudo Instructions) .....	95
8.4	Special Operations .....	95
8.4.1	Reset Sequence .....	96
8.4.2	Interrupts .....	96
8.4.3	Wait and Stop Mode .....	96
8.4.4	Active Background Mode .....	96
8.5	Summary Instruction Table .....	97

## Chapter 9 Internal Clock Source (RS08ICSV1)

9.1	Introduction .....	105
9.1.1	Features .....	107

9.1.2	Modes of Operation .....	107
9.1.3	Block Diagram .....	108
9.2	External Signal Description .....	109
9.3	Register Definition .....	109
9.3.1	ICS Control Register 1 (ICSC1) .....	109
9.3.2	ICS Control Register 2 (ICSC2) .....	111
9.3.3	ICS Trim Register (ICSTRM) .....	112
9.3.4	ICS Status and Control (ICSSC) .....	112
9.4	Functional Description .....	113
9.4.1	Operational Modes .....	113
9.4.2	Mode Switching .....	115
9.4.3	Bus Frequency Divider .....	115
9.4.4	Low Power Bit Usage .....	115
9.4.5	Internal Reference Clock .....	116
9.4.6	Optional External Reference Clock .....	116
9.4.7	Fixed Frequency Clock .....	117

## Chapter 10 Liquid Crystal Display Module (S08LCDV2)

10.1	Introduction .....	119
10.1.1	LCD Clock Sources .....	119
10.1.2	LCD Modes of Operation .....	119
10.1.3	LCD Power Supply Configurations .....	119
10.1.4	Open Drain Mode .....	120
10.1.5	Read Only Register Operations .....	120
10.1.6	Features .....	123
10.1.7	Modes of Operation .....	123
10.1.8	Block Diagram .....	125
10.2	External Signal Description .....	125
10.2.1	LCD[29:0] .....	126
10.2.2	V <sub>LL1</sub> , V <sub>LL2</sub> , V <sub>LL3</sub> .....	126
10.2.3	V <sub>cap1</sub> , V <sub>cap2</sub> .....	126
10.3	Register Definition .....	126
10.3.1	LCD Control Register 0 (LCDC0) .....	126
10.3.2	LCD Control Register 1 (LCDC1) .....	127
10.3.3	LCD Voltage Supply Register (LCDSUPPLY) .....	128
10.3.4	LCD Blink Control Register (LCDBCTL) .....	129
10.3.5	LCD Status Register (LCDS) .....	130
10.3.6	LCD Pin Enable Registers 0–3 (LCDPEN0–LCDPEN37) .....	131
10.3.7	Backplane Enable Registers 0–3 (BPEN0–BPEN3) .....	131
10.3.8	LCD Waveform Registers (LCDWF[28:0]) .....	132
10.4	Functional Description .....	135
10.4.1	LCD Driver Description .....	136
10.4.2	LCDWF Registers .....	144
10.4.3	LCD Display Modes .....	144

10.4.4	LCD Charge Pump, Voltage Divider, and Power Supply Operation .....	146
10.4.5	Resets .....	149
10.4.6	Interrupts .....	150
10.5	Initialization Section .....	150
10.5.1	Initialization Sequence .....	150
10.5.2	Initialization Examples .....	151
10.6	Application Information .....	155
10.6.1	LCD Seven Segment Example Description .....	156
10.6.2	LCD Contrast Control .....	159

## Chapter 11 Analog Comparator (RS08ACMPV1)

11.1	Introduction .....	161
11.1.1	Features .....	163
11.1.2	Modes of Operation .....	163
11.1.3	Block Diagram .....	163
11.2	External Signal Description .....	164
11.3	Register Definition .....	164
11.3.1	ACMP Status and Control Register (ACMPSC) .....	165
11.4	Functional Description .....	165

## Chapter 12 10-Bit Analog-to-Digital Converter (RS08ADC10V1)

12.1	Introduction .....	167
12.1.1	ADC Channel Assignments .....	167
12.1.2	Alternate Clock .....	168
12.1.3	Hardware Trigger .....	169
12.1.4	Temperature Sensor .....	169
12.1.5	Features .....	170
12.1.6	Block Diagram .....	170
12.2	External Signal Description .....	171
12.2.1	Analog Power ( $V_{DDAD}$ ) .....	172
12.2.2	Analog Ground ( $V_{SSAD}$ ) .....	172
12.2.3	Voltage Reference High ( $V_{REFH}$ ) .....	172
12.2.4	Voltage Reference Low ( $V_{REFL}$ ) .....	172
12.2.5	Analog Channel Inputs (ADx) .....	172
12.3	Register Definition .....	172
12.3.1	Status and Control Register 1 (ADCSC1) .....	172
12.3.2	Status and Control Register 2 (ADCSC2) .....	174
12.3.3	Data Result High Register (ADCRH) .....	175
12.3.4	Data Result Low Register (ADCRL) .....	175
12.3.5	Compare Value High Register (ADCCVH) .....	175
12.3.6	Compare Value Low Register (ADCCVL) .....	176
12.3.7	Configuration Register (ADCCFG) .....	176
12.3.8	Pin Control 1 Register (APCTL1) .....	177

12.3.9	Pin Control 2 Register (APCTL2)	178
12.3.10	Pin Control 3 Register (APCTL3)	179
12.4	Functional Description	180
12.4.1	Clock Select and Divide Control	181
12.4.2	Input Select and Pin Control	181
12.4.3	Hardware Trigger	181
12.4.4	Conversion Control	181
12.4.5	Automatic Compare Function	184
12.4.6	MCU Wait Mode Operation	184
12.4.7	MCU Stop Mode Operation	185
12.5	Initialization Information	185
12.5.1	ADC Module Initialization Example	186
12.6	Application Information	187
12.6.1	External Pins and Routing	188
12.6.2	Sources of Error	189

## Chapter 13 16-Bit Timer/PWM (RS08TPMV2)

13.1	Introduction	193
13.1.1	Features	195
13.1.2	Modes of Operation	195
13.1.3	Block Diagram	196
13.2	Signal Description	198
13.2.1	Detailed Signal Descriptions	198
13.3	Register Definition	202
13.3.1	TPM Status and Control Register (TPMSC)	202
13.3.2	TPM-Counter Registers (TPMCNTH:TPMCNTL)	203
13.3.3	TPM Counter Modulo Registers (TPMMODH:TPMMODL)	204
13.3.4	TPM Channel n Status and Control Register (TPMCnSC)	205
13.3.5	TPM Channel Value Registers (TPMCnVH:TPMCnVL)	206
13.4	Functional Description	208
13.4.1	Counter	208
13.4.2	Channel Mode Selection	210
13.5	Reset Overview	213
13.5.1	General	213
13.5.2	Description of Reset Operation	213
13.6	Interrupts	213
13.6.1	General	213
13.6.2	Description of Interrupt Operation	214

## Chapter 14 Modulo Timer (RS08MTIMV1)

14.1	Introduction	217
14.1.1	Features	219
14.1.2	Modes of Operation	219

14.1.3	Block Diagram .....	220
14.2	External Signal Description .....	220
14.3	Register Definition .....	220
14.3.1	MTIM Status and Control Register (MTIMSC) .....	221
14.3.2	MTIM Clock Configuration Register (MTIMCLK) .....	222
14.3.3	MTIM Counter Register (MTIMCNT) .....	222
14.3.4	MTIM Modulo Register (MTIMMOD) .....	223
14.4	Functional Description .....	224
14.4.1	MTIM Operation Example .....	225

## Chapter 15 Serial Communications Interface (S08SCIV4)

15.1	Introduction .....	227
15.1.1	SCI Working In Fixed Mode .....	227
15.1.2	SCI Working In Mixed Mode .....	227
15.1.3	SOPT Register Setting .....	228
15.1.4	Open Drain Mode .....	229
15.1.5	Features .....	231
15.1.6	Modes of Operation .....	231
15.1.7	Block Diagram .....	231
15.2	Register Definition .....	234
15.2.1	SCI Baud Rate Registers (SCIBDH, SCIBDL) .....	234
15.2.2	SCI Control Register 1 (SCIC1) .....	235
15.2.3	SCI Control Register 2 (SCIC2) .....	236
15.2.4	SCI Status Register 1 (SCIS1) .....	237
15.2.5	SCI Status Register 2 (SCIS2) .....	239
15.2.6	SCI Control Register 3 (SCIC3) .....	240
15.2.7	SCI Data Register (SCID) .....	241
15.3	Functional Description .....	241
15.3.1	Baud Rate Generation .....	241
15.3.2	Transmitter Functional Description .....	242
15.3.3	Receiver Functional Description .....	243
15.3.4	Interrupts and Status Flags .....	245
15.3.5	Additional SCI Functions .....	246

## Chapter 16 Serial Peripheral Interface (S08SPIV3)

16.1	Introduction .....	249
16.1.1	Open Drain Mode .....	249
16.1.2	Features .....	251
16.1.3	Block Diagrams .....	251
16.1.4	SPI Baud Rate Generation .....	253
16.2	External Signal Description .....	254
16.2.1	SPSCK — SPI Serial Clock .....	254
16.2.2	MOSI — Master Data Out, Slave Data In .....	254

16.2.3	MISO — Master Data In, Slave Data Out .....	254
16.2.4	$\overline{SS}$ — Slave Select .....	254
16.3	Modes of Operation .....	255
16.3.1	SPI in Stop Modes .....	255
16.4	Register Definition .....	255
16.4.1	SPI Control Register 1 (SPIC1) .....	255
16.4.2	SPI Control Register 2 (SPIC2) .....	256
16.4.3	SPI Baud Rate Register (SPIBR) .....	257
16.4.4	SPI Status Register (SPIS) .....	258
16.4.5	SPI Data Register (SPID) .....	259
16.5	Functional Description .....	260
16.5.1	SPI Clock Formats .....	260
16.5.2	SPI Interrupts .....	263
16.5.3	Mode Fault Detection .....	263

## Chapter 17 Development Support

17.1	Introduction .....	265
17.2	Features .....	265
17.3	RS08 Background Debug Controller (BDC) .....	266
17.3.1	BKGD Pin Description .....	267
17.3.2	Communication Details .....	267
17.3.3	SYNC and Serial Communication Timeout .....	270
17.4	BDC Registers and Control Bits .....	271
17.4.1	BDC Status and Control Register (BDCSCR) .....	271
17.4.2	BDC Breakpoint Match Register .....	272
17.5	RS08 BDC Commands .....	273
17.5.1	BDC_RESET .....	276
17.5.2	BACKGROUND .....	276
17.5.3	READ_STATUS .....	277
17.5.4	WRITE_CONTROL .....	277
17.5.5	READ_BYTE .....	278
17.5.6	READ_BYTE_WS .....	279
17.5.7	WRITE_BYTE .....	279
17.5.8	WRITE_BYTE_WS .....	280
17.5.9	READ_BKPT .....	280
17.5.10	WRITE_BKPT .....	280
17.5.11	GO .....	281
17.5.12	TRACE1 .....	281
17.5.13	READ_BLOCK .....	281
17.5.14	WRITE_BLOCK .....	282
17.5.15	READ_A .....	282
17.5.16	WRITE_A .....	282
17.5.17	READ_CCR_PC .....	283
17.5.18	WRITE_CCR_PC .....	283



---

17.5.19	READ_SPC	284
17.5.20	WRITE_SPC	284
17.6	BDC Hardware Breakpoint	284
17.7	BDM in Stop and Wait Modes	285
17.8	BDC Command Execution	285



# Chapter 1

## Device Overview

### 1.1 Introduction

The MC9RS08LA8 MCU is member of the low-cost RS08 family of 8-bit microcontroller units (MCUs). The device is composed of standard on-chip modules including a very small and highly efficient RS08 CPU core, on-chip RAM, nonvolatile memory, a 16-bit TPM, an 8-bit modulo timer (MTIM), a 2-channel serial communications interface (SCI), a serial peripheral interface (SPI), a 6-channel 10-bit analog-to-digital converter (ADC), an analog comparator (ACMP), and a liquid crystal display module (LCD).

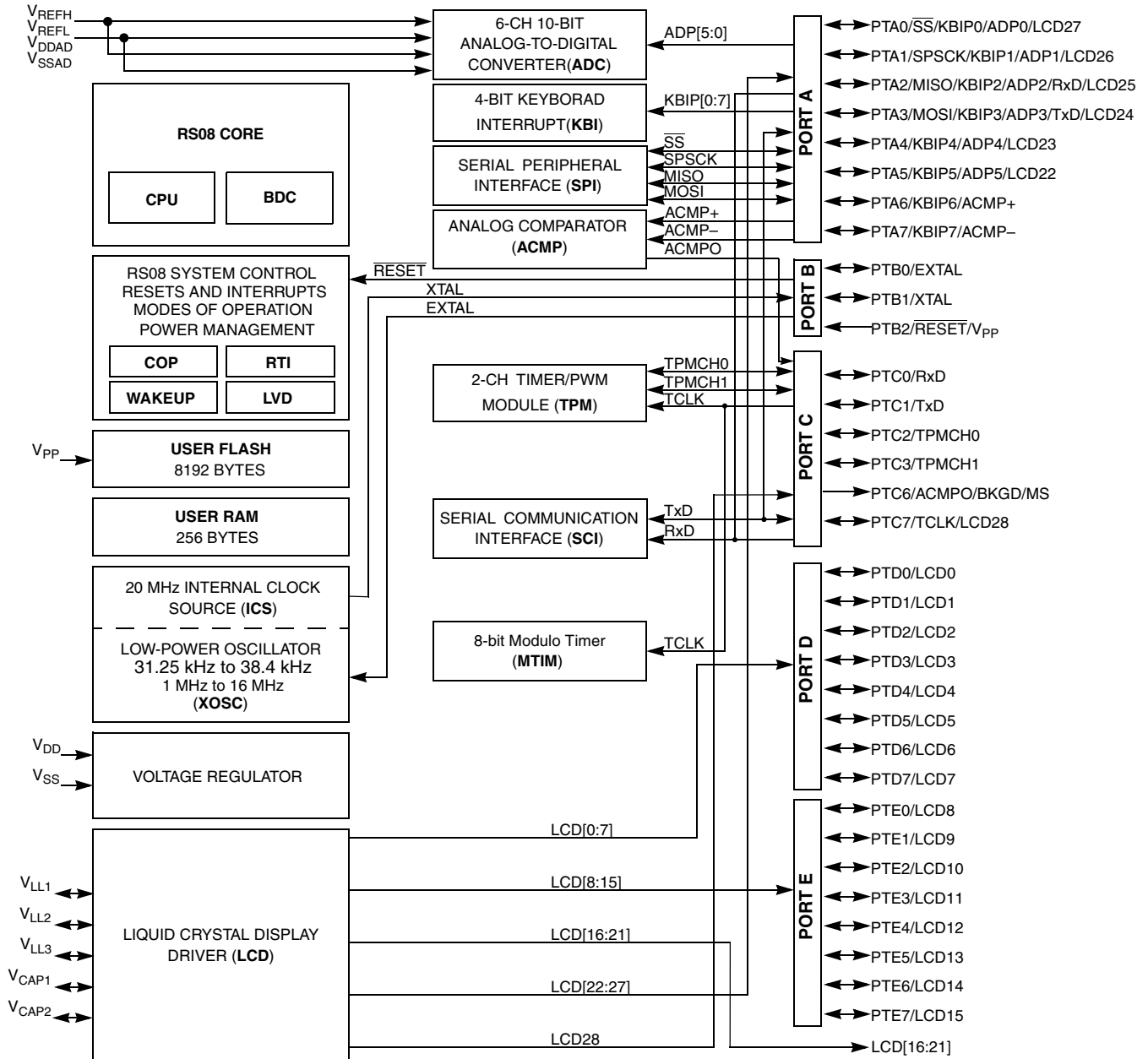
Table 1-1 summarizes the peripheral availability per package type for the devices available in the MC9RS08LA8.

**Table 1-1. Devices in the MC9RS08LA8**

Feature	Device
	MC9RS08LA8
Package	48-pin
FLASH	8,192
RAM	256
RTI	yes
ACMP	1
ADC	10-bit 6-ch
KBI	8-ch
SCI	2-ch
SPI	yes
TPM	2-ch
MTIM	1
LCD	8 × 21 4 × 25
I/O pins	33
Package types	48 QFN 48 LQFP

## 1.2 MCU Block Diagram

The block diagram in Figure 1-1 shows the structure of the MC9RS08LA8 MCU.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 1-1. MC9RS08LA8 Block Diagram**

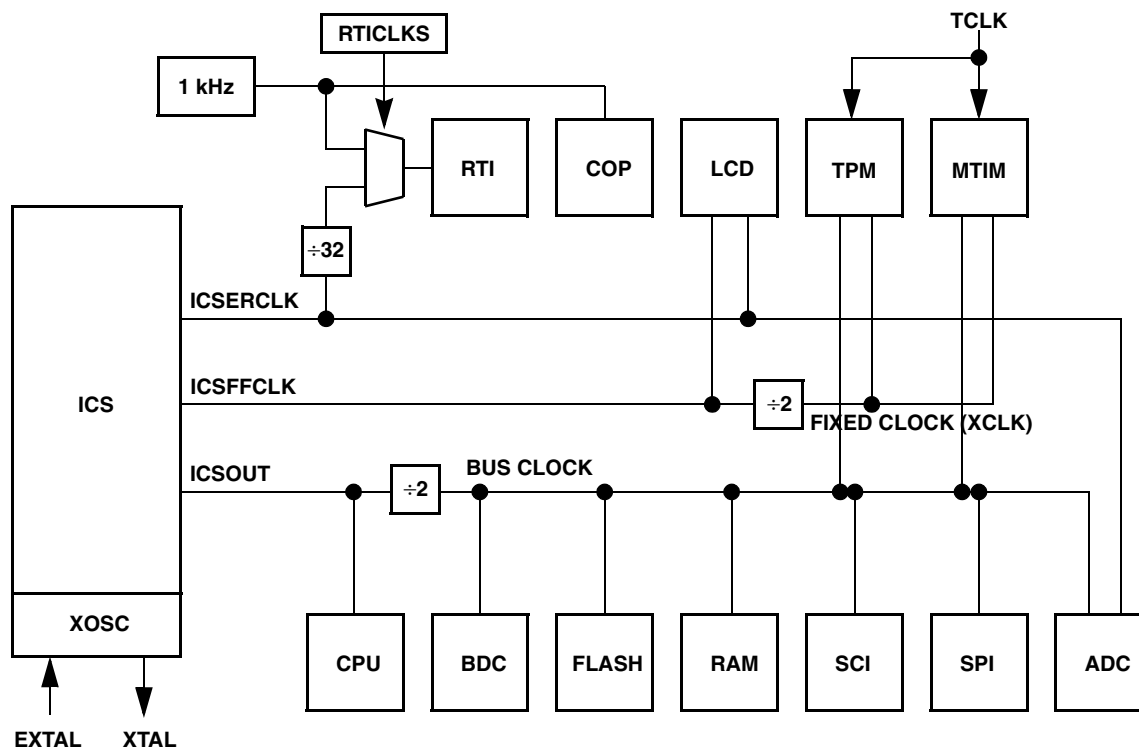
Table 1-2 lists the functional versions of the on-chip modules.

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

Module		Version
Analog Comparator	(ACMP)	1
Analog-to-Digital Converter	(ADC10)	1
Central Processing Unit	(CPU)	1
Keyboard Interrupt	(KBI)	1
Internal Clock Source	(ICS)	1
Oscillator	(XOSC)	1
Serial Communications Interface	(SCI)	4
Serial Peripheral Interface	(SPI)	3
Timer Pulse-Width Modulator	(TPM)	2
Modulo Timer	(MTIM)	1
Liquid Crystal Display Module	(LCD)	1

### 1.3 System Clock Distribution

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



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

The ICS supplies the following clock sources:

- ICSOUT — This clock source is used as the CPU clock and is divided by 2 to generate the peripheral BUS CLOCK. Control bits in the ICS control registers determine which of three clock sources is connected:
  - Internal reference clock
  - External reference clock
  - Frequency-locked loop (FLL) outputSee [Chapter 9, “Internal Clock Source \(RS08ICSV1\)”](#) for details on configuring the ICSOUT clock.
- ICSECLK — This is the external reference clock and can be selected as the alternate clock for the ADC and LCD modules. [Section 9.4.6, “Optional External Reference Clock”](#) explains the ICSECLK in more detail. See [Chapter 12, “10-Bit Analog-to-Digital Converter \(RS08ADC10V1\)”](#) and [Chapter 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#) for more information regarding the use of ICSECLK with these modules.
- ICSFFCLK — This clock can be selected as the alternate clock for LCD module. It is divided by 2 to generate the FIXED CLOCK (XCLK). The FIXED CLOCK can be selected as clock source for the TPM and MTIM modules. The frequency of the ICSFFCLK is determined by the settings of the ICS. See the [Section 9.4.7, “Fixed Frequency Clock”](#) for details.
- TCLK — TCLK is the optional external clock source for the TPM and MTIM modules. The TCLK must be limited to 1/4th the frequency of the bus clock for synchronization. See [Chapter 13, “16-Bit Timer/PWM \(RS08TPMV2\)”](#) and [Chapter 14, “Modulo Timer \(RS08MTIMV1\)”](#) for more details.
- 1 kHz — This clock is generated from an internal low power oscillator that is completely independent of the ICS module. The clock can be selected as the clock source to the COP module. It can also See [Section 5.4, “Computer Operating Properly \(COP\) Watchdog”](#) for details on using the 1 kHz clock with this module.

# Chapter 2 Pins and Connections

## 2.1 Introduction

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

## 2.2 Device Pin Assignment

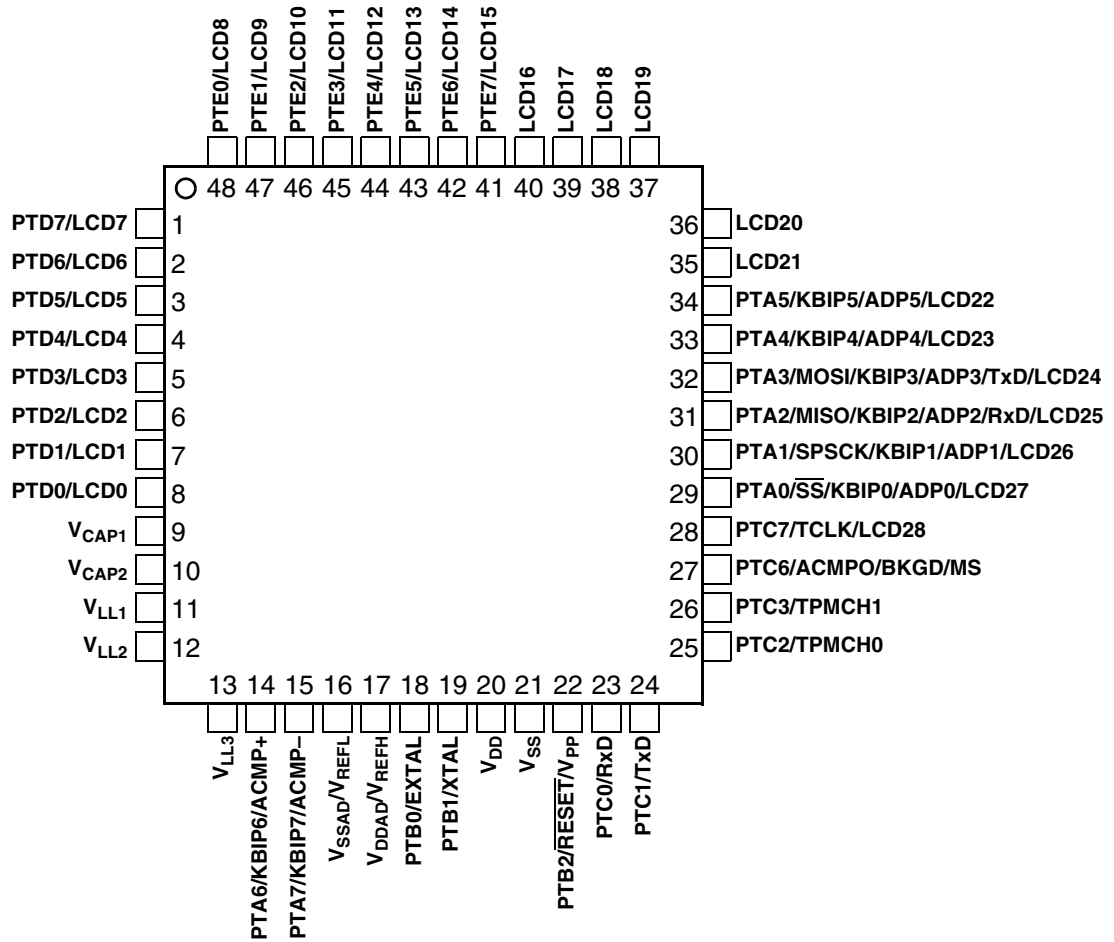
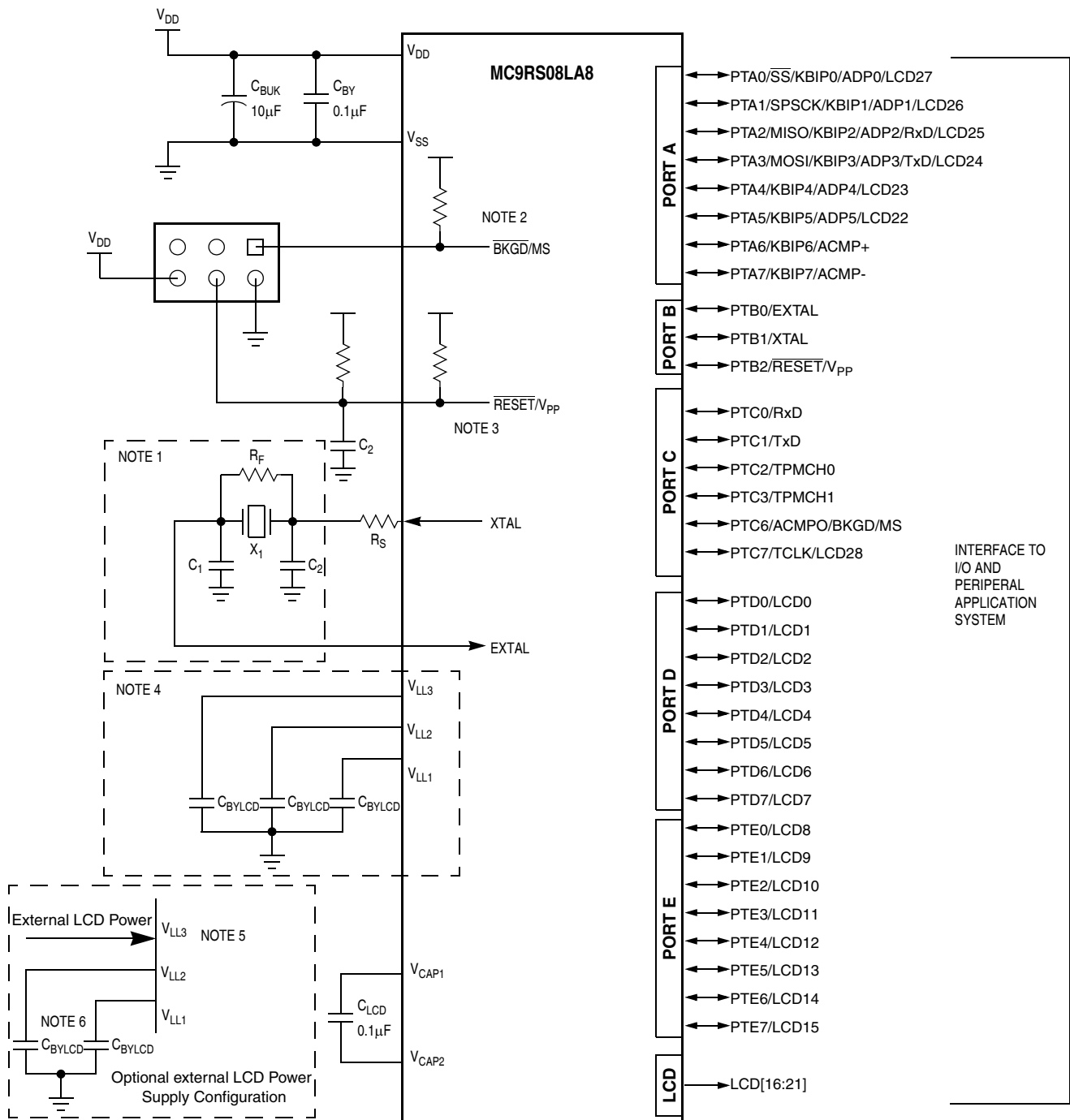


Figure 2-1. MC9RS08LA8 in 48-Pin QFN/LQFP Package

## 2.3 Recommended System Connections

Figure 2-2 shows pin connections that are common to almost all MC9RS08LA8 application systems.





**NOTE**

1. Not required if the internal oscillator is used
2. BKGD/MS is the same pin as PTC6
3. An RC filter on RESET is recommended for EMC-sensitive applications
4.  $V_{LL1}$ ,  $V_{LL2}$ , and  $V_{LL3}$  can be powered internally using  $V_{DD}$  based on LCD software configuration
5.  $V_{LL3}$  can be connected to external LCD power supply based on LCD software configuration
6.  $V_{LL1}$  and  $V_{LL2}$  must be floating when internal resistor network is used. The bypass capacitors are not needed.

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

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

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

Typically, application systems have two separate capacitors across the power pins. In this case, there must be a bulk electrolytic capacitor, such as a 10- $\mu$ F tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1- $\mu$ F ceramic bypass capacitor located as close to the MCU power pins as practical to suppress high-frequency noise.

$V_{DDAD}$  and  $V_{SSAD}$  are the analog power supply pins for the MCU. This voltage source supplies power to the ADC module. A 0.1- $\mu$ F ceramic bypass capacitor must be located as close to the analog power pins as practical to suppress high-frequency noise.

### 2.3.2 Oscillator (XTAL, EXTAL)

Immediately after reset, the MCU uses an internally generated clock provided by the internal clock source (ICS) module. The internal frequency is nominally 16 MHz and the default ICS settings will provide for a 4 MHz bus out of reset.

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

$R_S$  (when used) and  $R_F$  must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally must be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

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

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

### 2.3.3 $\overline{PTB2}/\overline{RESET}/V_{PP}$ Pin

After a power-on reset (POR) into user mode, the  $\overline{RESET}/V_{PP}$  pin defaults to a general-purpose input port pin, PTB2. Setting RSTPE in SOPT configures the pin to be the  $\overline{RESET}$  input pin. After configured as  $\overline{RESET}$ , the pin will remain as  $\overline{RESET}$  until next reset. The  $\overline{RESET}$  pin can be used to reset the MCU from an external source when the pin is driven low. When enabled as the  $\overline{RESET}$  pin (RSTPE=1), the internal pullup device is automatically enabled.

External  $V_{PP}$  voltage (typically 12V, see *MC9RS08LA8 Data Sheet*) is required on this pin when performing flash programming or erasing. The  $V_{PP}$  connection is always connected to the internal flash module regardless of the pin function. To avoid over stressing the flash, external  $V_{PP}$  voltage must be removed and voltage higher than  $V_{DD}$  must be avoided when flash programming or erasing is not taking place.

#### NOTE

This pin does not contain a clamp diode to  $V_{DD}$  and must not be driven above  $V_{DD}$  when flash programming or erasing is not taking place.

### 2.3.4 PTC6/ACMPO/BKGD/MS

During a power-on-reset (POR) or background debug force reset, the PTC6/ACMPO/BKGD/MS pin functions as a mode select pin. Immediately after reset rises the pin functions as the background pin and can be used for background debug communication. While functioning as a background/mode select pin, the pin includes an internal pullup device, input hysteresis, a standard output driver, and no output slew rate control.

The background debug communication function is enabled when BKGDPE in SOPT is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTC6/ACMPO/BKGD/MS pin's alternative pin function.

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

The BKGD pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock could be as fast as the bus clock rate, so there 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.

### 2.3.5 LCD Pins

#### 2.3.5.1 LCD Power ( $V_{LL1}$ , $V_{LL2}$ , $V_{LL3}$ , $V_{CAP1}$ , $V_{CAP2}$ )

The  $V_{LL1}$ ,  $V_{LL2}$ ,  $V_{LL3}$ ,  $V_{CAP1}$ , and  $V_{CAP2}$  pins are dedicated to providing power to the LCD module. For detailed information about LCD these pins see [Chapter 10, "Liquid Crystal Display Module \(S08LCDV2\)"](#).

#### 2.3.5.2 LCD Driver (LCD[0:28])

There are 29 LCD driver pins in MC9RS08LA8. 23 of the LCD driver pins can operate as GPIO and other functions. Immediately after reset, the LCD driver pins are high-impedance. LCD/GPIO pins are

configured as GPIO by default. For detailed information about LCD driver pins see [Chapter 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#).

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

The MC9RS08LA8 MCU supports up to 33 general-purpose I/O pins, which are shared with on-chip peripheral functions (timers, serial I/O, ADC, keyboard interrupts, LCD, etc.).

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

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

Immediately after reset, all pins are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

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

Pin Number	<-- Lowest Priority --> Highest					
48	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4	Alt 5
1	PTD7					LCD7
2	PTD6					LCD6
3	PTD5					LCD5
4	PTD4					LCD4
5	PTD3					LCD3
6	PTD2					LCD2
7	PTD1					LCD1
8	PTD0					LCD0
9						V <sub>CAP1</sub>
10						V <sub>CAP2</sub>
11						V <sub>LL1</sub>
12						V <sub>LL2</sub>
13						V <sub>LL3</sub>
14	PTA6	KBIP6	ACMP+			
15	PTA7	KBIP7	ACMP-			
16				V <sub>SSAD</sub> /V <sub>REFL</sub>		
17				V <sub>DDAD</sub> /V <sub>REFH</sub>		
18	PTB0			EXTAL		
19	PTB1			XTAL		
20				V <sub>DD</sub>		
21				V <sub>SS</sub>		
22	PTB2		RESET	V <sub>PP</sub>		

Table 2-1. Pin Availability by Package Pin-Count (continued)

Pin Number	<-- Lowest Priority --> Highest					
48	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4	Alt 5
23	PTC0		RxD			
24	PTC1		TxD			
25	PTC2		TPMCH0			
26	PTC3		TPMCH1			
27	PTC6	ACMPO	BKGD	MS		
28	PTC7		TCLK			LCD28
29	PTA0	$\overline{SS}$	KBIP0	ADP0		LCD27
30	PTA1	SPSCK	KBIP1	ADP1		LCD26
31	PTA2	MISO	KBIP2	RxD	ADP2	LCD25
32	PTA3	MOSI	KBIP3	TxD	ADP3	LCD24
33	PTA4		KBIP4	ADP4		LCD23
34	PTA5		KBIP5	ADP5		LCD22
35						LCD21
36						LCD20
37						LCD19
38						LCD18
39						LCD17
40						LCD16
41	PTE7					LCD15
42	PTE6					LCD14
43	PTE5					LCD13
44	PTE4					LCD12
45	PTE3					LCD11
46	PTE2					LCD10
47	PTE1					LCD9
48	PTE0					LCD8

**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](#) illustrates the priority if multiple modules are enabled. The highest priority module will have control over the pin. Selecting a higher priority pin function with a lower priority function already enabled will cause spurious edges to the lower priority module. All modules that share a pin must be disabled before enabling another module.



# Chapter 3

## Modes of Operation

### 3.1 Introduction

The operating modes of the MC9RS08LA8 are described in this section. 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
- Run mode:
  - CPU running
  - System clocks running
  - Full voltage regulation is maintained
- Wait mode:
  - CPU halts operation to conserve power
  - System clocks running
  - Full voltage regulation is maintained
- Stop mode: CPU and bus clocks stopped
  - System clocks are stopped;
  - All internal circuits remain powered for fast recovery

### 3.3 Run Mode

This is the normal operating mode for the MC9RS08LA8. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory with execution beginning at the address \$3FFD. A JMP instruction (opcode \$BC) with operand located at \$3FFE–\$3FFF must be programmed for correct reset operation into the vector fetching process as in HC08/S08 families, the user program is responsible for performing a JMP instruction to relocate the program counter to the correct user program start location.

### 3.4 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the RS08 core. The BDC provides the means for analyzing MCU operation during software development.

Active background mode is entered in any of four ways:

- When the BKGD/MS pin is low during POR or immediately after issuing a background debug force reset
- When a BACKGROUND command is received through the BKGD pin
- When a BGND instruction is executed
- When a BDC breakpoint is encountered

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
  - 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 user application program into the flash program memory before the MCU is operated in run mode for the first time. When the MC9RS08LA8 is shipped from the Freescale factory, the flash program memory is usually 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.

For additional information about the active background mode, refer to the [Chapter 17, “Development Support”](#) chapter of this document.

### 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 program counter (PC) is halted at the position where the WAIT instruction is executed. When an interrupt request occurs:

1. MCU exits wait mode and resumes processing
2. PC is incremented by one and fetches the next instruction to be processed.

It is the responsibility of the user program to poll the corresponding interrupt source that woke the MCU, because no vector fetching process is involved.

While the MCU is in wait mode, not all background debug commands can be used. 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.

Table 3-1 summarize the behavior of the MCU in wait mode.

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

Mode	CPU	Regulator	ICS	I/O Pins	RTI	LCD	ADC	ACMP	Digital Peripherals
Wait	Standby	On	Optionally On	States held	Optionally On	Optionally On	Optionally On	Optionally On	Optionally On

### 3.6 Stop Modes

Stop mode is entered upon execution of a STOP instruction when the STOPE bit in the system option register is set. In stop mode, all internal clocks to the CPU and the modules are halted. If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU will not enter stop mode and an illegal opcode reset is forced.

Table 3-2 summarizes the behavior of the MCU in stop mode.

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

Mode	CPU	Regulator	ICS <sup>1</sup>	I/O Pins	RTI	LCD <sup>2</sup>	ADC <sup>3</sup>	ACMP <sup>4</sup>	Digital Peripherals
Stop	Standby	Optionally On	Optionally On	States held	Optionally On	Optionally On	Optionally On	Optionally On	Standby

<sup>1</sup> ICS require IREFSTEN = 1 and LVDE and LVDSE must be set to allow operation in stop

<sup>2</sup> Requires the asynchronous LCD reference, LVDE and LVDESE bits both in the SPMSC1 to be set, otherwise LCD is in standby mode

<sup>3</sup> Requires the asynchronous ADC reference, LVDE and LVDESE bits both in the SPMSC1 to be set, otherwise ADC is in standby mode

<sup>4</sup> If bandgap reference is required, the LVDE and LVDSE bits in the SPMSC1 must both be set before entering stop.

Upon entering stop mode, all of the clocks in the MCU are halted. The ICS is turned off by default when the IREFSTEN bit is cleared and the voltage regulator is put in standby. The states of all of the internal registers and logic, as well as the RAM content, are maintained. The I/O pin states are held.

Exit from stop is done by asserting  $\overline{\text{RESET}}$ , any asynchronous interrupt that has been enabled, or the real-time interrupt. The asynchronous interrupts are the KBI pins, LVD interrupt, ADC interrupt or the ACMP interrupt.

If stop is exited by asserting the  $\overline{\text{RESET}}$  pin, the MCU will be reset and program execution starts at location \$3FFD. If exited by means of an asynchronous interrupt or real-time interrupt, the MCU will increment the program counter (PC), which halted at the location where the STOP instruction was executed, and the next instruction will be fetched and executed accordingly. It is the responsibility of the user program to probe for the corresponding interrupt source that woke the CPU.

A separate self-clocked source ( $\approx 1$  kHz) for the real-time interrupt allows a wakeup from stop mode with no external components. When RTIS = 000, the real-time interrupt function is disabled. When MCU is in

STOP mode, LVD is disabled, and RTICKS = 1, the internal 1 kHz oscillator is disabled and power consumption is lower.

The external reference clock can also be enabled for the real-time interrupt to allow a wakeup from stop mode. The external clock reference is enabled by setting the EREFSTEN bit. For the ICS to run in stop mode, the LVDE and LVDSE bits in the SPMSC1 must both be set before entering stop.

To enable ADC in STOP mode the asynchronous ADC clock and LVD must be enabled by setting LVDE and LVDSE, otherwise the ADC is in standby.

### 3.6.1 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 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#). 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.

[Table 3-3](#) summarizes the behavior of the MCU in stop when entry into the active background mode is enabled.

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

Mode	CPU	Regulator	ICS	I/O Pins	RTI	LCD <sup>1</sup>	ADC <sup>2</sup>	ACMP <sup>3</sup>	Digital Peripherals
Stop	Standby	On	On	States held	Optionally On	Optionally On	Optionally On	Optionally On	Standby

<sup>1</sup> Requires the asynchronous LCD reference, LVDE and LVDESE bits both in the SPMSC1 to be set, otherwise LCD is in standby mode

<sup>2</sup> Requires the asynchronous ADC reference, LVDE and LVDESE bits both in the SPMSC1 to be set, otherwise ADC is in standby mode

<sup>3</sup> If bandgap reference is required, the LVDE and LVDSE bits in the SPMSC1 must both be set before entering stop.

### 3.6.2 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, the voltage regulator remains active.

[Table 3-4](#) summarizes the behavior of the MCU in stop mode when LVD reset is enabled.

**Table 3-4. LVD Enabled Stop Mode Behavior**

Mode	CPU	Regulator	ICS	I/O Pins	RTI	LCD <sup>1</sup>	ADC <sup>2</sup>	ACMP <sup>3</sup>	Digital Peripherals
Stop	Standby	On	Optionally On	States held	Optionally On	Optionally On	Optionally On	Optionally On	Standby

<sup>1</sup> Requires the asynchronous LCD reference, LVDE and LVDESE bits both in the SPMSC1 to be set, otherwise LCD is in standby mode

<sup>2</sup> Requires the asynchronous ADC reference, LVDE and LVDESE bits both in the SPMSC1 to be set, otherwise ADC is in standby mode

<sup>3</sup> If bandgap reference is required, the LVDE and LVDSE bits in the SPMSC1 must both be set before entering stop.



# Chapter 4

## Memory

### 4.1 MC9RS08LA8 Memory Map

Figure 4-1 shows the memory map for the MC9RS08LA8. On-chip memory in the MC9RS08LA8 consists of RAM, flash program memory for nonvolatile data storage, plus I/O and control/status registers. The registers are divided into the following groups:

- Fast access RAM using tiny and short instructions (\$0000–\$000D)
- Indirect data access D[X] (\$000E)
- Index register X for D[X] (\$000F)
- Frequently used peripheral registers (\$0010–\$001E, \$0020–\$004F)
- PAGESEL register (\$001F)
- RAM for MC9RS08LA8 (\$0050–\$00BF, \$0100–\$017F)
- Paging window (\$00C0–\$00FF)
- High-page registers (\$0200–\$027F)
- Nonvolatile memory for MC9RS08LA8 (\$2000–\$3FFF)

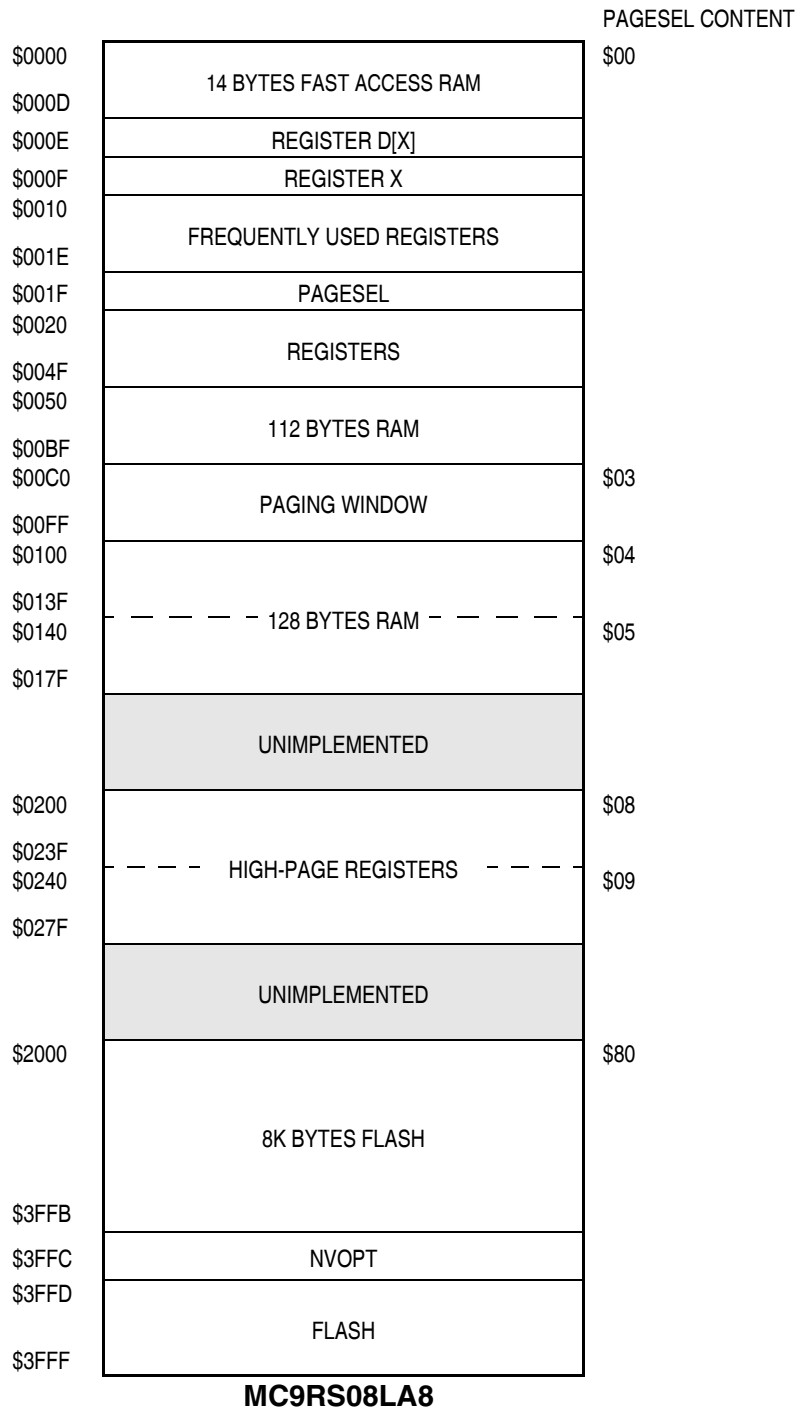


Figure 4-1. MC9RS08LA8 Memory Map

## 4.2 Unimplemented Memory

Attempting to access either data or execute an instruction fetched from unimplemented memories address will cause reset.

## 4.3 Indexed/Indirect Addressing

Register D[X] and register X together perform the indirect data access. Register D[X] is mapped to address \$000E. Register X is located in address \$000F. The 8-bit register X contains the address that is used when register D[X] is accessed. Register X is cleared to zero upon reset. By programming register X, any location on the first page (\$0000–\$00FF) can be read/written via register D[X]. Figure 4-3 shows the relationship between D[X] and register X. For example, in HC08/S08 syntax *lda,x* is comparable to *lda D[X]* in RS08 coding when register X has been programmed with the index value.

The physical location of \$000E is in RAM. Accessing the location through D[X] returns \$000E RAM content when register X contains \$0E. The physical location of \$000F is register X, itself. Reading the location through D[X] returns register X content; writing to the location modifies register X.

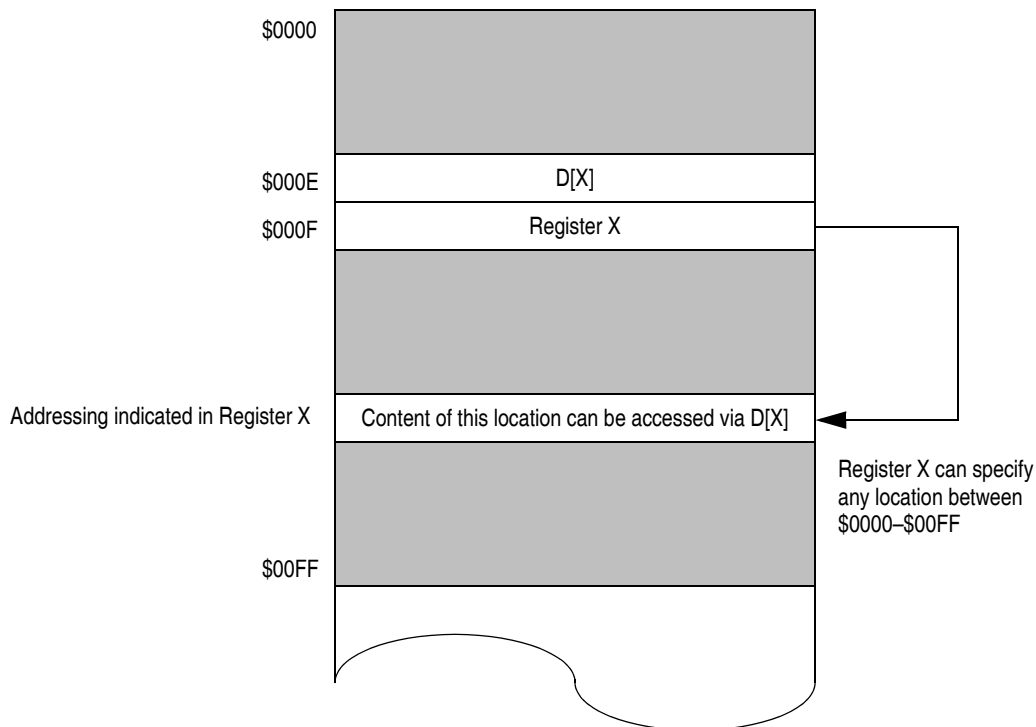


Figure 4-2. Indexed/Indirect Addressing

### 4.3.1 Accessing High-Page RAM

MC9RS08LA8 has 2-page high-page RAM from \$0100 to \$017F. This RAM exceeds maximum address direct-addressing mode can reach. To accessing this RAM, setting PAGESEL with \$04 will map the former 64-byte RAM in paging window. Setting PAGESEL with \$05 will map the latter 64-byte RAM in

paging window. Accessing the paging window by direct-addressing mode can access both RAM pages indirectly.

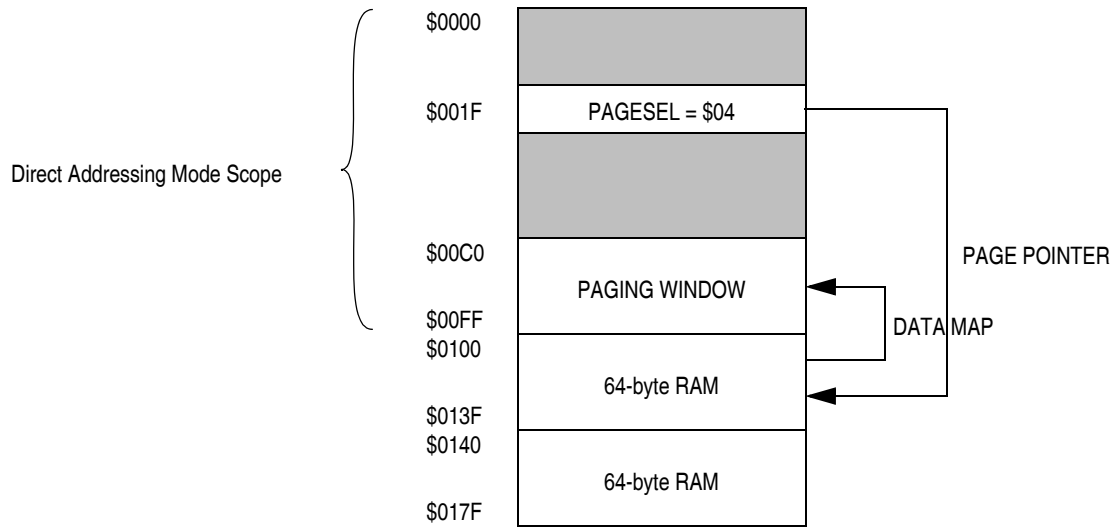


Figure 4-3. Accessing High-Page RAM

### 4.3.2 Accessing High-Page Register

MC9RS08LA8 has two high-pages to store control registers from \$0200. Setting PAGESEL with \$08 and \$09 will map both pages into paging window respectively. Accessing the paging window by direct-addressing mode can access high-page registers indirectly.



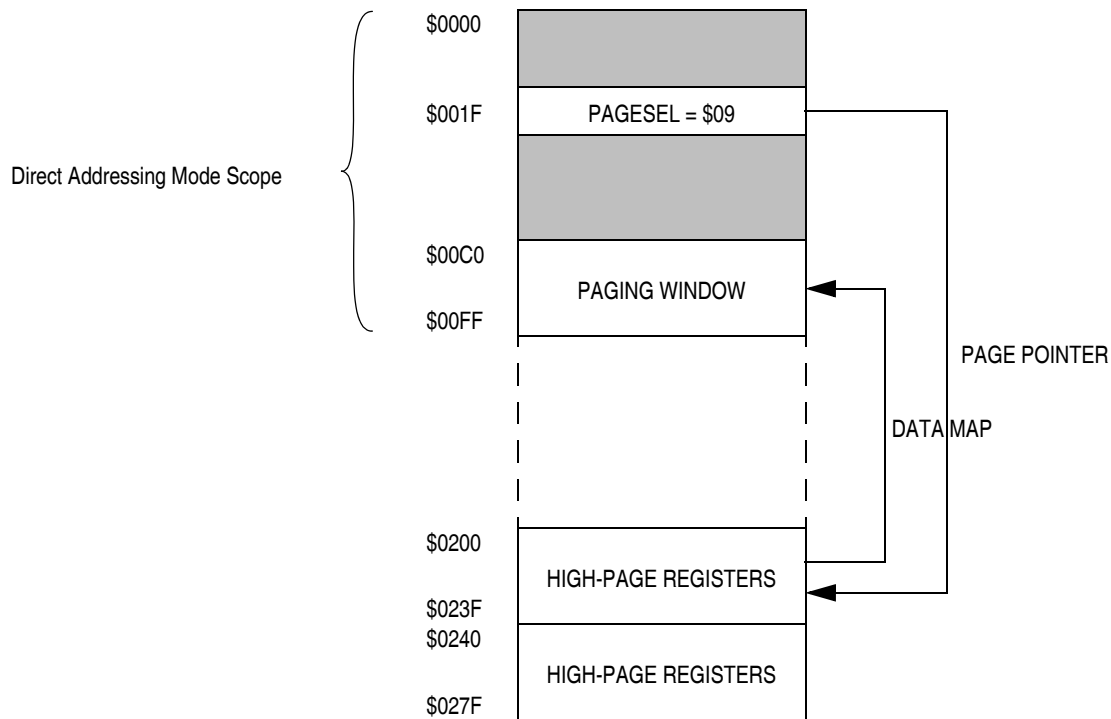


Figure 4-4. Accessing High-Page Registers

## 4.4 Register Addresses and Bit Assignments

The fast access RAM area can be accessed by using tiny, short, and direct addressing mode instructions. For tiny addressing mode instructions, the operand is encoded along with the opcode to a single byte.

Frequently used registers can make use of the short addressing mode instructions for faster load, store, and clear operations. For short addressing mode instructions, the operand is encoded along with the opcode to a single byte.

In [Table 4-1](#), the register names in column two are in bold to set them apart from the bit names to the right. Cells 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. When writing to these bits, write a 0 unless otherwise specified.

Table 4-1. Register Summary (Sheet 1 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0000		Fast Access RAM							
\$000D		Fast Access RAM							
\$000E	<b>D[X]</b>	Bit 7	6	5	4	3	2	1	Bit 0
\$000F	<b>X</b>	Bit 7	6	5	4	3	2	1	Bit 0
\$0010	<b>ADCSC1</b>	COCO	AIEN	ADCO	ADCH				
\$0011	<b>ADCSC2</b>	ADACT	ADTRG	ACFE	ACFGT	0	0	R	R
\$0012	<b>ADCRH</b>	0	0	0	0	0	0	ADR9	ADR8
\$0013	<b>ADCRL</b>	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
\$0014	<b>ADCCVH</b>	0	0	0	0	—	—	ADCV9	ADCV8
\$0015	<b>ADCCVL</b>	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
\$0016	<b>ADCCFG</b>	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
\$0017	<b>APCTL1</b>	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
\$0018	<b>SRS</b>	POR	PIN	COP	ILOP	ILAD	0	LVD	0
\$0019	<b>SOPT</b>	COPE	COPT	STOPE	0	SCICS	SCIMS	BKGDPE	RSTPE
\$001A	<b>SIP1</b>	LVD	SPI	ACMP	ADC	MTIM	KBI	LCD	RTI
\$001B	<b>SIP2</b>	0	SCIT	SCIR	SCIE	0	TPMCH1	TPMCH0	TPM
\$001C	<b>KBISC</b>	0	0	0	0	KBF	KBACK	KBIE	KBMOD
\$001D	<b>KBIPE</b>	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
\$001E	<b>KBIES</b>	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
\$001F	<b>PAGESEL</b>	Bit 7	6	5	4	3	2	1	Bit 0
\$0020	<b>LCDWF0</b>	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
\$0021	<b>LCDWF1</b>	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
\$0022	<b>LCDWF2</b>	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
\$0023	<b>LCDWF3</b>	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
\$0024	<b>LCDWF4</b>	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
\$0025	<b>LCDWF5</b>	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
\$0026	<b>LCDWF6</b>	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
\$0027	<b>LCDWF7</b>	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
\$0028	<b>LCDWF8</b>	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
\$0029	<b>LCDWF9</b>	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
\$002A	<b>LCDWF10</b>	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
\$002B	<b>LCDWF11</b>	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
\$002C	<b>LCDWF12</b>	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
\$002D	<b>LCDWF13</b>	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
\$002E	<b>LCDWF14</b>	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
\$002F	<b>LCDWF15</b>	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
\$0030	<b>LCDWF16</b>	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
\$0031	<b>LCDWF17</b>	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
\$0032	<b>LCDWF18</b>	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
\$0033	<b>LCDWF19</b>	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
\$0034	<b>LCDWF20</b>	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
\$0035	<b>LCDWF21</b>	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21

Table 4-1. Register Summary (Sheet 2 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0036	<b>LCDWF22</b>	BPHLCD22	BPGLCD22	BPFLCD22	BPELCD22	BPDLCD22	BPCLCD22	BPBLCD22	BPALCD22
\$0037	<b>LCDWF23</b>	BPHLCD23	BPGLCD23	BPFLCD23	BPELCD23	BPDLCD23	BPCLCD23	BPBLCD23	BPALCD23
\$0038	<b>LCDWF24</b>	BPHLCD24	BPGLCD24	BPFLCD24	BPELCD24	BPDLCD24	BPCLCD24	BPBLCD24	BPALCD24
\$0039	<b>LCDWF25</b>	BPHLCD25	BPGLCD25	BPFLCD25	BPELCD25	BPDLCD25	BPCLCD25	BPBLCD25	BPALCD25
\$003A	<b>LCDWF26</b>	BPHLCD26	BPGLCD26	BPFLCD26	BPELCD26	BPDLCD26	BPCLCD26	BPBLCD26	BPALCD26
\$003B	<b>LCDWF27</b>	BPHLCD27	BPGLCD27	BPFLCD27	BPELCD27	BPDLCD27	BPCLCD27	BPBLCD27	BPALCD27
\$003C	<b>LCDWF28</b>	BPHLCD28	BPGLCD28	BPFLCD28	BPELCD28	BPDLCD28	BPCLCD28	BPBLCD28	BPALCD28
\$003D	Reserved	—	—	—	—	—	—	—	—
\$003E	Reserved	—	—	—	—	—	—	—	—
\$003F	Reserved	—	—	—	—	—	—	—	—
\$0040	<b>LCDC0</b>	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
\$0041	<b>LCDC1</b>	LCDIEN	0	0	0	0	FCDEN	LCDWAI	LCDSTP
\$0042	<b>LCDSUPPLY</b>	CPSEL	0	LADJ1	LADJ0	0	0	VSUPPLY1	VSUPPLY0
\$0043	Reserved	—	—	—	—	—	—	—	—
\$0044	<b>LCDBCTL</b>	BLINK	ALT	BLANK	0	BMODE	BRATE2	BRATE1	BRATE0
\$0045	<b>LCDS</b>	LCDIF	0	0	0	0	0	0	0
\$0046	<b>PTAD</b>	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
\$0047	<b>PTADD</b>	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
\$0048	<b>PTBD</b>	0	0	0	0	0	PTBD2	PTBD1	PTBD0
\$0049	<b>PTBDD</b>	0	0	0	0	0	0	PTBDD1	PTBDD0
\$004A	<b>PTCD</b>	PTCD7	PTCD6	0	0	PTCD3	PTCD2	PTCD1	PTCD0
\$004B	<b>PTCDD</b>	PTCDD7	0	0	0	PTCDD3	PTCDD2	PTCDD1	PTCDD0
\$004C	<b>PTDD</b>	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
\$004D	<b>PTDDD</b>	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
\$004E	<b>PTED</b>	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
\$004F	<b>PTEDD</b>	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
\$0050– \$00BF		RAM							
\$00C0– \$00FF		Paging Window							
\$0100– \$017F		RAM							
\$0180– \$01FF	Unimplemented	—	—	—	—	—	—	—	—
\$0200	<b>PTAPE</b>	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
\$0201	<b>PTAPUD</b>	PTAPUD7	PTAPUD6	PTAPUD5	PTAPUD4	PTAPUD3	PTAPUD2	PTAPUD1	PTAPUD0
\$0202	<b>PTADS</b>	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
\$0203	<b>PTASE</b>	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
\$0204	<b>PTBPE</b>	0	0	0	0	0	PTBPE2	PTBPE1	PTBPE0
\$0205	<b>PTBPUD</b>	0	0	0	0	0	PTBPUD2	PTBPUD1	PTBPUD0
\$0206	<b>PTBDS</b>	0	0	0	0	0	0	PTBDS1	PTBDS0
\$0207	<b>PTBSE</b>	0	0	0	0	0	0	PTBSE1	PTBSE0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0208	<b>PTCPE</b>	PTCPE7	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
\$0209	<b>PTCPUD</b>	PTCPUD7	0	0	0	PTCPUD3	PTCPUD2	PTCPUD1	PTCPUD0
\$020A	<b>PTCDS</b>	PTCDS7	PTCDS6	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
\$020B	<b>PTCSE</b>	PTCSE7	PTCSE6	0	0	PTCSE3	PTCSE2	PTCSE1	PTCSE0
\$020C	<b>PTDPE</b>	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
\$020D	<b>PTDPUD</b>	PTDPUD7	PTDPUD6	PTDPUD5	PTDPUD4	PTDPUD3	PTDPUD2	PTDPUD1	PTDPUD0
\$020E	<b>PTDDS</b>	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
\$020F	<b>PTDSE</b>	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
\$0210	<b>PTEPE</b>	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
\$0211	<b>PTEPUD</b>	PTEPUD7	PTEPUD6	PTEPUD5	PTEPUD4	PTEPUD3	PTEPUD2	PTEPUD1	PTEPUD0
\$0212	<b>PTEDS</b>	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
\$0213	<b>PTESE</b>	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
\$0214	Reserved	—	—	—	—	—	—	—	—
\$0215	Reserved	—	—	—	—	—	—	—	—
\$0216	<b>SDIDH</b>	—	—	—	—	ID11	ID10	ID9	ID8
\$0217	<b>SDIDL</b>	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
\$0218	<b>SPIC1</b>	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
\$0219	<b>SPIC2</b>	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
\$021A	<b>SPIBR</b>	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
\$021B	<b>SPIS</b>	SPRF	0	SPTEF	MODF	0	0	0	0
\$021C	Reserved	—	—	—	—	—	—	—	—
\$021D	<b>SPID</b>	Bit 7	6	5	4	3	2	1	Bit 0
\$021E	Reserved	—	—	—	—	—	—	—	—
\$021F	Reserved	—	—	—	—	—	—	—	—
\$0220	<b>TPMSC</b>	TOF	TOIE	CPWMS	CLKS		PS		
\$0221	<b>TPMCNTH</b>	Bit 15	14	13	12	11	10	9	8
\$0222	<b>TPMCNTL</b>	7	6	5	4	3	2	1	Bit 0
\$0223	<b>TPMMODH</b>	Bit 15	14	13	12	11	10	9	8
\$0224	<b>TPMMODL</b>	7	6	5	4	3	2	1	Bit 0
\$0225	<b>TPMC0SC</b>	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
\$0226	<b>TPMC0VH</b>	Bit 15	14	13	12	11	10	9	8
\$0227	<b>TPMC0VL</b>	7	6	5	4	3	2	1	Bit 0
\$0228	<b>TPMC1SC</b>	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
\$0229	<b>TPMC1VH</b>	Bit 15	14	13	12	11	10	9	8
\$022A	<b>TPMC1VL</b>	7	6	5	4	3	2	1	Bit 0
\$022B	<b>ACMPSC</b>	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD	
\$022C	<b>ICSC1</b>	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
\$022D	<b>ICSC2</b>	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
\$022E	<b>ICSTRM</b>	TRIM							
\$022F	<b>ICSSC</b>	0	0	0	0	CLKST		OSCINIT	FTRIM
\$0230	<b>SCIBDH</b>	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
\$0231	<b>SCIBDL</b>	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0

Table 4-1. Register Summary (Sheet 4 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0232	<b>SCIC1</b>	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
\$0233	<b>SCIC2</b>	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
\$0234	<b>SCIS1</b>	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
\$0235	<b>SCIS2</b>	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
\$0236	<b>SCIC3</b>	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
\$0237	<b>SCID</b>	Bit 7	6	5	4	3	2	1	Bit 0
\$0238	<b>SRTISC</b>	RTIF	RTIACK	RTICLKs	RTIE	0	RTIS		
\$0239	<b>SPMSC1</b>	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	—	BGBE
\$023A	Unimplemented	—	—	—	—	—	—	—	—
\$023B		—	—	—	—	—	—	—	—
\$023C	<b>MTIMSC</b>	TOF	TOIE	TRST	TSTP	0	0	0	0
\$023D	<b>MTIMCLK</b>	0	0	CLKS			PS		
\$023E	<b>MTIMCNT</b>	COUNT							
\$023F	<b>MTIMMOD</b>	MOD							
\$0240	<b>LCDPEN0</b>	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
\$0241	<b>LCDPEN1</b>	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
\$0242	<b>LCDPEN2</b>	PEN23	PEN22	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
\$0243	<b>LCDPEN3</b>	0	0	0	PEN28	PEN27	PEN26	PEN25	PEN24
\$0244	Reserved	—	—	—	—	—	—	—	—
\$0245	Reserved	—	—	—	—	—	—	—	—
\$0246	Reserved	—	—	—	—	—	—	—	—
\$0247	Reserved	—	—	—	—	—	—	—	—
\$0248	<b>LCDBPEN0</b>	BPEN7	BPEN6	BPEN5	BPEN4	BPEN3	BPEN2	BPEN1	BPEN0
\$0249	<b>LCDBPEN1</b>	BPEN15	BPEN14	BPEN13	BPEN12	BPEN11	BPEN10	BPEN9	BPEN8
\$024A	<b>LCDBPEN2</b>	BPEN23	BPEN22	BPEN21	BPEN20	BPEN19	BPEN18	BPEN17	BPEN16
\$024B	<b>LCDBPEN3</b>	0	0	0	BPEN28	BPEN27	BPEN26	BPEN25	BPEN24
\$024C	Reserved	—	—	—	—	—	—	—	—
\$024D	Reserved	—	—	—	—	—	—	—	—
\$024E	Reserved	—	—	—	—	—	—	—	—
\$024F	Reserved	—	—	—	—	—	—	—	—
\$0250	<b>FOPT</b>	0	0	0	0	0	0	0	SECD
\$0251	<b>FLCR</b>	0	0	0	0	HVEN	MASS	0	PGM
\$0252	Reserved	—	—	—	—	—	—	—	—
\$0253	Reserved	—	—	—	—	—	—	—	—
\$0254	Unimplemented	—	—	—	—	—	—	—	—
\$027F		—	—	—	—	—	—	—	—
\$3FFA <sup>1</sup>	Reserved	Reserved for Room Temperature ICS Trim							
\$3FFB <sup>1</sup>	Reserved	—	—	—	—	—	—	—	FTRIM
\$3FFC	<b>NVOPT</b>	0	0	0	0	0	0	0	SECD

— =Unimplemented or Reserved

<sup>1</sup> If using the MCU untrimmed, \$3FFA and \$3FFB may be used by applications.

## 4.5 RAM (System RAM)

The MC9RS08LA8 includes three sections of static RAM. The locations from \$0000 to \$000D can be directly accessed using the more efficient tiny addressing mode instructions and short addressing mode instructions. Location \$000E can either be accessed through D[X] register when register X is \$0E or through the paging window location \$00CE when PAGESEL register is \$00. The second section of RAM starts from \$0050 to \$00BF, and it can be accessed using direct addressing mode instructions. The third section of RAM starts from \$0100 to \$017F. By setting PAGESEL with \$04, the content of RAM starting from \$0100 to \$013F can be accessed by direct addressing mode instructions in paging window. And by setting PAGESEL with \$05, the content of RAM starting from \$0140 to \$017F can be accessed by direct addressing mode in paging window.

The RAM retains data when the MCU is in low-power wait and stop mode. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

## 4.6 Flash

The flash memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because the device does not include on-chip charge pump circuitry, external  $V_{PP}$  is required for program and erase operations.

## 4.6.1 Features

Features of the flash memory include:

- Flash size
  - MC9RS08LA8 — 8, 192 bytes
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Security feature for flash

## 4.6.2 Flash Programming Procedure

Flash memory is programmed on a row basis. A row consists of 64 consecutive bytes starting from addresses \$XX00, \$XX40, \$XX80, or \$XXC0. Use the following procedure to program a row of flash memory.

1. Apply external  $V_{PP}$ .
2. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
3. Write data to flash location, via the high-page accessing window \$00C0–\$00FF, within the address range of the row to be programmed. (Prior to the data writing operation, the PAGE register must be configured correctly to map the high-page accessing window to the corresponding flash row).
4. Wait for  $t_{nvs}$  (5  $\mu$ s at minimum).
5. Set the HVEN bit.
6. Wait for  $t_{pgs}$  (10  $\mu$ s at minimum).
7. Write data to the flash location to be programmed.
8. Wait for  $t_{prog}$  (20  $\mu$ s to 40  $\mu$ s)
9. Repeat steps 6 and 7 until all bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for  $t_{nvh}$  (5  $\mu$ s at minimum)
12. Clear the HVEN bit.
13. After  $t_{rev}$  (1  $\mu$ s), the memory can be accessed in read mode again.
14. Remove external  $V_{PP}$ .

This program sequence is repeated throughout the memory until all data is programmed.

### NOTE

Flash memory cannot be programmed or erased by software code executed from flash locations. To program or erase flash, commands must be executed from RAM or BDC commands. User code must not enter wait or stop during erase or program.

These operations must be performed in the order as shown, unrelated operations can be taken between the steps.

### 4.6.3 Flash Mass Erase Operation

Use the following procedure to mass erase the entire flash memory.

1. Apply external  $V_{PP}$ .
2. Set the MASS bit in the flash control register.
3. Write data to flash location, via the high-page accessing window \$00C0–\$00FF. (Prior to the data writing operation, the PAGE register must be configured correctly to map the high-page accessing window to the flash location).
4. Wait for  $t_{nvs}$ .
5. Set the HVEN bit.
6. Wait for  $t_{merase}$ .
7. Clear the MASS bit.
8. Wait for  $t_{nvhl}$ .
9. Clear the HVEN bit.
10. After  $t_{rcv}$ , the memory can be accessed in read mode again.
11. Remove external  $V_{PP}$ .

#### NOTE

Flash memory cannot be programmed or erased by software code executed from flash location. To program or erase flash, commands must be executed from RAM or BDC commands. User code must not enter wait or stop during erase or program.

These operations must be performed in the order as shown, unrelated operations can be taken between the steps.

### 4.6.4 Flash Security

The MC9RS08LA8 includes circuitry to help prevent unauthorized access to the contents of flash memory. When security is engaged, flash is considered a secure resource. The RAM, direct-page registers, and background debug controller are considered unsecured resources. Attempts to access a secure memory location are blocked if through the background debug interface, or when BKGDPPE is set, reads return all 0s).

Security is engaged or disengaged based on the state of a nonvolatile register bit (SECD) 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 (SECD = 0), next time the device is reset via POR, internal reset, or external reset, security is engaged. In order to disengage security, mass erase must be performed via BDM commands and followed by pin reset or POR.

The separate background debug controller can still be used for registers and RAM access. Flash mass erase is possible by writing to the flash control register that follows the flash mass erase procedure listed in [Section 4.6.3, “Flash Mass Erase Operation”](#) via BDM commands.



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

1. Mass erase flash via background BDM commands or RAM loaded program.
2. Perform reset and the device will boot up with security disengaged.

#### NOTE

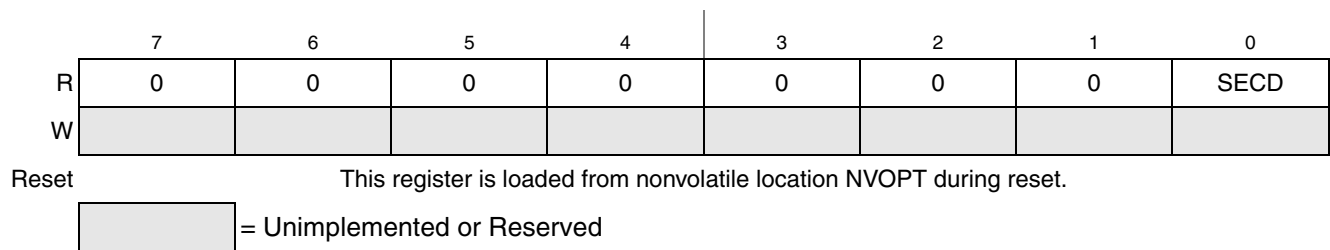
When the device boots up to normal operating mode, MS pin is high during reset, flash security is engaged if SECD programmed (SECD = 0). BKGDPE is reset to 0; all BDM communication is blocked, and background debug is not allowed.

### 4.6.5 Flash Registers and Control Bits

The flash module has a nonvolatile register, NVOPT (\$3FFC), in flash memory which is copied into the corresponding control register, FOPT (\$0250), at reset.

### 4.6.6 Flash Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from flash into FOPT. Bits 7 through 1 are not used and always read 0. This register may be read at any time, but writes have no meaning or effect.

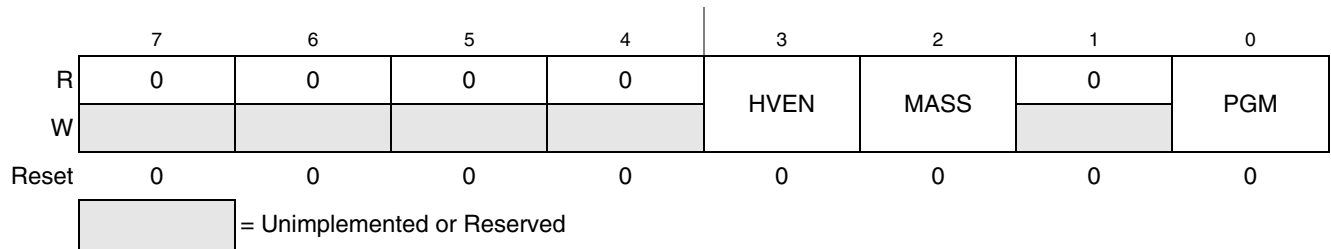


**Figure 4-5. Flash Options Register (FOPT)**

**Table 4-2. FOPT Register Field Descriptions**

Field	Description
0 SECD	<b>Security State Code</b> — This bit field determines the security state of the MCU. When the MCU is secured, the contents flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to <a href="#">Section 4.6.4, “Flash Security”</a> 0 Security engaged 1 Security disengaged

## 4.6.7 Flash Control Register (FLCR)



**Figure 4-6. Flash Control Register (FLCR)**

**Table 4-3. FLCR Register Field Descriptions**

Field	Description
3 HEVEN	<b>High Voltage Enable</b> — This read/write bit enables high voltages to the flash array for program and erase operation. HVEN can be set only if either PGM = 1 or MASS = 1 and the proper sequence for program or erase is followed. 0 High voltage disabled to array. 1 High voltage enabled to array.
2 MASS	<b>Mass Erase Control Bit</b> — This read/write bit configures the memory for mass erase operation. 0 Mass erase operation not selected. 1 Mass erase operation selected.
0 PGM <sup>1</sup>	<b>Program Control Bit</b> — This read/write bit configures the memory for program operation. PGM is interlocked with the MASS bit such that both bits cannot be equal to 1 or set to 1 at the same time. 0 Program operation not selected. 1 Program operation selected.

<sup>1</sup> When flash security is engaged, writing to PGM bit has no effect. As a result, flash programming is not allowed.

## 4.6.8 Page Select Register (PAGESEL)

There is a 64-byte window (\$00C0–\$00FF) in the direct-page reserved for paging access. Programming the page select register determines the corresponding 64-byte block on the memory map for direct-page access. For example, when the PAGESEL register is programmed with value \$08, the high-page register (\$0200–\$023F) can be accessed through the paging window (\$00C0–\$00FF) via direct addressing mode instructions.

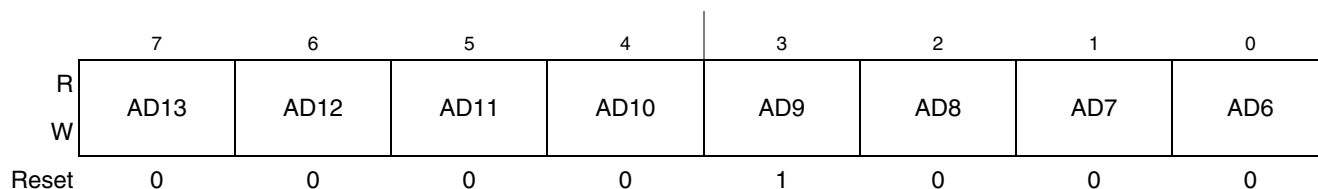


Figure 4-7. Page Select Register (PAGESEL)

Table 4-4. PAGESEL Field Descriptions

Field	Description
7:0 AD[13:6]	<b>Page Selector</b> — These bits define the address line bit 6 to bit 13, which determines the 64-byte block boundary of the memory block accessed via the direct paging window.

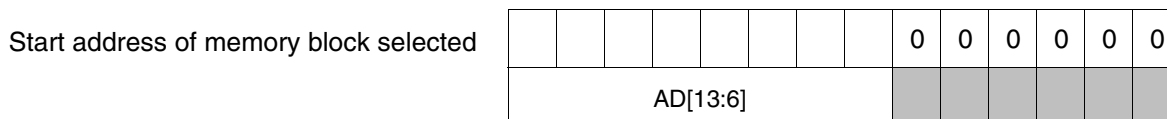


Figure 4-8. Memory Block Boundary Selector

Table 4-5 shows the memory block to be accessed through paging window (\$00C0–\$00FF).

Table 4-5. Paging Window

Page	Memory	Contents
\$00	\$0000–\$003F	RAM, D[X], X, PAGESEL, and registers
\$01	\$0040–\$007F	RAM and registers
\$02	\$0080–\$00BF	RAM
\$03	\$00C0–\$00FF	Paging window itself
\$04	\$0100–\$013F	RAM
\$05	\$0140–\$017F	RAM
\$06–\$07	\$0180–\$01FF	Not Available
\$08–\$09	\$0200–\$027F	High-page registers

Table 4-5. Paging Window (continued)

Page	Memory	Contents
\$0A-\$7F	\$0280-\$1FFF	Not Available
\$80-\$FF	\$2000-\$3FFF	Flash

# Chapter 5

## Resets, Interrupts, and System Configuration

### 5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9RS08LA8. Some interrupt sources from peripheral modules are discussed in great detail in other chapters of this 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, are not part of on-chip peripheral systems with their own sections but are part of the system control logic.

### 5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- System reset status register (SRS) to indicate source of most recent reset
- System interrupt pending registers (SIP1 and SIP2) to indicate status of pending system interrupts
  - ACMP interrupt with enable
  - Keyboard interrupt with enable
  - ADC interrupt with enable
  - SCI interrupt with enable
  - SPI interrupt with enable
  - TPM interrupt with enable
  - MTIM interrupt with enable
  - LCD interrupt with enable
  - LVD interrupt with enable
  - RTI interrupt with enable

### 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 started from location \$3FFD. A JMP instruction (opcode \$BC) with operand located at \$3FFE–\$3FFF must be programmed into the user application for correct reset operation. The operand defines the location at which the user program will start. On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup/pulldown devices disabled.

The MC9RS08LA8 has seven sources for reset:

- External pin reset (PIN) — enabled using RSTPE in SOPT
- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Background debug forced reset via BDC command BDC\_RESET

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

## 5.4 Computer Operating Properly (COP) Watchdog

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

After any reset, the COPE becomes set in SOPT, which enables the COP watchdog (see [Section 5.8.2, “System Options Register \(SOPT\)”](#), for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

There is an associated short and long timeout controlled by COPT in SOPT. [Table 5-1](#) summarizes the control functions of the COPT bit. The COP watchdog operates from the 1 kHz clock source and defaults to the associated long time-out ( $2^8$  cycles).

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

COPT	1 kHz COP Overflow Count <sup>1</sup>
0	$2^5$ cycles (32 ms)
1	$2^8$ cycles (256 ms)

<sup>1</sup> Values shown in this column are based on  $t_{RT1} \approx 1$  ms.

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

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

When the MCU enters stop mode, the COP counter is re-initialized to zero upon entry to stop mode. The COP counter begins from zero as soon as the MCU exits stop mode.

## 5.5 Interrupts

The MC9RS08LA8 does not include an interrupt controller with vector table lookup mechanism as used on the HC08 and HCS08 devices. However, the interrupt sources from the modules such as LVD, KBI, and ACMP are still available to wake the CPU from wait or stop mode. It is the responsibility of the user application to poll the corresponding module to determine the source of wakeup.

Each wakeup source of the module is associated with a corresponding interrupt enable bit. If the bit is disabled, the interrupt source is gated, and that particular source cannot wake the CPU from wait or stop mode. However, the corresponding interrupt flag will still be set to indicate that external wakeup event has occurred.

The system interrupt pending registers (SIP1 and SIP2) indicate the status of the system pending interrupt. When the read-only bit of SIPs is enabled, it shows there is a pending interrupt to be serviced from the indicated module. Writing to the register bit has no effect. The pending interrupt flag will be cleared automatically when all the corresponding interrupt flags from the indicated module are cleared.

## 5.6 Low-Voltage Detect (LVD) System

The MC9RS08LA8 includes a system to protect against low voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and an LVD circuit with a predefined trip voltage. The LVD circuit is enabled with LVDE in SPMSC1. The LVD is disabled upon entering stop mode unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, the current consumption in stop 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. As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above the  $V_{LVD}$  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  $V_{LVD}$ . The LVD bit in the SRS register is set following 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), LVDF in SPMSC1 will be set and an LVD interrupt request will occur.

## 5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts. The RTI is driven by either the 1 kHz internal clock reference or the external clock reference (ICSERCLK) from the ICS module. The external clock reference is divided by 32 by the RTI logic to produce a low frequency clock for applications requiring more accurate real-time interrupts. The RTICLK bit in SRTISC is used to select the RTI clock source. Both the 1 kHz and the external clock sources for the RTI can be used when the MCU is in run, wait or stop mode. When using the external oscillator in normal or wait mode, setting ERCLKEN = 1. When using the external oscillator in stop mode, set ERCLKEN = 1 and EREFSTEN = 1.

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 or disable RTI. 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 zeros, and no interrupts will be generated.

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

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

Some control bits in the SOPT register 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 System Reset Status Register (SRS)

This register includes six read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by the BDC\_RESET command, 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
LVR:	U	0	0	0	0	0	1	0
Any other reset:	0	(1)	(1)	(1)	(1)	0	0	0

U = Unaffected by reset

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

**Figure 5-1. System Reset Status (SRS)**



Table 5-2. SRS Register Field Descriptions

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

## 5.8.2 System Options Register (SOPT)

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

	7	6	5	4	3	2	1	0
R	COPE	COPT	STOPE	0	SCICS	SCIMS	BKGDPE	RSTPE
W								
Reset	1	1	0	0	0	0	1 <sup>1</sup>	u
POR	1	1	0	0	0	0	1	0

= Unimplemented or Reserved
 u = Unaffected

<sup>1</sup> When the device is reset into normal operating mode (MS is high during reset), BKGDPE is reset to 1 if flash security is disengaged (SECD = 1); BKGDPE is reset to 0 if flash security is engaged (SECD = 0). When the device is reset into active BDM mode (MS is low during reset), BKGDPE is always reset to 1 such that BDM communication is allowed.

Figure 5-2. System Options Register (SOPT)

Table 5-3. SOPT Register Field Descriptions

Field	Description
7 COPE	<b>COP Watchdog Enable</b> — 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	<b>COP Watchdog Timeout</b> — This write-once bit selects the timeout period of the COP. 0 Short timeout period ( $2^5$ cycles) selected. 1 Long timeout period ( $2^8$ cycles) selected.
5 STOPE	<b>Stop Mode Enable</b> — 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.
3 SCICS	<b>SCI Channel Select</b> — In SCI fixed mode, this bit indicates which SCI channel is used. When clear, TxD and RxD on PTC1 and PTC0 are used. When set, TxD and RxD on PTA3 and PTA2 are used. In SCI mixed mode, writing this bit indicates the TxD channel is used. Reading this bit indicates the RxD channel data is received. Writing 0 to this bit takes PTC1 as the TxD channel. Writing 1 to this bit takes PTA3 as the TxD channel. Reading 0 from this bit indicates the data from PTC0 is received. Reading 1 from this bit indicates the data from PTA2 is received. For more detailed information, Please see SCI chapter. 0 In SCI fixed mode, PTC1 is used as TxD and PTC0 is used as RxD. In SCI mixed mode, reading this value indicates the data from PTC0 received and writing this value takes PTC1 as TxD channel. 1 In SCI fixed mode, PTA3 is used as TxD and PTA2 is used as RxD. In SCI mixed mode, reading this value indicates the data from PTA2 received and writing this value takes PTA3 as TxD channel.
2 SCIMS	<b>SCI Mode Select</b> — When set, this bit enables SCI works at mixed mode. When clear, this bit enable SCI works at clear mode. For more detailed information, Please see SCI chapter. 0 Fixed mode selected. 1 Mixed mode selected.
1 BKGDPPE <sup>1</sup>	<b>Background Debug Mode Pin Enable</b> — This write-once bit when set enables the PTC6/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 BKGDMS function following any MCU reset. 0 PTC6/ACMPO/BKGD/MS pin functions as PTC6 or ACMPO 1 PTC6/ACMPO/BKGD/MS pin functions as BKGD/MS
0 RSTPE	<b>RESET Pin Enable</b> — When set, this write-once bit enables the PTB2/RESET/V <sub>PP</sub> pin to function as RESET. When clear, the pin functions as one of its input-only alternative functions. This pin is input-only port function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on RESET. 0 PTB2/RESET/V <sub>PP</sub> pin function as PTB2/V <sub>PP</sub> 1 PTB2/RESET/V <sub>PP</sub> pin function as RESET/V <sub>PP</sub>

<sup>1</sup> BKGDPPE can write only once from value 1 to 0. Writing from value 0 to 1 by user software is not allowed. BKGDPPE can be changed back to 1 only by a POR or reset with proper condition.

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

This read-only registers are included so host development systems can identify the RS08 derivative and revision number. This allows the development software to locate specific memory blocks, registers, and control bits in a target MCU.

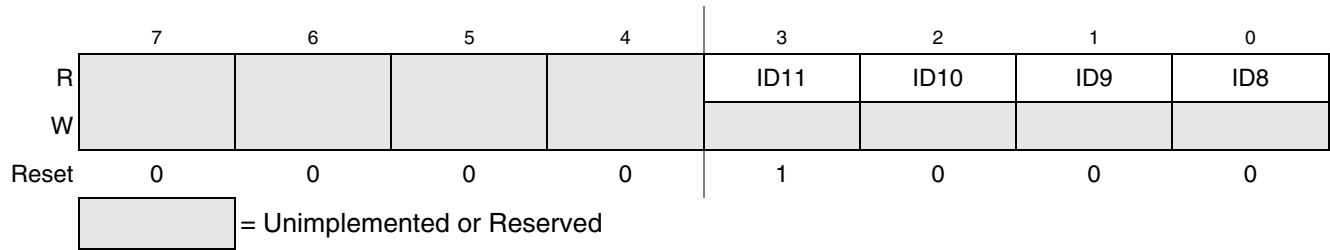


Figure 5-3. System Device Identification Register — High (SDIDH)

Table 5-4. SDIDH Register Field Descriptions

Field	Description
3:0 ID[11:8]	<b>Part Identification Number</b> — Each derivative in the HCS08 Family has a unique identification number. The MC9RS08LA8 is hard coded to the value 0x0804. See also ID bits in <a href="#">Figure 5-4., “System Device Identification Register — Low (SDIDL)”</a> .

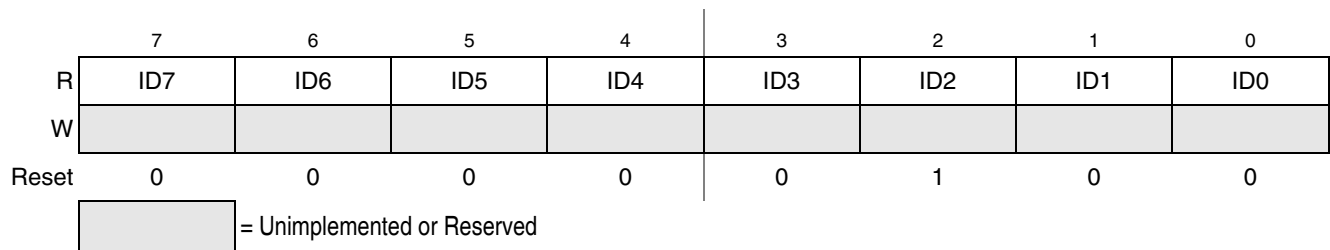


Figure 5-4. System Device Identification Register — Low (SDIDL)

Table 5-5. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	<b>Part Identification Number</b> — Each derivative in the RS08 Family has a unique identification number. The MC9RS08LA8 is hard coded to the value 0x0804. See also ID bits in <a href="#">Figure 5-3., “System Device Identification Register — High (SDIDH)”</a> .

### 5.8.4 System PMC Real-Time Interrupt Status and Control (SRTISC)

SRTISC contains the status and control bits associated with the PMC real-time interrupt function.

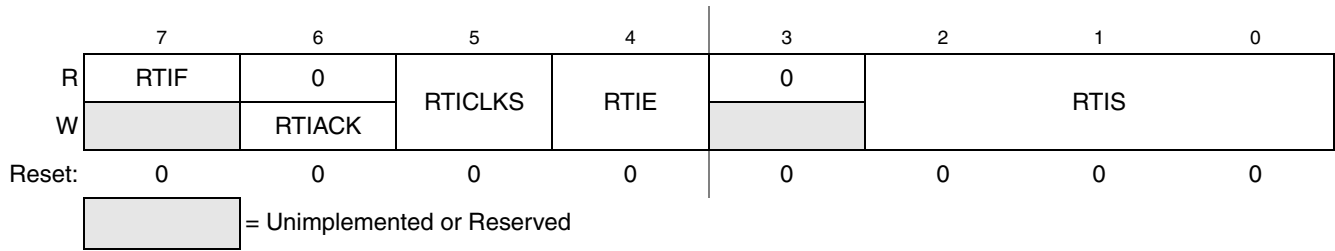


Figure 5-5. PMC Real-Time Interrupt Status and Control (SRTISC)

Table 5-6. SRTISC Register Field Descriptions

Field	Description
7 RTIF	<b>Real-Time Flag</b> — 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	<b>Real-Time Interrupt Acknowledge</b> — 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.
5 RTICLKs	<b>Real-Time Interrupt Clock Select</b> — This read/write bit selects the clock source for the real-time interrupt. 0 Real-time interrupt request clock source is internal 1 kHz oscillator. 1 Real-time interrupt request clock source is external clock.
4 RTIE	<b>Real-Time Interrupt Enable</b> — This read-write bit enables real-time interrupts. 0 Real-time interrupts disabled. 1 Real-time interrupts enabled.
2:0 RTIS[2:0]	<b>Real-Time Interrupt Select</b> — These read/write bits select the period for the RTI.

Table 5-7. Real-Time Interrupt Period

RTIS	1 kHz RTI Timeout <sup>1</sup>	T <sub>extclk</sub> RTI Timeout <sup>2</sup>
000	Disable RTI	Disable RTI
001	8 ms	256 × T <sub>extclk</sub>
010	32 ms	1,024 × T <sub>extclk</sub>
011	64 ms	2,048 × T <sub>extclk</sub>
100	128 ms	4,096 × T <sub>extclk</sub>
101	256 ms	8,192 × T <sub>extclk</sub>
110	512 ms	16,284 × T <sub>extclk</sub>
111	1.024 s	32,768 × T <sub>extclk</sub>

<sup>1</sup> Timeout values shown based on RTI clock source of 1ms period. Consult electricals for tolerances of internal 1 kHz source, t<sub>RTI</sub>.

<sup>2</sup> T<sub>extclk</sub> is the period of external clock source connected to RTI.

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

This register contains status and control bits to support the low-voltage detect function, and to enable the bandgap voltage reference for use by the ACMP and the ADC module.

	7	6	5	4	3	2	1	0
R	LVDF <sup>1</sup>	0	LVDIE	LVDRE <sup>2</sup>	LVDSE	LVDE <sup>2</sup>	0	BGBE
W		LVDACK						
Reset:	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

<sup>1</sup> LVDF will be set in the case when  $V_{Supply}$  transitions below the trip point or after reset and  $V_{Supply}$  is already below  $V_{LVD}$ .

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

**Figure 5-6. System Power Management Status and Control 1 Register (SPMSC1)**

**Table 5-8. SPMSC1 Register Field Descriptions**

Field	Description
7 LVDF	<b>Low-Voltage Detect Flag</b> — The LVDF bit indicates the low-voltage detect status if LVDE is set. 0 low-voltage is being or has been detected. 1 low-voltage has not been detected.
6 LVDACK	<b>Low-Voltage Detect Acknowledge</b> — Write a 1 to LVDACK clears the LVD interrupt request and clears LVDF if a low voltage is not detected.
5 LVDIE	<b>Low-Voltage Detect Interrupt Enable</b> — The LVDIE bit controls the LVD interrupt if LVDE is set. This bit has no effect if LVDE is 0. 0 LVD interrupt disabled. 1 LVD interrupt enabled.
4 LVDRE	<b>Low-Voltage Detect Reset Enable</b> — The LVDRE bit controls the LVD reset if LVDE is set. This bit has no effect if the LVDE is 0. LVD reset has priority over LVD interrupt, if both are enabled. In all test modes, the LVDRE bit value is ignored and functions as if the LVDRE is 0. 0 LVD reset disabled. 1 LVD reset enabled.
3 LVDSE	<b>Low-Voltage Detect Stop Enable</b> — The LVDSE bit controls the behavior of the LVD when the MCU stop mode is entered if LVDE is set. This bit has no effect if the LVDE is 0. 0 LVD disabled in MCU stop mode. 1 LVD enabled in MCU stop mode.
2 LVDE	<b>Low-Voltage Detect Enable</b> — The LVDE bit controls whether the LVD is enabled. 0 LVD is disabled. 1 LVD is enabled.
0 BGBE	<b>Bandgap Buffer Enable</b> — The BGBE bit is used to enable the bandgap buffered output. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

## 5.8.6 System Interrupt Pending Register 1 (SIP1)

This register contains status of the pending interrupt from the modules.

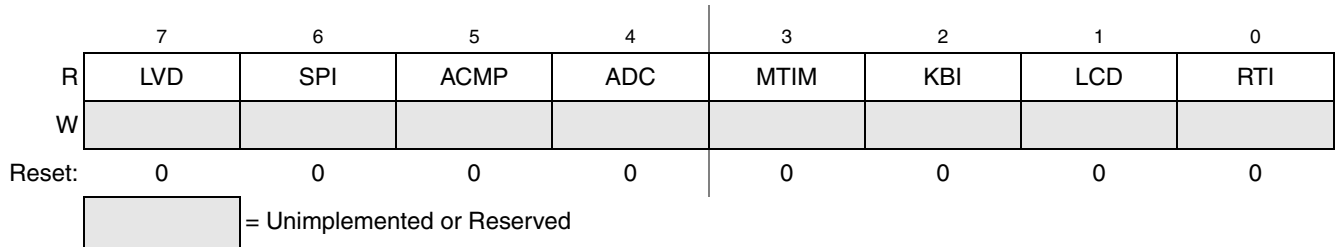


Figure 5-7. System Interrupt Pending Register 1 (SIP1)

Table 5-9. SIP1 Register Field Descriptions

Field	Description
7 LVD	<b>LVD Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from LVD module. Clearing the corresponding flag in LVD module clears this bit. Reset also clears this bit. 0 There is no pending LVD interrupt. 1 There is a pending LVD interrupt.
6 SPI	<b>SPI Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from SPI module. Clearing the corresponding flag in SPI module clears this bit. Reset also clears this bit. 0 There is no pending SPI interrupt. 1 There is a pending SPI interrupt.
5 ACMP	<b>ACMP Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from ACMP module. Clearing the corresponding flag in ACMP module clears this bit. Reset also clears this bit. 0 There is no pending ACMP interrupt. 1 There is a pending ACMP interrupt.
4 ADC	<b>ADC Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from ADC module. Clearing the corresponding flag in ADC module clears this bit. Reset also clears this bit. 0 There is no pending ADC interrupt. 1 There is a pending ADC interrupt.
3 MTIM	<b>MTIM Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from MTIM module. Clearing the corresponding flag in MTIM module clears this bit. Reset also clears this bit. 0 There is no pending MTIM interrupt. 1 There is a pending MTIM interrupt.
2 KBI	<b>KBI Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from KBI module. Clearing the corresponding flag in KBI module clears this bit. Reset also clears this bit. 0 There is no pending KBI interrupt. 1 There is a pending KBI interrupt.
1 LCD	<b>LCD Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from LCD module. Clearing the corresponding flag in LCD module clears this bit. Reset also clears this bit. 0 There is no pending LCD interrupt. 1 There is a pending LCD interrupt.
0 RTI	<b>RTI Interrupt Pending</b> — This read-only bit indicates there is a pending interrupt from RTI module. Clearing the corresponding flag in RTI module clears this bit. Reset also clears this bit. 0 There is no pending RTI interrupt. 1 There is a pending RTI interrupt.

### 5.8.7 System Interrupt Pending Register 2 (SIP2)

This register contains status of the pending interrupt from the modules.

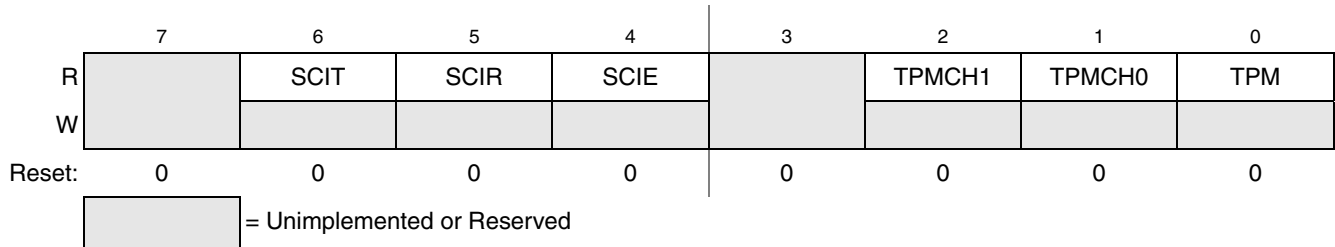


Figure 5-8. System Interrupt Pending Register 2 (SIP2)

Table 5-10. SIP2 Register Field Descriptions

Field	Description
6 SCIT	<b>SCI Transmit Interrupt Pending</b> — This read-only bit indicates there is a pending transmit interrupt from SCI module. This flag can be triggered by TDRE and TC. Clearing the corresponding flag in SCI module clears this bit. Reset also clears this bit. 0 There is no pending SCI transmit interrupt. 1 There is a pending SCI transmit interrupt.
5 SCIR	<b>SCI Receive Interrupt Pending</b> — This read-only bit indicates there is a pending receive interrupt from SCI module. This flag can be triggered by IDLE, RDRF, LBKDIF, or RXEDGIF. Clearing the corresponding flag in SCI module clears this bit. Reset also clears this bit. 0 There is no receive pending SCI interrupt. 1 There is a receive pending SCI interrupt.
4 SCIE	<b>SCI Error Interrupt Pending</b> — This read-only bit indicates there is a pending error interrupt from SCI module. This flag can be triggered by OR, NF, FE, PF. Clearing the corresponding flag in SCI module clears this bit. Reset also clears this bit. 0 There is no pending SCI error interrupt. 1 There is a pending SCI error interrupt.
2 TPMCH1	<b>TPM Channel 1 Interrupt Pending</b> — This read-only bit indicates there is a pending TPM channel 1 overflow (CH1F) interrupt from TPM module. Clearing the corresponding flag in TPM module clears this bit. Reset also clears this bit. 0 There is no pending TPM channel 1 overflow interrupt. 1 There is a pending TPM channel 1 overflow interrupt.
1 TPMCH0	<b>TPM Channel 0 Interrupt Pending</b> — This read-only bit indicates there is a pending TPM channel 0 overflow (CH0F) interrupt from TPM module. Clearing the corresponding flag in TPM module clears this bit. Reset also clears this bit. 0 There is no pending TPM channel 0 overflow interrupt. 1 There is a pending TPM channel 0 overflow interrupt.
0 TPM	<b>TPM Interrupt Pending</b> — This read-only bit indicates there is a pending TPM overflow (TOF) interrupt from TPM module. Clearing the corresponding flag in TPM module clears this bit. Reset also clears this bit. 0 There is no pending TPM overflow interrupt. 1 There is a pending TPM overflow interrupt.





# Chapter 6

## Parallel Input/Output

### 6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9RS08LA8 has five I/O ports which include a total of 33 general-purpose I/O pins. See [Chapter 2, “Pins and Connections”](#), for more information about pin assignments and external hardware considerations for these pins.

All of these I/O pins are shared with on-chip peripheral functions as shown in [Table 2-1](#). The peripheral modules have priority over the I/Os so that when a peripheral is enabled, the I/O functions associated with the shared pins are disabled. After reset, the shared peripheral functions are disabled so that the pins are controlled by the I/O. All of the I/Os are configured as inputs ( $PTxDDn = 0$ ) with pullup/pulldown devices disabled ( $PTxPEn = 0$ ), except for output-only pin PTC6, which defaults to the BKGD/MS function.

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](#).

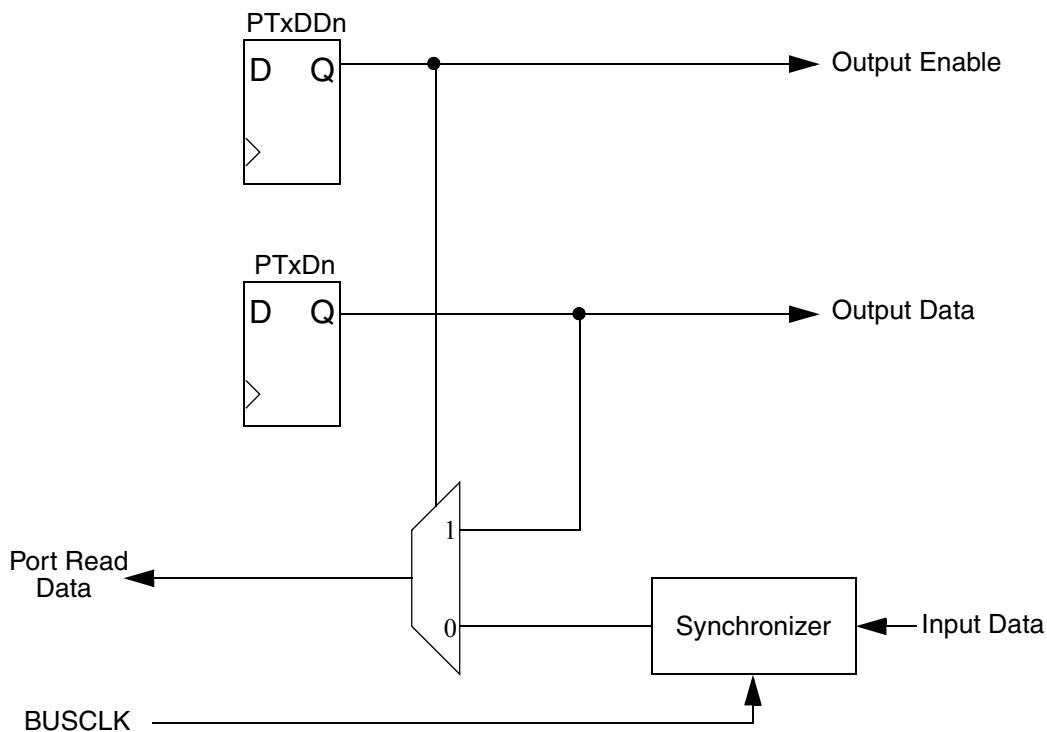


Figure 6-1. Parallel I/O Block Diagram

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 alternative digital function and an analog function, the analog function has priority. 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 temporarily with an old data value that happened to be in the port data register.

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 pullup/pulldown and slew rate for the pins.

## 6.2 Pin Behavior in Low-Power Modes

In wait and stop modes, all pin states are maintained because internal logic stays powered up. Upon recovery, all pin functions are the same as before entering stop.

## 6.3 Parallel I/O and Pin Control Registers

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

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

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

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

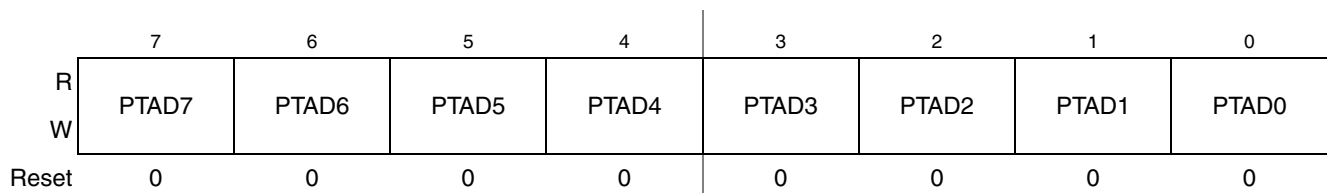


Figure 6-2. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
7:0 PTAD[7:0]	<b>Port A Data Register Bits</b> — 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.

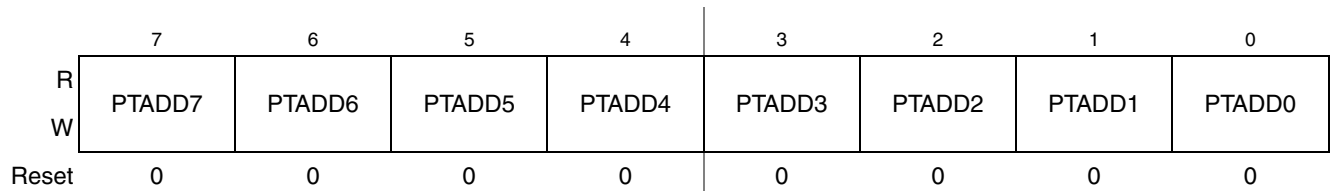


Figure 6-3. Data Direction for Port A Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
7:0 PTADD[7:0]	<b>Data Direction for Port A Bits</b> — 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.

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

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

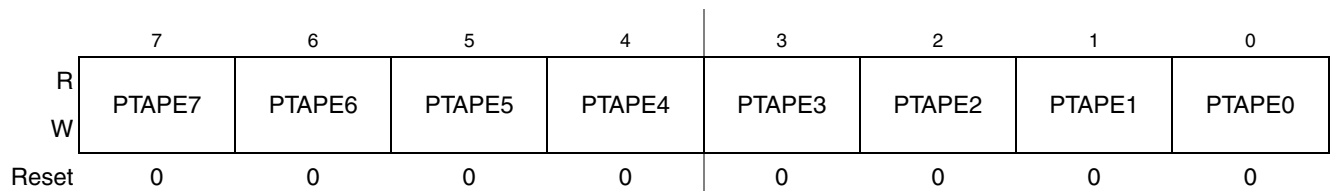
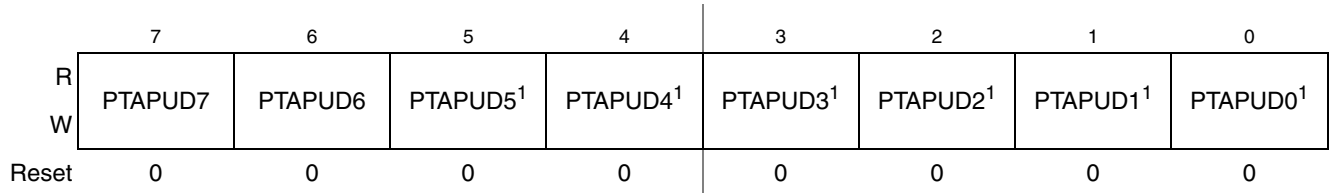


Figure 6-4. Internal Pulling Enable for Port A (PTAPE)

Table 6-3. PTAPE Register Field Descriptions

Field	Description
[7:0] PTAPE[7:0]	<b>Internal Pullup Enable for Port A Bits</b> — 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 pulling devices are disabled. 0 Internal pulling device disabled for port A bit n. 1 Internal pulling device enabled for port A bit n.

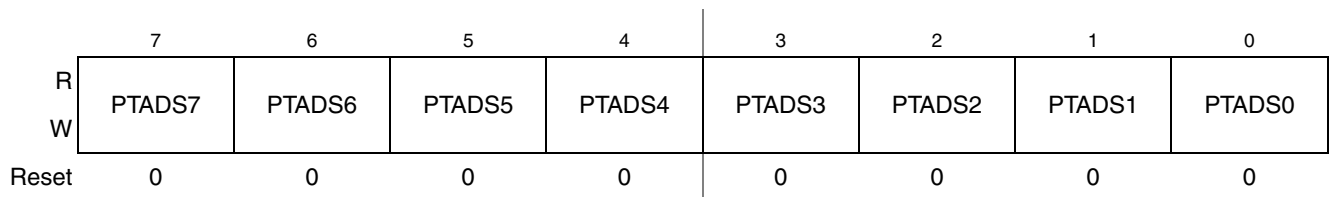


<sup>1</sup> V<sub>DD</sub> must be connected to V<sub>LL3</sub> externally if this port will be pulled up. For more informations. Please see [Chapter 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#).

**Figure 6-5. Pullup/Pulldown Device Control for Port A (PTAPUD)**

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

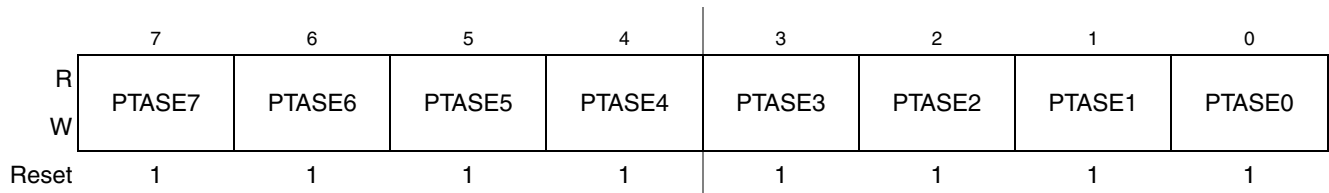
Field	Description
7:0 PTAPUD [7:0]	<b>Pullup/Pulldown Device Control for Port A Bits</b> — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTA pin. The actual pullup/pulldown device is only enabled by enabling the associated PTAPE bit. 0 Internal pullup device is selected for port A bin n. 1 Internal pulldown device is selected for port A bin n.



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

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

Field	Description
7:0 PTADS[7:0]	<b>Output Drive Strength Selection for Port A Bits</b> — Each of these control bits selects between low and high output driver 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 bin n.



**Figure 6-7. Output Slew Rate Control Enable (PTASE)**

Table 6-6. PTASE Register Field Descriptions

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

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

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

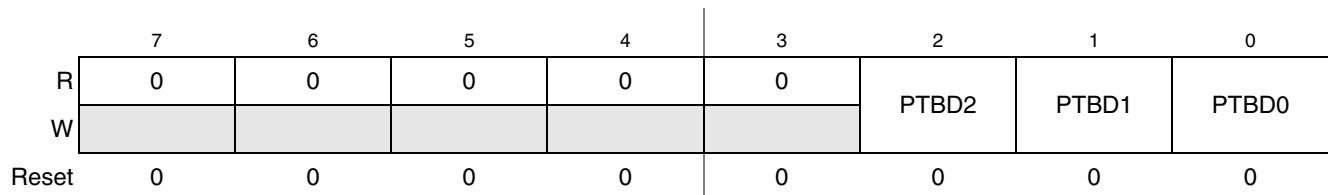


Figure 6-8. Port B Data Register (PTBD)

Table 6-7. PTBD Register Field Descriptions

Field	Description
2:0 PTBD[2:0]	<p><b>Port B Data Register Bits</b> — For port B pins that are inputs, reads return the logic level on the pin. For port B pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port B pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTBD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.</p>

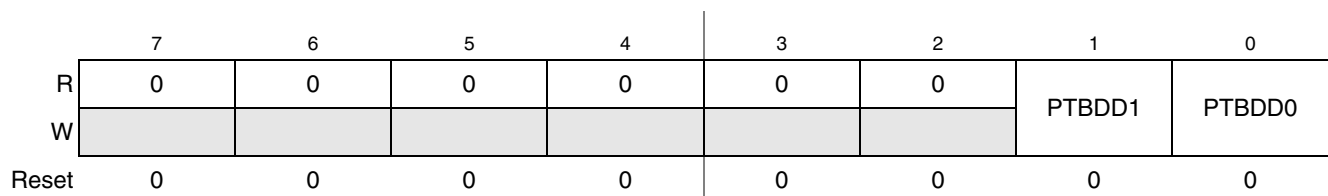


Figure 6-9. Data Direction for Port B (PTBDD)

Table 6-8. PTBDD Register Field Descriptions

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

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

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

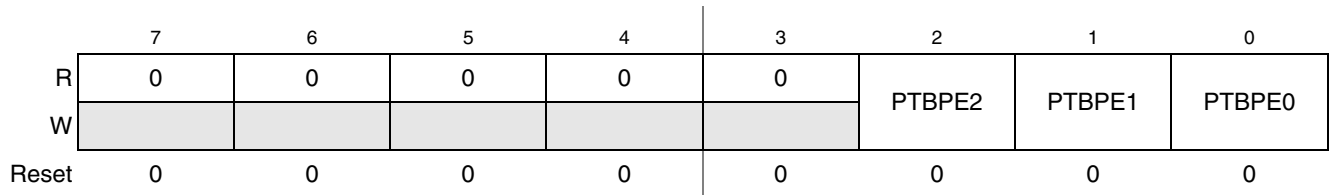


Figure 6-10. Internal Pulling Enable for Port B (PTBPE)

Table 6-9. PTBPE Register Field Descriptions

Field	Description
2:0 PTBPE[2:0]	<p><b>Internal Pullup Enable for Port B Bits</b> — Each of these control bits determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled.</p> <p>0 Internal pulling device disabled for port B bit n. 1 Internal pulling device enabled for port B bit n.</p>

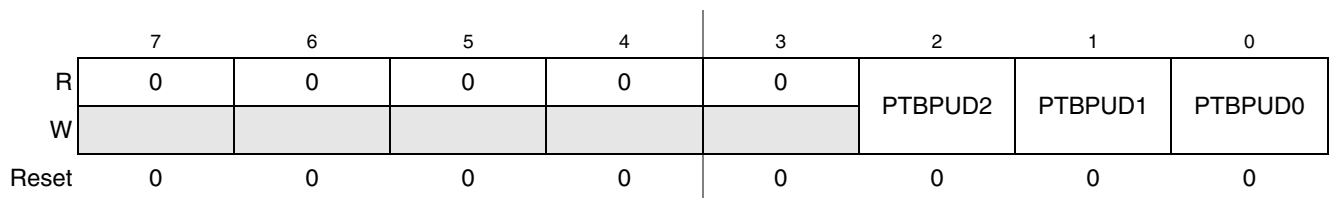


Figure 6-11. Pullup/Pulldown Device Control for Port B (PTBPUD)

Table 6-10. PTBPUD Register Field Descriptions

Field	Description
2:0 PTBPUD[2:0]	<p><b>Pullup/Pulldown Device Control for Port B Bits</b> — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTB pin. The actual pullup/pulldown device is only enabled by enabling the associated PTBPE bit.</p> <p>0 Internal pullup device is selected for port B bin n. 1 Internal pulldown device is selected for port B bin n.</p>

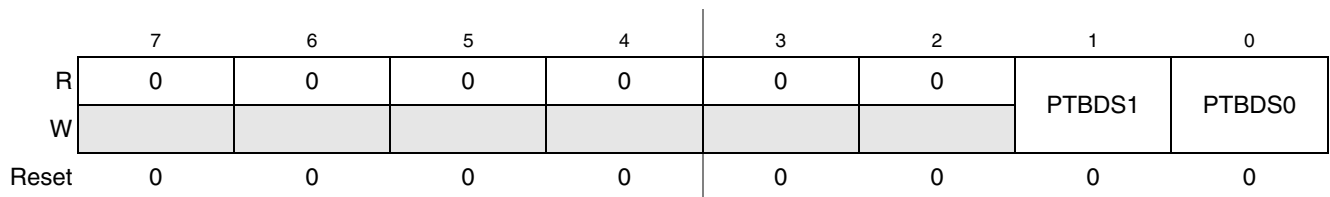


Figure 6-12. Drive Strength Selection for Port B (PTBDS)

Table 6-11. PTBDS Register Field Descriptions

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

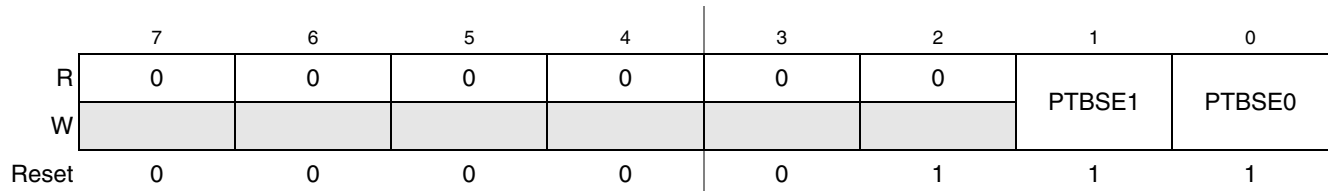


Figure 6-13. Output Slew Rate Control Enable (PTBSE)

Table 6-12. PTBSE Register Field Descriptions

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

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

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

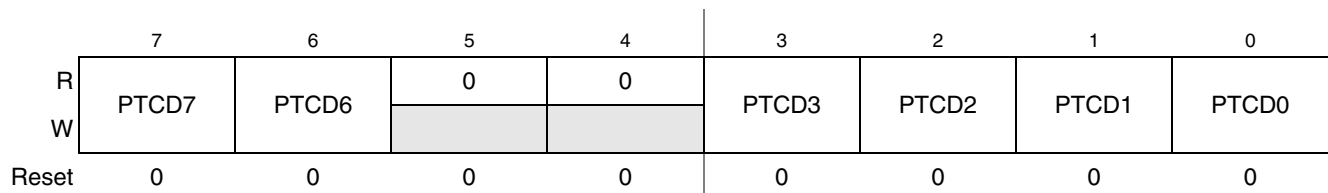


Figure 6-14. Port C Data Register (PTCD)

Table 6-13. PTCD Register Field Descriptions

Field	Description
7:6,3:0 PTCD [7:6,3:0]	<b>Port C Data Register Bits</b> — For port C pins that are inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port C pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTCD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

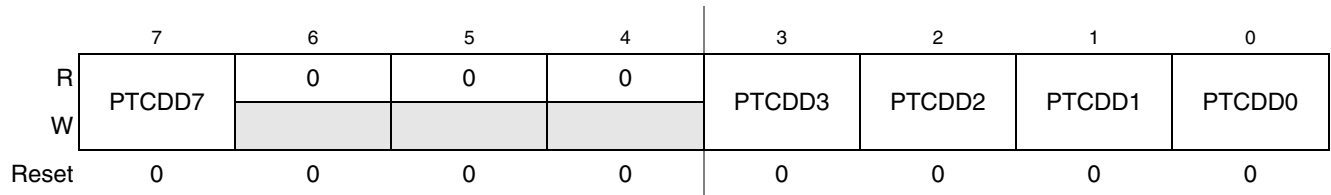


Figure 6-15. Data Direction for Port C (PTCDD)

Table 6-14. PTCDD Register Field Descriptions

Field	Description
7,3:0 PTCDD[7,3:0]	<b>Data Direction for Port C Bits</b> — These read/write bits control the direction of port C pins and what is read for PTCDD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port C bit n and PTCDD reads return the contents of PTCDDn.

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

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

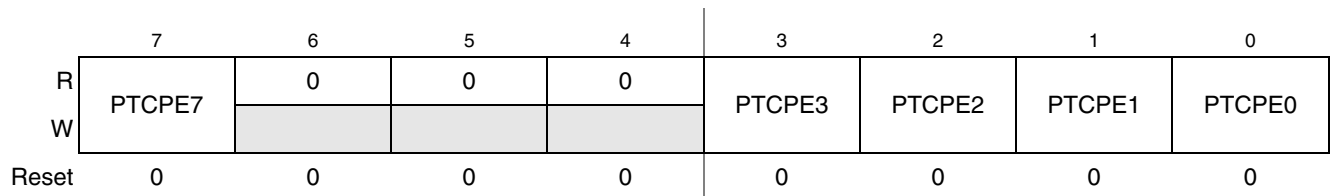
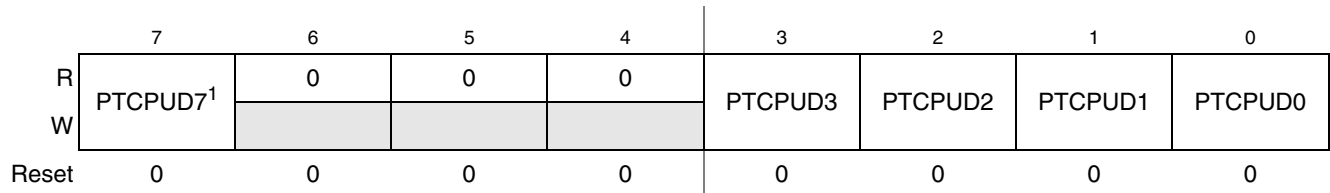


Figure 6-16. Internal Pulling Enable for Port C (PTCPE)

Table 6-15. PTCPE Register Field Descriptions

Field	Description
7,3:0 PTCPE [7,3:0]	<b>Internal Pullup Enable for Port C Bits</b> — Each of these control bits determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled. 0 Internal pulling device disabled for port C bit n. 1 Internal pulling device enabled for port C bit n.



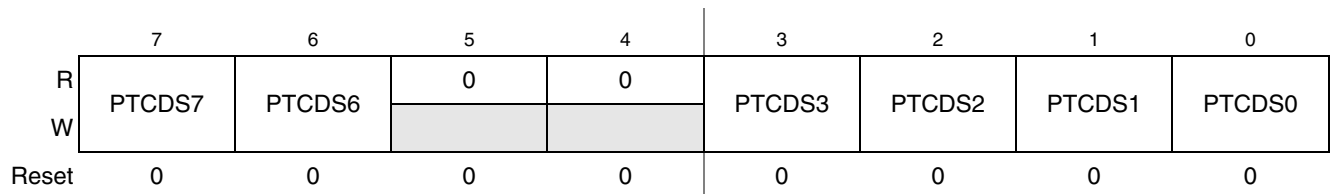


<sup>1</sup>  $V_{DD}$  must be connected to  $V_{LL3}$  externally if this port will be pulled up. For more informations. Please see [Chapter 10, "Liquid Crystal Display Module \(S08LCDV2\)"](#).

**Figure 6-18. Pullup/Pulldown Device Control for Port C (PTCPUD)**

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

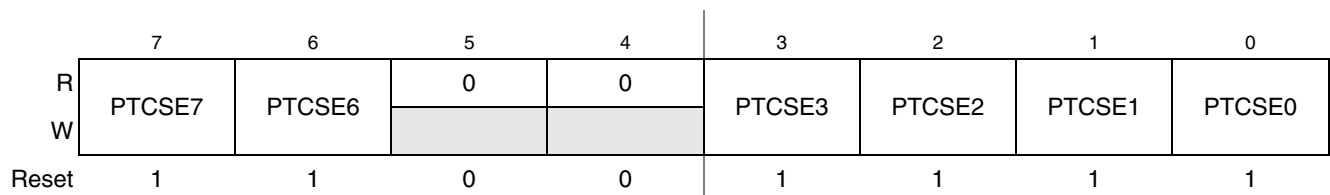
Field	Description
7,3:0 PTCPUD [7,3:0]	<b>Pullup/Pulldown Device Control for Port C Bits</b> — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTC pin. The actual pullup/pulldown device is only enabled by enabling the associated PTCPE bit. 0 Internal pullup device is selected for port C bin n. 1 Internal pulldown device is selected for port C bin n.



**Figure 6-19. Drive Strength Selection for Port C (PTCDS)**

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

Field	Description
7:6,3:0 PTCDS[7:6]	<b>Output Drive Strength Selection for Port A Bits</b> — Each of these control bits selects between low and high output driver for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port C bit n. 1 High output drive strength selected for port C bin n.



**Figure 6-20. Output Slew Rate Control Enable for Port C (PTCSE)**

Table 6-18. PTCSE Register Field Descriptions

Field	Description
7:6,3:0 PTCSE [7:6,3:0]	<p><b>Output Slew Rate Control Enable for Port C Bits</b> — Each of these control bits determine whether output slew rate control is enabled for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect.</p> <p>0 Output slew rate control disabled for port C bit n. 1 Output slew rate control enabled for port C bit n.</p>

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

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

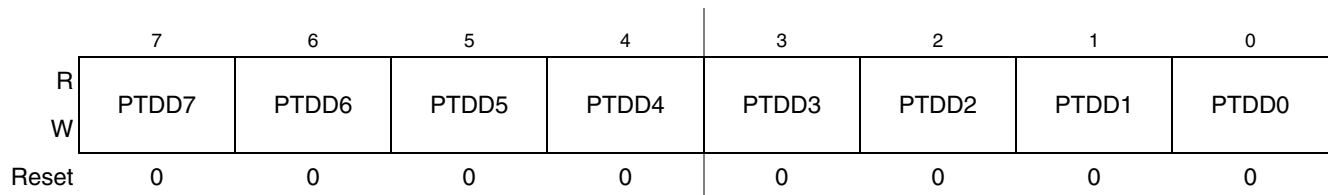


Figure 6-21. Port D Data Register (PTDD)

Table 6-19. PTDD Register Field Descriptions

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

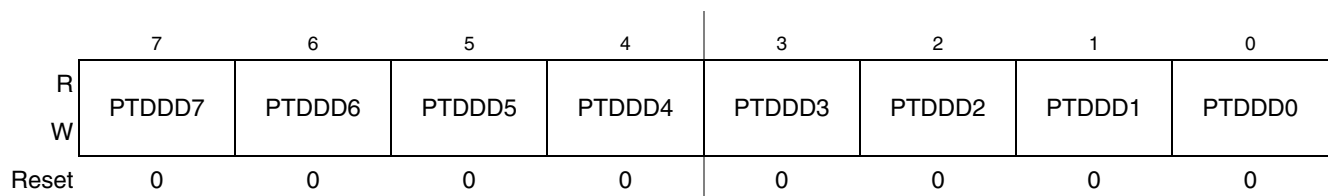


Figure 6-22. Data Direction for Port D (PTDDD)

Table 6-20. PTDDD Register Field Descriptions

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

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

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

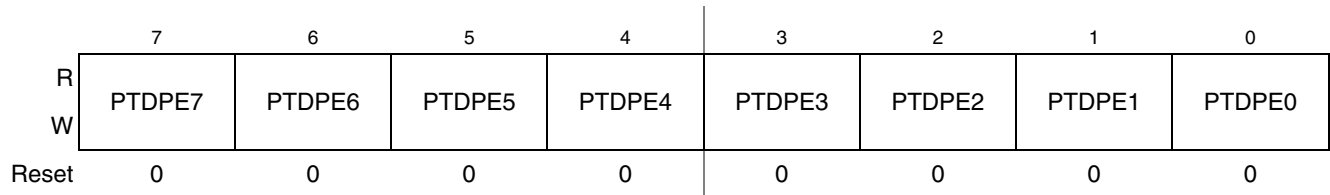
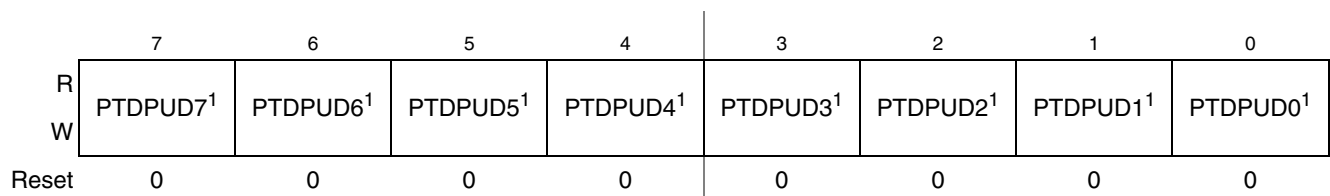


Figure 6-23. Internal Pulling Enable for Port D (PTDPE)

Table 6-21. PTDPE Register Field Descriptions

Field	Description
7:0 PTDPE[7:0]	<p><b>Internal Pullup Enable for Port D Bits</b> — Each of these control bits determines if the internal pullup device is enabled for the associated PTD pin. For port D pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled.</p> <p>0 Internal pulling device disabled for port D bit n. 1 Internal pulling device enabled for port D bit n.</p>



<sup>1</sup>  $V_{DD}$  must be connected to  $V_{LL3}$  externally if this port will be pulled up. For more informations. Please see [Chapter 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#).

Figure 6-26. Pullup/Pulldown Device Control for Port D (PTDPUD)

Table 6-22. PTDPUD Register Field Descriptions

Field	Description
7:0 PTDPUD [7:0]	<p><b>Pullup/Pulldown Device Control for Port D Bits</b> — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTD pin. The actual pullup/pulldown device is only enabled by enabling the associated PTDPE bit.</p> <p>0 Internal pullup device is selected for port D bin n. 1 Internal pulldown device is selected for port D bin n.</p>

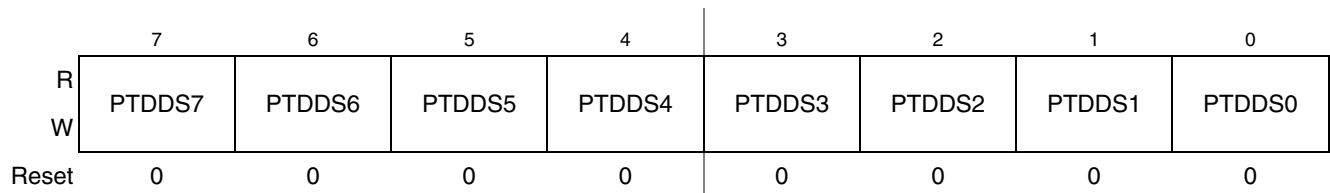


Figure 6-27. Drive Strength Selection for Port D (PTDDS)

Table 6-23. PTDDS Register Field Descriptions

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

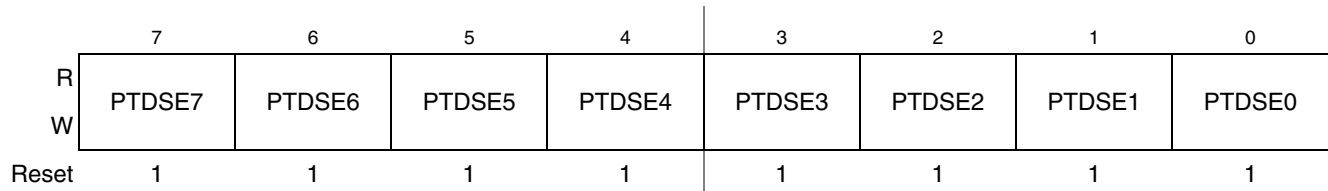


Figure 6-28. Output Slew Rate Control Enable for Port D (PTDSE)

Table 6-24. PTDSE Register Field Descriptions

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

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

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

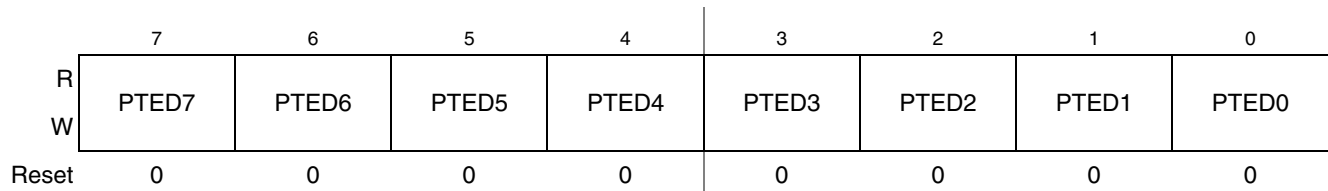


Figure 6-29. Port E Data Register (PTED)

Table 6-25. PTED Register Field Descriptions

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

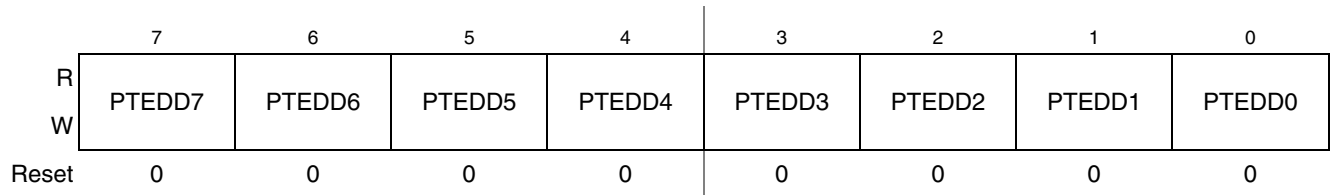


Figure 6-30. Data Direction for Port E (PTEDD)

Table 6-26. PTEDD Register Field Descriptions

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

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

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

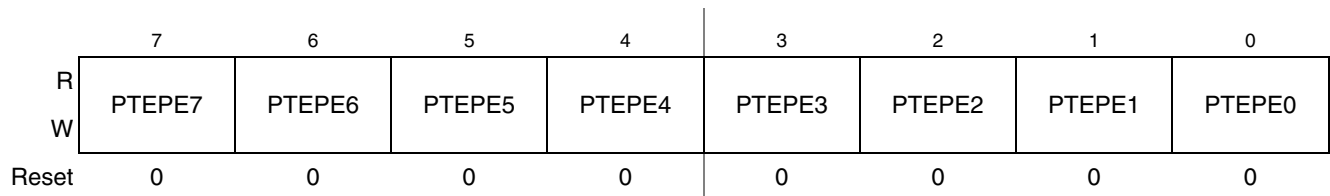
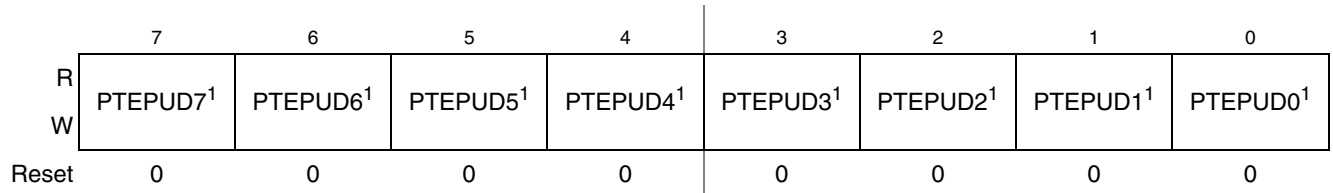


Figure 6-31. Internal Pulling Enable for Port E (PTEPE)

Table 6-27. PTEPE Register Field Descriptions

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

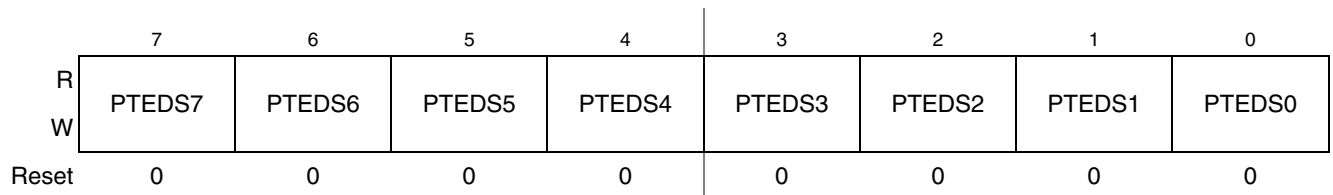


<sup>1</sup> V<sub>DD</sub> must be connected to V<sub>LL3</sub> externally if this port will be pulled up. For more informations. Please see Chapter 10, “Liquid Crystal Display Module (S08LCDV2)”.

**Figure 6-33. Pullup/Pulldown Device Control for Port E (PTEPUD)**

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

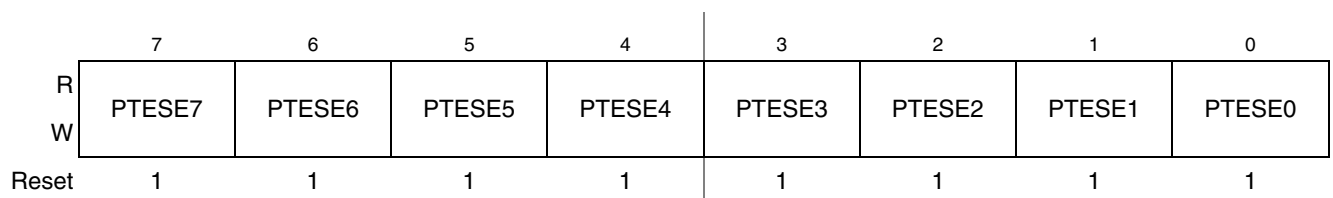
Field	Description
7:0 PTEPUD[7:0]	<b>Pullup/Pulldown Device Control for Port E Bits</b> — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTE pin. The actual pullup/pulldown device is only enabled by enabling the associated PTEPE bit. 0 Internal pullup device is selected for port E bin n. 1 Internal pulldown device is selected for port E bin n.



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

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

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



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

Table 6-30. PTESE Register Field Descriptions

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





---

# Chapter 7

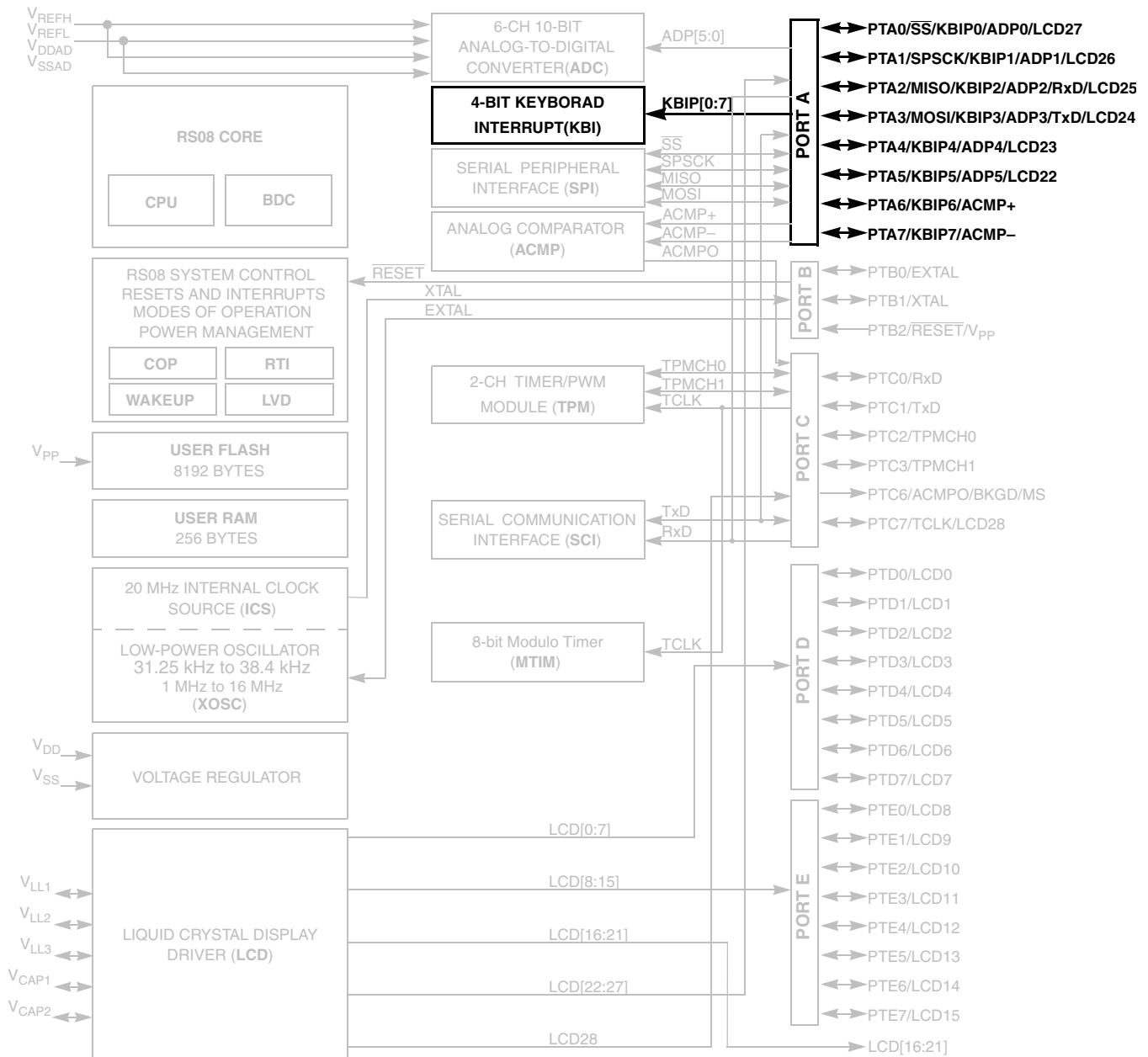
## Keyboard Interrupt (RS08KBIV1)

### 7.1 Introduction

The keyboard interrupt (KBI) module provides up to four independently enabled external interrupt sources.

All KBI pins share KBI functionality with LCD pins. KBI functionality must be disabled for these pins to operate as LCD pins.

[Figure 7-1](#) shows the MC9RS08LA8 block diagram highlighting the KBI block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 7-1. MC9RS08LA8 Block Diagram Highlighting KBI Block and Pins**

### 7.1.1 Features

The KBI features include:

- Each keyboard interrupt pin has individual pin enable bit

- 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

## 7.1.2 Modes of Operation

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

### 7.1.2.1 Operation in Wait Mode

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

### 7.1.2.2 Operation in Stop Mode

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

### 7.1.2.3 Operation in Active Background Mode

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

## 7.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 1-1.

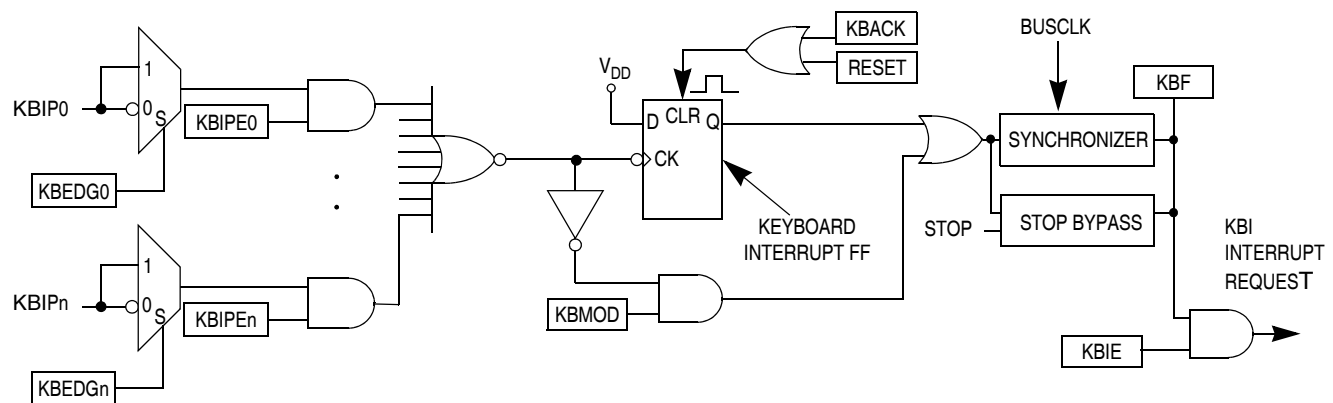


Figure 7-2. Keyboard Block Diagram

## 7.2 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.

Figure 7-1 shows KBI signal properties.

**Table 7-1. Interface A — Detailed Signal Descriptions**

Signal	I/O	Description
KBIPn	I	Keyboard interrupt pins

## 7.3 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 [Chapter 4, “Memory”](#) for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names and relative address offsets.

The KBI registers are summarized in [Table 7-2](#).

**Table 7-2. KBI Register Summary**

Name		7	6	5	4	3	2	1	0
KBISC	R	0	0	0	0	KBF	0	KBIE	KBMOD
	W						KBACK		
KBIPE	R	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
	W								
KBIES	R	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
	W								

### 7.3.1 KBI Status and Control Register (KBISC)

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

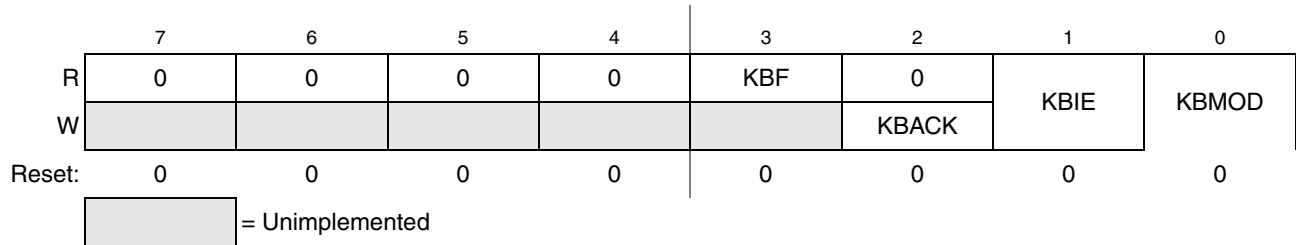


Figure 7-3. KBI Status and Control Register (KBISC)

Table 7-3. KBISC Register Field Descriptions

Field	Description
3 KBF	<b>Keyboard Interrupt Flag</b> — KBF indicates that a keyboard interrupt is detected. Writes have no effect on KBF. 0 No keyboard interrupt detected. 1 Keyboard interrupt detected.
2 KBACK	<b>Keyboard Acknowledge</b> — Writing a 1 to KBACK is part of the flag-clearing mechanism. KBACK always reads as 0.
1 KBIE	<b>Keyboard Interrupt Enable</b> — KBIE enables keyboard interrupt requests. 0 Keyboard interrupt request not enabled. 1 Keyboard interrupt request enabled.
0 KBMOD	<b>Keyboard Detection Mode</b> — 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.

### 7.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.



Figure 7-4. KBI Pin Enable Register (KBIPE)

Table 7-4. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPEn	<b>Keyboard Pin Enables</b> — Each of the KBIPEn bits enables the corresponding keyboard interrupt pin. 0 Corresponding pin not enabled as keyboard interrupt. 1 Corresponding pin enabled as keyboard interrupt.

### 7.3.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

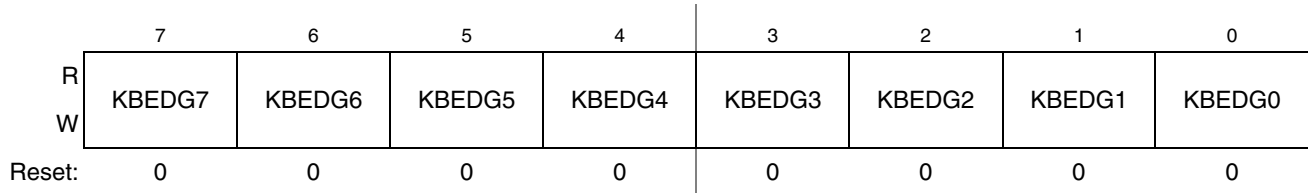


Figure 7-5. KBI Edge Select Register (KBIES)

Table 7-5. KBIES Register Field Descriptions

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

## 7.4 Functional Description

This on-chip peripheral module is called a keyboard interrupt (KBI) module because it was originally 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 its 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).

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

When the MCU enters stop mode, the synchronous edge-detection logic is bypassed (because clocks are stopped). In stop mode, KBI inputs act as asynchronous level-sensitive inputs so they can wake the MCU from stop mode.

### 7.4.1 Edge Only Sensitivity

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.

## 7.4.2 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.

## 7.4.3 KBI Pullup/Pulldown Device

The KBI pins does not automatically configure an internal pullup/pulldown device when a KBI pin is enabled. An internal pull device can be used by configuring the associated I/O port pull device enable register (PTBPE) and pullup/pulldown control register (PTBPUD).

## 7.4.4 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 must do the following:

1. Mask keyboard interrupts by clearing KBIE in KBISC.
2. If using internal pullup/pulldown device, configure the associated I/O port pullup/pulldown device.
3. Enable the KBI polarity by setting the appropriate KBEDGn bits in KBIES.
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 8

## Central Processor Unit (RS08CPUV1)

### 8.1 Introduction

This chapter is a summary of information about the registers, addressing modes, and instruction set of the RS08 Family CPU. For a more detailed discussion, refer to the RS08 Core Reference Manual, volume 1, Freescale Semiconductor document order number RS08RMv1.

The RS08 CPU has been developed to target extremely low-cost embedded applications using a process-independent design methodology, allowing it to keep pace with rapid developments in silicon processing technology.

The main features of the RS08 core are:

- Streamlined programmer's model
- Subset of HCS08 instruction set with minor instruction extensions
- Minimal instruction set for cost-sensitive embedded applications
- New instructions for shadow program counter manipulation, SHA and SLA
- New short and tiny addressing modes for code size optimization
- 16K bytes accessible memory space
- Reset will fetch the first instruction from \$3FFD
- Low-power modes supported through the execution of the STOP and WAIT instructions
- Debug and flash programming support using the background debug controller module
- Illegal address and opcode detection with reset

### 8.2 Programmer's Model and CPU Registers

Figure 8-1 shows the programmer's model for the RS08 CPU. These registers are not located in the memory map of the microcontroller. They are built directly inside the CPU logic.

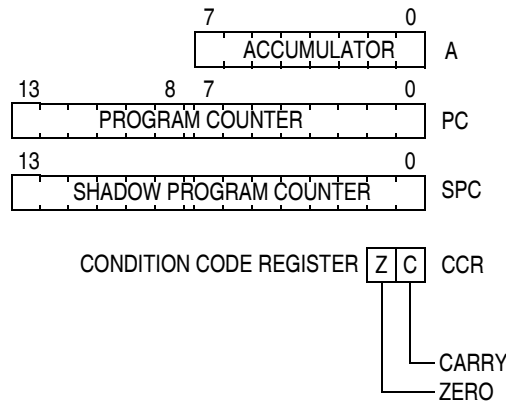


Figure 8-1. CPU Registers

In addition to the CPU registers, there are three memory mapped registers that are tightly coupled with the core address generation during data read and write operations. They are the indexed data register (D[X]), the index register (X), and the page select register (PAGESEL). These registers are located at \$000E, \$000F, and \$001F, respectively.

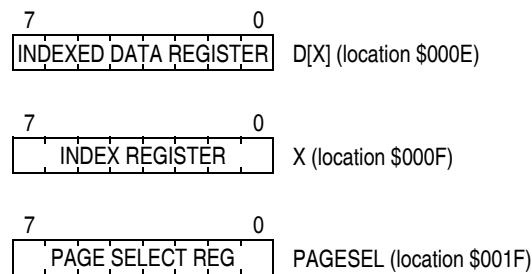


Figure 8-2. Memory Mapped Registers

### 8.2.1 Accumulator (A)

This general-purpose 8-bit register is the primary data register for RS08 MCUs. Data can be read from memory into A with a load accumulator (LDA) instruction. The data in A can be written into memory with a store accumulator (STA) instruction. Various addressing mode variations allow a great deal of flexibility in specifying the memory location involved in a load or store instruction. Exchange instructions allow values to be exchanged between A and SPC high (SHA) and also between A and SPC low (SLA).

Arithmetic, shift, and logical operations can be performed on the value in A as in ADD, SUB, RORA, INCA, DECA, AND, ORA, EOR, etc. In some of these instructions, such as INCA and LSLA, the value in A is the only input operand and the result replaces the value in A. In other cases, such as ADD and AND, there are two operands: the value in A and a second value from memory. The result of the arithmetic or logical operation replaces the value in A.

Some instructions, such as memory-to-memory move instructions (MOV), do not use the accumulator. DBNZ also relieves A because it allows a loop counter to be implemented in a memory variable rather than the accumulator.

During reset, the accumulator is loaded with \$00.

## 8.2.2 Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction or operand to be fetched.

During normal execution, the program counter automatically increments to the next sequential memory location each time an instruction or operand is fetched. Jump, branch, 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 \$3FFD and the program will start execution from this specific location.

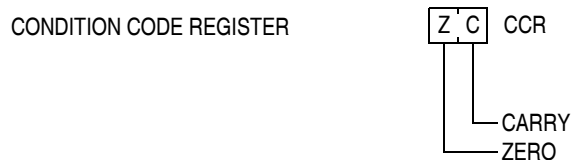
## 8.2.3 Shadow Program Counter (SPC)

The shadow program counter is a 14-bit register. During a subroutine call using either a JSR or a BSR instruction, the return address will be saved into the SPC. Upon completion of the subroutine, the RTS instruction will restore the content of the program counter from the shadow program counter.

During reset, the shadow program counter is loaded with \$3FFD.

## 8.2.4 Condition Code Register (CCR)

The 2-bit condition code register contains two status flags. The content of the CCR in the RS08 is not directly readable. The CCR bits can be tested using conditional branch instructions such as BCC and BEQ. These two register bits are directly accessible through the BDC interface. The following paragraphs provide detailed information about the CCR bits and how they are used. [Figure 8-3](#) identifies the CCR bits and their bit positions.



**Figure 8-3. Condition Code Register (CCR)**

The status bits (Z and C) are cleared to 0 after reset.

The two status bits indicate the results of arithmetic and other instructions. Conditional branch instructions will either branch to a new program location or allow the program to continue to the next instruction after the branch, depending on the values in the CCR status bit. Conditional branch instructions, such as BCC, BCS, and BNE, cause a branch depending on the state of a single CCR bit.

Often, the conditional branch immediately follows the instruction that caused the CCR bit(s) to be updated, as in this sequence:

```

        cmp     #5           ;compare accumulator A to 5
        blo    lower        ;branch if A smaller 5
more:   decr     ;do this if A not higher than or same as 5
lower:

```

Other instructions may be executed between the test and the conditional branch as long as the only instructions used are those which do not disturb the CCR bits that affect the conditional branch. For instance, a test is performed in a subroutine or function and the conditional branch is not executed until the subroutine has returned to the main program. This is a form of parameter passing (that is, information is returned to the calling program in the condition code bits).

#### Z — Zero Flag

The Z bit is set to indicate the result of an operation was \$00.

Branch if equal (BEQ) and branch if not equal (BNE) are simple branches that branch based solely on the value in the Z bit. All load, store, move, arithmetic, logical, shift, and rotate instructions cause the Z bit to be updated.

#### C — Carry

After an addition operation, the C bit is set if the source operands were both greater than or equal to \$80 or if one of the operands was greater than or equal to \$80 and the result was less than \$80. This is equivalent to an unsigned overflow. A subtract or compare performs a subtraction of a memory operand from the contents of a CPU register so after a subtract operation, the C bit is set if the unsigned value of the memory operand was greater than the unsigned value of the CPU register. This is equivalent to an unsigned borrow or underflow.

Branch if carry clear (BCC) and branch if carry set (BCS) are branches that branch based solely on the value in the C bit. The C bit is also used by the unsigned branches BLO and BHS. Add, subtract, shift, and rotate instructions cause the C bit to be updated. The branch if bit set (BRSET) and branch if bit clear (BRCLR) instructions copy the tested bit into the C bit to facilitate efficient serial-to-parallel conversion algorithms. Set carry (SEC) and clear carry (CLC) allow the carry bit to be set or cleared directly. This is useful in combination with the shift and rotate instructions and for routines that pass status information back to a main program, from a subroutine, in the C bit.

The C bit is included in shift and rotate operations so those operations can easily be extended to multi-byte operands. The shift and rotate operations can be considered 9-bit shifts that include an 8-bit operand or CPU register and the carry bit of the CCR. After a logical shift, C holds the bit that was shifted out of the 8-bit operand. If a rotate instruction is used next, this C bit is shifted into the operand for the rotate, and the bit that gets shifted out the other end of the operand replaces the value in C so it can be used in subsequent rotate instructions.

### 8.2.5 Indexed Data Register (D[X])

This 8-bit indexed data register allows the user to access the data in the direct page address space indexed by X. This register resides at the memory mapped location \$000E. For details on the D[X] register, please refer to [Section 8.3.8, “Indexed Addressing Mode \(IX, Implemented by Pseudo Instructions\).”](#)

### 8.2.6 Index Register (X)

This 8-bit index register allows the user to index or address any location in the direct page address space. This register resides at the memory mapped location \$000F. For details on the X register, please refer to [Section 8.3.8, “Indexed Addressing Mode \(IX, Implemented by Pseudo Instructions\).”](#)

### 8.2.7 Page Select Register (PAGESEL)

This 8-bit page select register allows the user to access all memory locations in the entire 16K-byte address space through a page window located from \$00C0 to \$00FF. This register resides at the memory mapped location \$001F. For details on the PAGESEL register, please refer to the RS08 Core Reference Manual.

## 8.3 Addressing Modes

Whenever the MCU reads information from memory or writes information into memory, an addressing mode is used to determine the exact address where the information is read from or written to. This section explains several addressing modes and how each is useful in different programming situations.

Every opcode tells the CPU to perform a certain operation in a certain way. Many instructions, such as load accumulator (LDA), allow several different ways to specify the memory location to be operated on, and each addressing mode variation requires a separate opcode. All of these variations use the same instruction mnemonic, and the assembler knows which opcode to use based on the syntax and location of the operand field. In some cases, special characters are used to indicate a specific addressing mode (such as the # [pound] symbol, which indicates immediate addressing mode). In other cases, the value of the operand tells the assembler which addressing mode to use. For example, the assembler chooses short addressing mode instead of direct addressing mode if the operand address is from \$0000 to \$001F. Besides allowing the assembler to choose the addressing mode based on the operand address, assembler directives can also be used to force direct or tiny/short addressing mode by using the “>” or “<” prefix before the operand, respectively.

Some instructions use more than one addressing mode. For example, the move instructions use one addressing mode to access the source value from memory and a second addressing mode to access the destination memory location. For these move instructions, both addressing modes are listed in the documentation. All branch instructions use relative (REL) addressing mode to determine the destination for the branch, but BRCLR, BRSET, CBEQ, and DBNZ also must access a memory operand. These instructions are classified by the addressing mode used for the memory operand, and the relative addressing mode for the branch offset is assumed.

The discussion in the following paragraphs includes how each addressing mode works and the syntax clues that instruct the assembler to use a specific addressing mode.

### 8.3.1 Inherent Addressing Mode (INH)

This addressing mode is used when the CPU inherently knows everything it needs to complete the instruction and no addressing information is supplied in the source code. Usually, the operands that the CPU needs are located in the CPU’s internal registers, as in LSLA, CLRA, INCA, SLA, RTS, and others. A few inherent instructions, including no operation (NOP) and background (BGND), have no operands.

### 8.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the offset address for branch instructions relative to the program counter. Typically, the programmer specifies the destination with a program label or an

expression in the operand field of the branch instruction; the assembler calculates the difference between the location counter (which points at the next address after the branch instruction at the time) and the address represented by the label or expression in the operand field. This difference is called the offset and is an 8-bit two's complement number. The assembler stores this offset in the object code for the branch instruction.

During execution, the CPU evaluates the condition that controls the branch. If the branch condition is true, the CPU sign-extends the offset to a 14-bit value, adds the offset to the current PC, and uses this as the address where it will fetch the next instruction and continue execution rather than continuing execution with the next instruction after the branch. Because the offset is an 8-bit two's complement value, the destination must be within the range  $-128$  to  $+127$  locations from the address that follows the last byte of object code for the branch instruction.

A common method to create a simple infinite loop is to use a branch instruction that branches to itself. This is sometimes used to end short code segments during debug. Typically, to get out of this infinite loop, use the debug host (through background commands) to stop the program, examine registers and memory, or to start execution from a new location. This construct is not used in normal application programs except in the case where the program has detected an error and wants to force the COP watchdog timer to timeout. (The branch in the infinite loop executes repeatedly until the watchdog timer eventually causes a reset.)

### 8.3.3 Immediate Addressing Mode (IMM)

In this addressing mode, the operand is located immediately after the opcode in the instruction stream. This addressing mode is used when the programmer wants to use an explicit value that is known at the time the program is written. A # (pound) symbol is used to tell the assembler to use the operand as a data value rather than an address where the desired value must be accessed.

The size of the immediate operand is always 8 bits. The assembler automatically will truncate or extend the operand as needed to match the size needed for the instruction. Most assemblers generate a warning if a 16-bit operand is provided.

It is the programmer's responsibility to use the # symbol to tell the assembler when immediate addressing must be used. The assembler does not consider it an error to leave off the # symbol because the resulting statement is still a valid instruction (although it may mean something different than the programmer intended).

### 8.3.4 Tiny Addressing Mode (TNY)

TNY addressing mode is capable of addressing only the first 16 bytes in the address map, from \$0000 to \$000F. This addressing mode is available for INC, DEC, ADD, and SUB instructions. A system can be optimized by placing the most computation-intensive data in this area of memory.

Because the 4-bit address is embedded in the opcode, only the least significant four bits of the address must be included in the instruction; this saves program space and execution time. During execution, the CPU adds 10 high-order 0s to the 4-bit operand address and uses the combined 14-bit address (\$000x) to access the intended operand.

### 8.3.5 Short Addressing Mode (SRT)

SRT addressing mode is capable of addressing only the first 32 bytes in the address map, from \$0000 to \$001F. This addressing mode is available for CLR, LDA, and STA instructions. A system can be optimized by placing the most computation-intensive data in this area of memory.

Because the 5-bit address is embedded in the opcode, only the least significant five bits of the address must be included in the instruction; this saves program space and execution time. During execution, the CPU adds nine high-order 0s to the 5-bit operand address and uses the combined 14-bit address (\$000x or \$001x) to access the intended operand.

### 8.3.6 Direct Addressing Mode (DIR)

DIR addressing mode is used to access operands located in direct address space (\$0000 through \$00FF).

During execution, the CPU adds six high-order 0s to the low byte of the direct address operand that follows the opcode. The CPU uses the combined 14-bit address (\$00xx) to access the intended operand.

### 8.3.7 Extended Addressing Mode (EXT)

In the extended addressing mode, the 14-bit address of the operand is included in the object code in the low-order 14 bits of the next two bytes after the opcode. This addressing mode is only used in JSR and JMP instructions for jump destination address in RS08 MCUs.

### 8.3.8 Indexed Addressing Mode (IX, Implemented by Pseudo Instructions)

Indexed addressing mode is sometimes called indirect addressing mode because an index register is used as a reference to access the intended operand.

An important feature of indexed addressing mode is that the operand address is computed during execution based on the current contents of the X index register located in \$000F of the memory map rather than being a constant address location that was determined during program assembly. This allows writing of a program that accesses different operand locations depending on the results of earlier program instructions (rather than accessing a location that was determined when the program was written).

The index addressing mode supported by the RS08 Family uses the register X located at \$000F as an index and D[X] register located at \$000E as the indexed data register. By programming the index register X, any location in the direct page can be read/written via the indexed data register D[X].

These pseudo instructions can be used with all instructions supporting direct, short, and tiny addressing modes by using the D[X] as the operand.

## 8.4 Special Operations

Most of what the CPU does is described by the instruction set, but a few special operations must be considered, such as how the CPU starts at the beginning of an application program after power is first applied. After the program begins running, the current instruction normally determines what the CPU will do next. Two exceptional events can cause the CPU to temporarily suspend normal program execution:

- Reset events force the CPU to start over at the beginning of the application program, which forces execution to start at \$3FFD.
- A host development system can cause the CPU to go to active background mode rather than continuing to the next instruction in the application program.

### 8.4.1 Reset Sequence

Processing begins at the trailing edge of a reset event. The number of things that can cause reset events can vary slightly from one RS08 derivative to another; however, the most common sources are: power-on reset, the external  $\overline{\text{RESET}}$  pin, low-voltage reset, COP watchdog timeout, illegal opcode detect, and illegal address access. For more information about how the MCU recognizes reset events and determines the difference between internal and external causes, refer to the [Resets and Interrupts](#) chapter.

Reset events force the MCU to immediately stop what it is doing and begin responding to reset. Any instruction that was in process will be aborted immediately without completing any remaining clock cycles. A short sequence of activities is completed to decide whether the source of reset was internal or external and to record the cause of reset. For the remainder of the time, the reset source remains active and the internal clocks are stopped to save power. At the trailing edge of the reset event, the clocks resume and the CPU exits from the reset condition. The program counter is reset to \$3FFD and an instruction fetch will be started after the release of reset.

For the device to execute code from the on-chip memory starting from \$3FFD after reset, care must be taken to not force the BKDG pin low on the end of reset because this will force the device into active background mode where the CPU will wait for a command from the background communication interface.

### 8.4.2 Interrupts

The interrupt mechanism in RS08 is not used to interrupt the normal flow of instructions; it is used to wake up the RS08 from wait and stop modes. In run mode, interrupt events must be polled by the CPU. The interrupt feature is not compatible with Freescale's HC05, HC08, or HCS08 Families.

### 8.4.3 Wait and Stop Mode

Wait and stop modes are entered by executing a WAIT or STOP instruction, respectively. In these modes, the clocks to the CPU are shut down to save power and CPU activity is suspended. The CPU remains in this low-power state until an interrupt or reset event wakes it up. Please refer to the [Resets and Interrupts](#) chapter for the effects of wait and stop on other device peripherals.

### 8.4.4 Active Background Mode

Active background mode refers to the condition in which the CPU has stopped executing user program instructions and is waiting for serial commands from the background debug system. Refer to the [Development Support](#) chapter for detailed information on active background mode.

The arithmetic left shift pseudo instruction is also available because its operation is identical to logical shift left.



## 8.5 Summary Instruction Table

### Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in [Table 8-1](#) through [Table 8-2](#).

#### Operators

( )	=	Contents of register or memory location shown inside parentheses
←	=	Is loaded with (read: “gets”)
↔	=	Exchange with
&	=	Boolean AND
	=	Boolean OR
⊕	=	Boolean exclusive-OR
:	=	Concatenate
+	=	Add

#### CPU registers

A	=	Accumulator
CCR	=	Condition code register
PC	=	Program counter
PCH	=	Program counter, higher order (most significant) six bits
PCL	=	Program counter, lower order (least significant) eight bits
SPC	=	Shadow program counter
SPCH	=	Shadow program counter, higher order (most significant) six bits
SPCL	=	Shadow program counter, lower order (least significant) eight bits

#### Memory and addressing

M	=	A memory location or absolute data, depending on addressing mode
<i>rel</i>	=	The relative offset, which is the two’s complement number stored in the last byte of machine code corresponding to a branch instruction
X	=	Pseudo index register, memory location \$000F
,X or D[X]	=	Memory location \$000E pointing to the memory location defined by the pseudo index register (location \$000F)

#### Condition code register (CCR) bits

Z	=	Zero indicator
C	=	Carry/borrow

#### CCR activity notation

–	=	Bit not affected
0	=	Bit forced to 0
1	=	Bit forced to 1
P	=	Bit set or cleared according to results of operation
U	=	Undefined after the operation

#### Machine coding notation

- dd = Low-order eight bits of a direct address \$0000–\$00FF (high byte assumed to be \$00)
- ii = One byte of immediate data
- hh = High-order 6-bit of 14-bit extended address prefixed with 2-bit of 0
- ll = Low-order byte of 14-bit extended address
- rr = Relative offset

### Source form

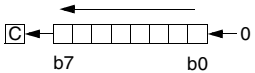
Everything in the source forms columns, *except expressions in italic characters*, is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic is always a literal expression. All commas, pound signs (#), parentheses, and plus signs (+) are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

- n* — Any label or expression that evaluates to a single integer in the range 0–7.
- x* — Any label or expression that evaluates to a single hexadecimal integer in the range \$0–\$F.
- opr8i* — Any label or expression that evaluates to an 8-bit immediate value.
- opr4a* — Any label or expression that evaluates to a Tiny address (4-bit value). The instruction treats this 4-bit value as the low order four bits of an address in the 16-Kbyte address space (\$0000–\$000F). This 4-bit value is embedded in the low order four bits in the opcode.
- opr5a* — Any label or expression that evaluates to a Short address (5-bit value). The instruction treats this 5-bit value as the low order five bits of an address in the 16-Kbyte address space (\$0000–\$001F). This 5-bit value is embedded in the low order 5 bits in the opcode.
- opr8a* — Any label or expression that evaluates to an 8-bit value. The instruction treats this 8-bit value as the low order eight bits of an address in the 16-Kbyte address space (\$0000–\$00FF).
- opr16a* — Any label or expression that evaluates to a 14-bit value. On the RS08 core, the upper two bits are always 0s. The instruction treats this value as an address in the 16-Kbyte address space.
- rel* — Any label or expression that refers to an address that is within –128 to +127 locations from the next address after the last byte of object code for the current instruction. The assembler will calculate the 8-bit signed offset and include it in the object code for this instruction.

### Address modes

- INH = Inherent (no operands)
- IMD = Immediate to Direct (in MOV instruction)
- IMM = Immediate
- DD = Direct to Direct (in MOV instruction)
- DIR = Direct
- SRT = Short
- TNY = Tiny
- EXT = Extended
- REL = 8-bit relative offset

Table 8-1. Instruction Set Summary (Sheet 1 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
ADC #opr8i ADC opr8a ADC ,X <sup>(1)</sup> ADC X	Add with Carry	$A \leftarrow (A) + (M) + (C)$ $A \leftarrow (A) + (X) + (C)$	↓	↓	IMM DIR IX DIR	A9 B9 B9 B9	ii dd 0E 0F	2 3 3 3
ADD #opr8i ADD opr8a ADD opr4a ADD ,X <sup>(1)</sup> ADD X	Add without Carry	$A \leftarrow (A) + (M)$	↓	↓	IMM DIR TNY IX DIR	AB BB 6x 6E 6F	ii dd	2 3 3 3 3
AND #opr8i AND opr8a AND ,X <sup>(1)</sup> AND X	Logical AND	$A \leftarrow (A) \& (M)$ $A \leftarrow (A) \& (X)$	↓	–	IMM DIR IX DIR	A4 B4 B4 B4	ii dd 0E 0F	2 3 3 3
ASLA <sup>(1)</sup>	Arithmetic Shift Left		↓	↓	INH	48		1
BCC rel	Branch if Carry Bit Clear	$PC \leftarrow (PC) + \$0002 + rel$ , if (C) = 0	–	–	REL	34	rr	3
BCLR n,opr8a	Clear Bit n in Memory	$M_n \leftarrow 0$	–	–	DIR (b0)	11	dd	5
BCLR n,D[X]					DIR (b1)	13	dd	5
					DIR (b2)	15	dd	5
					DIR (b3)	17	dd	5
					DIR (b4)	19	dd	5
					DIR (b5)	1B	dd	5
					DIR (b6)	1D	dd	5
					DIR (b7)	1F	dd	5
					IX (b0)	11	0E	5
					IX (b1)	13	0E	5
					IX (b2)	15	0E	5
					IX (b3)	17	0E	5
	IX (b4)	19	0E	5				
IX (b5)	1B	0E	5					
IX (b6)	1D	0E	5					
IX (b7)	1F	0E	5					
BCLR n,X					DIR (b0)	11	0F	5
					DIR (b1)	13	0F	5
					DIR (b2)	15	0F	5
					DIR (b3)	17	0F	5
					DIR (b4)	19	0F	5
					DIR (b5)	1B	0F	5
					DIR (b6)	1D	0F	5
					DIR (b7)	1F	0F	5
BCS rel	Branch if Carry Bit Set (Same as BLO)	$PC \leftarrow (PC) + \$0002 + rel$ , if (C) = 1	–	–	REL	35	rr	3
BEQ rel	Branch if Equal	$PC \leftarrow (PC) + \$0002 + rel$ , if (Z) = 1	–	–	REL	37	rr	3
BGND	Background	Enter Background Debug Mode	–	–	INH	BF		5+
BHS rel <sup>(1)</sup>	Branch if Higher or Same (Same as BCC)	$PC \leftarrow (PC) + \$0002 + rel$ , if (C) = 0	–	–	REL	34	rr	3
BLO rel <sup>(1)</sup>	Branch if Lower (Same as BCS)	$PC \leftarrow (PC) + \$0002 + rel$ , if (C) = 1	–	–	REL	35	rr	3
BNE rel	Branch if Not Equal	$PC \leftarrow (PC) + \$0002 + rel$ , if (Z) = 0	–	–	REL	36	rr	3
BRA rel	Branch Always	$PC \leftarrow (PC) + \$0002 + rel$	–	–	REL	30	rr	3
BRN rel <sup>(1)</sup>	Branch Never	$PC \leftarrow (PC) + \$0002$	–	–	REL	30	00	3

1. This is a pseudo instruction supported by the normal RS08 instruction set.

2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-1. Instruction Set Summary (Sheet 2 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear	$PC \leftarrow (PC) + \$0003 + rel, \text{ if } (Mn) = 0$	-	↓	DIR (b0)	01	dd rr	5
DIR (b1)					03	dd rr	5	
DIR (b2)					05	dd rr	5	
DIR (b3)					07	dd rr	5	
DIR (b4)					09	dd rr	5	
DIR (b5)					0B	dd rr	5	
DIR (b6)					0D	dd rr	5	
DIR (b7)					0F	dd rr	5	
IX (b0)					01	0E rr	5	
IX (b1)					03	0E rr	5	
IX (b2)					05	0E rr	5	
IX (b3)					07	0E rr	5	
IX (b4)					09	0E rr	5	
IX (b5)					0B	0E rr	5	
IX (b6)					0D	0E rr	5	
IX (b7)					0F	0E rr	5	
DIR (b0)					01	0F rr	5	
DIR (b1)					03	0F rr	5	
DIR (b2)					05	0F rr	5	
DIR (b3)					07	0F rr	5	
DIR (b4)					09	0F rr	5	
DIR (b5)	0B	0F rr	5					
DIR (b6)	0D	0F rr	5					
DIR (b7)	0F	0F rr	5					
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set	$PC \leftarrow (PC) + \$0003 + rel, \text{ if } (Mn) = 1$	-	↓	DIR (b0)	00	dd rr	5
DIR (b1)					02	dd rr	5	
DIR (b2)					04	dd rr	5	
DIR (b3)					06	dd rr	5	
DIR (b4)					08	dd rr	5	
DIR (b5)					0A	dd rr	5	
DIR (b6)					0C	dd rr	5	
DIR (b7)					0E	dd rr	5	
IX (b0)					00	0E rr	5	
IX (b1)					02	0E rr	5	
IX (b2)					04	0E rr	5	
IX (b3)					06	0E rr	5	
IX (b4)					08	0E rr	5	
IX (b5)					0A	0E rr	5	
IX (b6)					0C	0E rr	5	
IX (b7)					0E	0E rr	5	
DIR (b0)					00	0F rr	5	
DIR (b1)					02	0F rr	5	
DIR (b2)					04	0F rr	5	
DIR (b3)					06	0F rr	5	
DIR (b4)					08	0F rr	5	
DIR (b5)	0A	0F rr	5					
DIR (b6)	0C	0F rr	5					
DIR (b7)	0E	0F rr	5					

1. This is a pseudo instruction supported by the normal RS08 instruction set.
2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

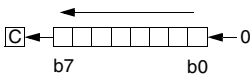
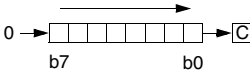
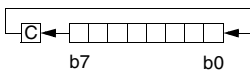
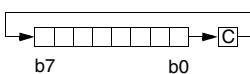
Table 8-1. Instruction Set Summary (Sheet 3 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory	$M_n \leftarrow 1$	-	-	DIR (b0)	10	dd	5
BSET <i>n,D[X]</i>					DIR (b1)	12	dd	5
					DIR (b2)	14	dd	5
					DIR (b3)	16	dd	5
					DIR (b4)	18	dd	5
					DIR (b5)	1A	dd	5
					DIR (b6)	1C	dd	5
					DIR (b7)	1E	dd	5
BSET <i>n,X</i>					IX (b0)	10	0E	5
					IX (b1)	12	0E	5
					IX (b2)	14	0E	5
					IX (b3)	16	0E	5
					IX (b4)	18	0E	5
					IX (b5)	1A	0E	5
					IX (b6)	1C	0E	5
	IX (b7)	1E	0E	5				
	DIR (b0)	10	0F	5				
	DIR (b1)	12	0F	5				
	DIR (b2)	14	0F	5				
	DIR (b3)	16	0F	5				
	DIR (b4)	18	0F	5				
	DIR (b5)	1A	0F	5				
	DIR (b6)	1C	0F	5				
	DIR (b7)	1E	0F	5				
BSR <i>rel</i>	Branch Subroutine	$PC \leftarrow (PC) + 2$ Push PC to shadow PC $PC \leftarrow (PC) + rel$	-	-	REL	AD	rr	3
CBEQA # <i>opr8i,rel</i> CBEQ <i>opr8a,rel</i> CBEQ <i>X,rel</i> <sup>(1),(2)</sup> CBEQ <i>X,rel</i> <sup>(1)</sup>	Compare and Branch if Equal	$PC \leftarrow (PC) + \$0003 + rel$ , if (A) - (M) = \$00 $PC \leftarrow (PC) + \$0003 + rel$ , if (A) - (M) = \$00 $PC \leftarrow (PC) + \$0003 + rel$ , if (A) - (X) = \$00	-	-	IMM DIR IX DIR	41 31 31 31	ii rr dd rr 0E rr 0F rr	4 5 5 5
CLC	Clear Carry Bit	$C \leftarrow 0$	-	0	INH	38		1
CLR <i>opr8a</i> CLR <i>opr5a</i> CLR <i>X</i> <sup>(1)</sup> CLRA CLR $X$ <sup>(1)</sup>	Clear	$M \leftarrow \$00$  $A \leftarrow \$00$ $X \leftarrow \$00$	1	-	DIR SRT IX INH INH	3F 8x / 9x 8E 4F 8F	dd	3 2 2 1 2
CMP # <i>opr8i</i> CMP <i>opr8a</i> CMP <i>X</i> <sup>(1)</sup> CMP $X$ <sup>(1)</sup>	Compare Accumulator with Memory	(A) - (M)  (A) - (X)	↓	↓	IMM DIR IX INH	A1 B1 B1 B1	ii dd 0E 0F	2 3 3 3
COMA	Complement (One's Complement)	$A \leftarrow (\bar{A})$	↓	1	INH	43		1
DBNZ <i>opr8a,rel</i> DBNZ <i>X,rel</i> <sup>(1)</sup> DBNZA <i>rel</i> DBNZX <i>rel</i> <sup>(1)</sup>	Decrement and Branch if Not Zero	$A \leftarrow (A) - \$01$ or $M \leftarrow (M) - \$01$ $PC \leftarrow (PC) + \$0003 + rel$ if (result) $\neq 0$ for DBNZ direct $PC \leftarrow (PC) + \$0002 + rel$ if (result) $\neq 0$ for DBNZA $X \leftarrow (X) - \$01$ $PC \leftarrow (PC) + \$0003 + rel$ if (result) $\neq 0$	-	-	DIR IX INH INH	3B 3B 4B 3B	dd rr 0E rr rr 0F rr	7 7 4 7
DEC <i>opr8a</i> DEC <i>opr4a</i> DEC <i>X</i> <sup>(1)</sup> DECA DEC $X$	Decrement	$M \leftarrow (M) - \$01$  $A \leftarrow (A) - \$01$ $X \leftarrow (X) - \$01$	↓	-	DIR TNY IX INH DIR	3A 5x 5E 4A 5F	dd	5 4 4 1 4
EOR # <i>opr8i</i> EOR <i>opr8a</i> EOR <i>X</i> <sup>(1)</sup> EOR $X$	Exclusive OR Memory with Accumulator	$A \leftarrow (A \oplus M)$  $A \leftarrow (A \oplus X)$	↓	-	IMM DIR IX DIR	A8 B8 B8 B8	ii dd 0E 0F	2 3 3 3

1. This is a pseudo instruction supported by the normal RS08 instruction set.

2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-1. Instruction Set Summary (Sheet 4 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
INC <i>opr8a</i> INC <i>opr4a</i> INC ,X <sup>(1)</sup> INCA INCX <sup>(1)</sup>	Increment	$M \leftarrow (M) + \$01$  $A \leftarrow (A) + \$01$ $X \leftarrow (X) + \$01$	↓	–	DIR TNY IX INH INH	3C 2x 2E 4C 2F	dd	5 4 4 1 4
JMP <i>opr16a</i>	Jump	PC ← Effective Address	–	–	EXT	BC	hh ll	4
JSR <i>opr16a</i>	Jump to Subroutine	PC ← (PC) + 3 Push PC to shadow PC PC ← Effective Address	–	–	EXT	BD	hh ll	4
LDA # <i>opr8i</i> LDA <i>opr8a</i> LDA <i>opr5a</i> LDA ,X <sup>(1)</sup>	Load Accumulator from Memory	$A \leftarrow (M)$	↓	–	IMM DIR SRT IX	A6 B6 Cx/Dx CE	ii dd	2 3 3 3
LDX # <i>opr8i</i> <sup>(1)</sup> LDX <i>opr8a</i> <sup>(1)</sup> LDX ,X <sup>(1)</sup>	Load Index Register from Memory	$\$0F \leftarrow (M)$	↓	–	IMD DIR IX	3E 4E 4E	ii 0F dd 0F 0E 0E	4 5 5
LSLA	Logical Shift Left		↓	↑	INH	48		1
LSRA	Logical Shift Right		↓	↑	INH	44		1
MOV <i>opr8a,opr8a</i> MOV # <i>opr8i,opr8a</i> MOV D[X], <i>opr8a</i> MOV <i>opr8a</i> ,D[X] MOV # <i>opr8i</i> ,D[X]	Move	$(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$	↓	–	DD IMD IX/DIR DIR/IX IMM/IX	4E 3E 4E 4E 3E	dd dd ii dd 0E dd dd 0E ii 0E	5 4 5 5 4
NOP	No Operation	None	–	–	INH	AC		1
ORA # <i>opr8i</i> ORA <i>opr8a</i> ORA ,X <sup>(1)</sup> ORA X	Inclusive OR Accumulator and Memory	$A \leftarrow (A) \mid (M)$ $A \leftarrow (A) \mid (X)$	↓	–	IMM DIR IX DIR	AA BA BA BA	ii dd 0E 0F	2 3 3 3
ROLA	Rotate Left through Carry		↓	↑	INH	49		1
RORA	Rotate Right through Carry		↓	↑	INH	46		1
RTS	Return from Subroutine	Pull PC from shadow PC	–	–	INH	BE		3
SBC # <i>opr8i</i> SBC <i>opr8a</i> SBC ,X <sup>(1)</sup> SBC X	Subtract with Carry	$A \leftarrow (A) - (M) - (C)$  $A \leftarrow (A) - (X) - (C)$	↓	↑	IMM DIR IX DIR	A2 B2 B2 B2	ii dd 0E 0F	2 3 3 3
SEC	Set Carry Bit	$C \leftarrow 1$	–	1	INH	39		1
SHA	Swap Shadow PC High with A	$A \leftrightarrow \text{SPCH}$	–	–	INH	45		1
SLA	Swap Shadow PC Low with A	$A \leftrightarrow \text{SPCL}$	–	–	INH	42		1
STA <i>opr8a</i> STA <i>opr5a</i> STA ,X <sup>(1)</sup> STA X	Store Accumulator in Memory	$M \leftarrow (A)$	↓	–	DIR SRT IX SRT	B7 Ex / Fx EE EF	dd	3 2 2 2

1. This is a pseudo instruction supported by the normal RS08 instruction set.

2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-1. Instruction Set Summary (Sheet 5 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
STX <i>opr8a</i> <sup>(1)</sup>	Store Index Register in Memory	$M \leftarrow (X)$	↓	–	DIR	4E	0F dd	5
STOP	Put MCU into stop mode		–	–	INH	AE		2+
SUB # <i>opr8i</i> SUB <i>opr8a</i> SUB <i>opr4a</i> SUB ,X <sup>(1)</sup> SUB X	Subtract	$A \leftarrow (A) - (M)$ $A \leftarrow (A) - (X)$	↓	↓	IMM DIR TNY IX DIR	A0 B0 7x 7E 7F	ii dd	2 3 3 3 3
TAX <sup>(1)</sup>	Transfer A to X	$X \leftarrow (A)$	↓	–	INH	EF		2
TST <i>opr8a</i> <sup>(1)</sup> TSTA <sup>(1)</sup> TST ,X <sup>(1)</sup> TSTX <sup>(1)</sup>	Test for Zero	(M) – \$00 (A) – \$00 (X) – \$00	↓	–	DD INH IX INH	4E AA 4E 4E	dd dd 00 0E 0E 0F 0F	5 2 5 5
TXA <sup>(1)</sup>	Transfer X to A	$A \leftarrow (X)$	↓	–	INH	CF		3
WAIT	Put MCU into WAIT mode		–	–	INH	AF		2+

1. This is a pseudo instruction supported by the normal RS08 instruction set.
2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-2. Opcode Map

	DIR	DIR	TNY	DIR/REL	INH	TNY	TNY	TNY	SRT	SRT	IMM/INH	DIR/EXT	SRT	SRT	SRT	SRT
HIGH	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOW	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRSET0 3 DIR	BSET0 2 DIR	INC 4 TNY	BRA 3 REL		DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	SUB 2 IMM	SUB 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
1	BRCLR0 3 DIR	BCLR0 2 DIR	INC 4 TNY	CBEQ 5 DIR	CBEQA 4 IMM	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	CMP 2 IMM	CMP 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
2	BRSET1 3 DIR	BSET1 2 DIR	INC 4 TNY		SLA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	SBC 2 IMM	SBC 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
3	BRCLR1 3 DIR	BCLR1 2 DIR	INC 4 TNY		COMA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT			LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
4	BRSET2 3 DIR	BSET2 2 DIR	INC 4 TNY	BCC 3 REL	LSRA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	AND 2 IMM	AND 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
5	BRCLR2 3 DIR	BCLR2 2 DIR	INC 4 TNY	BCS 3 REL	SHA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT			LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
6	BRSET3 3 DIR	BSET3 2 DIR	INC 4 TNY	BNE 3 REL	RORA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	LDA 2 IMM	LDA 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
7	BRCLR3 3 DIR	BCLR3 2 DIR	INC 4 TNY	BEQ 3 REL		DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT		STA 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
8	BRSET4 3 DIR	BSET4 2 DIR	INC 4 TNY	CLC 1 INH	LSLA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	EOR 2 IMM	EOR 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
9	BRCLR4 3 DIR	BCLR4 2 DIR	INC 4 TNY	SEC 1 INH	ROLA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	ADC 2 IMM	ADC 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
A	BRSET5 3 DIR	BSET5 2 DIR	INC 4 TNY	DEC 5 DIR	DECA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	ORA 2 IMM	ORA 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
B	BRCLR5 3 DIR	BCLR5 2 DIR	INC 4 TNY	DBNZ 6 DIR	DBNZA 4 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	ADD 2 IMM	ADD 3 DIR	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
C	BRSET6 3 DIR	BSET6 2 DIR	INC 4 TNY	INC 5 DIR	INCA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	NOP 1 INH	JMP 4 EXT	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
D	BRCLR6 3 DIR	BCLR6 2 DIR	INC 4 TNY			DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	BSR 3 REL	JSR 4 EXT	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
E	BRSET7 3 DIR	BSET7 2 DIR	INC 4 TNY	MOV 4 IMD	MOV 5 DD	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	STOP 2+ INH	RTS 3 INH	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT
F	BRCLR7 3 DIR	BCLR7 2 DIR	INC 4 TNY	CLR 3 DIR	CLRA 1 INH	DEC 4 TNY	ADD 3 TNY	SUB 3 TNY	CLR 2 SRT	CLR 2 SRT	WAIT 2+ INH	BGND 5+ INH	LDA 3 SRT	LDA 3 SRT	STA 2 SRT	STA 2 SRT

INH Inherent  
IMM Immediate  
DIR Direct  
EXT Extended  
DD Direct-Direct

REL Relative  
SRT Short  
TNY Tiny

IMD Immediate-Direct

Gray box is decoded as illegal instruction

High Byte of Opcode in Hexadecimal B

Low Byte of Opcode in Hexadecimal 0 SUB<sup>3</sup> DIR<sup>2</sup> RS08 Cycles  
Opcode Mnemonic  
Number of Bytes /  
Addressing Mode



---

## Chapter 9

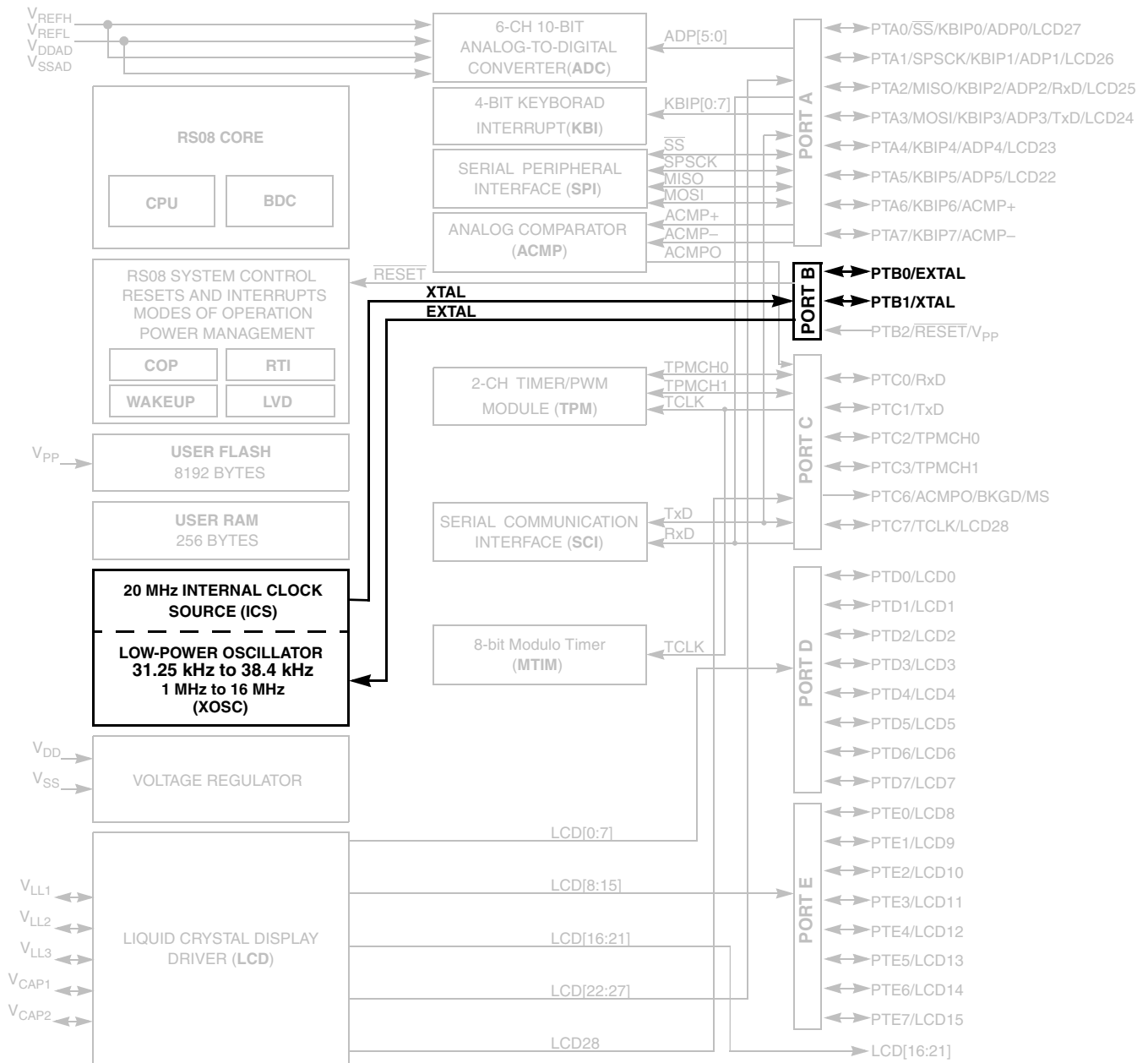
# Internal Clock Source (RS08ICSV1)

### 9.1 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 this FLL clock or the internal reference clock as a source for the MCU system clock, ICSOUT.

Whichever clock source is chosen, ICSOUT is passed through a bus clock divider (BDIV), which allows to derive a lower final output clock frequency. ICSOUT is twice the bus frequency.

[Figure 9-1](#) shows the MC9RS08LA8 series block diagram highlighting the ICS block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 9-1. MC9RS08LA8 Series Block Diagram Highlighting ICS Block and Pins**

## 9.1.1 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
- Internal or external reference clocks 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 clocks 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
- 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

## 9.1.2 Modes of Operation

There are seven modes of operation for the ICS: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

### 9.1.2.1 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.

### 9.1.2.2 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.

### 9.1.2.3 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.

### 9.1.2.4 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.

### 9.1.2.5 FLL Bypassed External (FBE)

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.

### 9.1.2.6 FLL Bypassed External Low Power (FBELP)

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.

### 9.1.2.7 Stop (STOP)

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

## 9.1.3 Block Diagram

Figure 9-2 is the ICS block diagram.

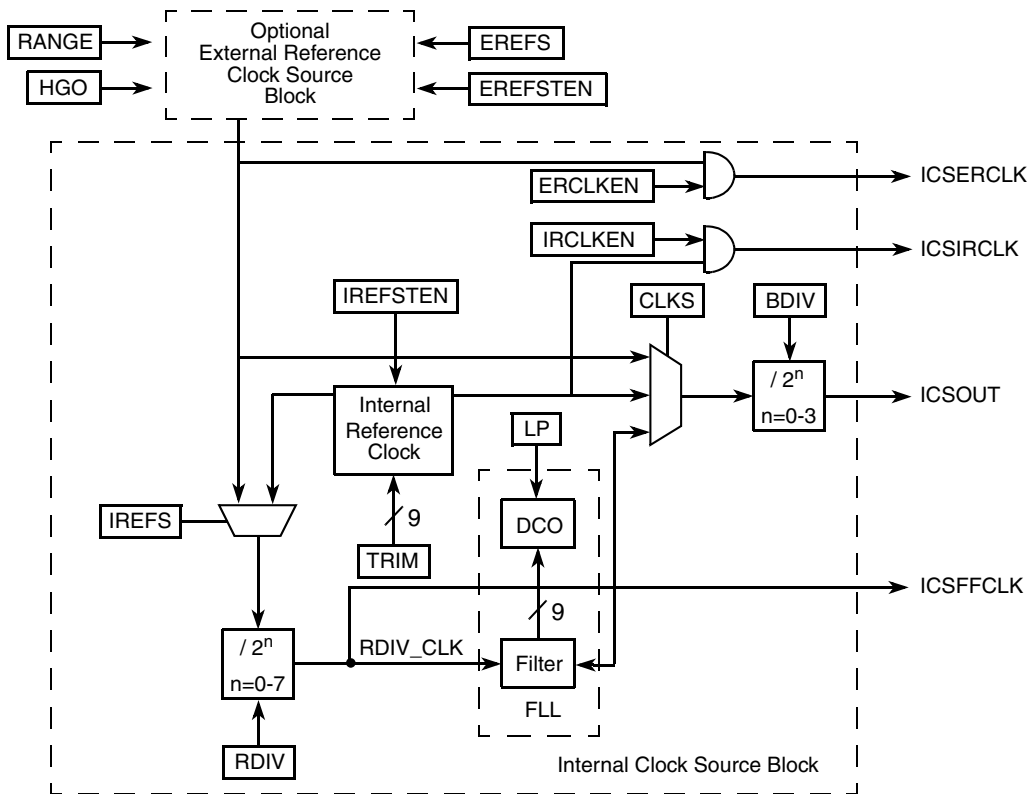


Figure 9-2. Internal Clock Source (ICS) Block Diagram

## 9.2 External Signal Description

There are no ICS signals that connect off chip.

## 9.3 Register Definition

Figure 9-1 is a summary of ICS registers.

Table 9-1. ICS Register Summary

Name		7	6	5	4	3	2	1	0
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
	W								
ICSC2	R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
	W								
ICSTRM	R	TRIM							
	W								
ICSSC	R	0	0	0	0	CLKST		OSCINIT	FTRIM
	W								

### 9.3.1 ICS Control Register 1 (ICSC1)

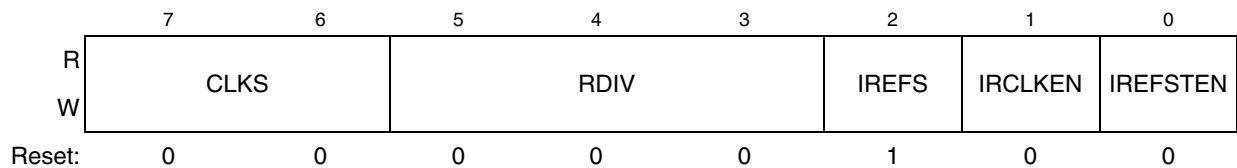


Figure 9-3. ICS Control Register 1 (ICSC1)

Table 9-2. ICS Control Register 1 Field Descriptions

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

## 9.3.2 ICS Control Register 2 (ICSC2)



Figure 9-4. ICS Control Register 2 (ICSC2)

Table 9-3. ICS Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	<b>Bus Frequency Divider</b> — 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	<b>Frequency Range Select</b> — 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	<b>High Gain Oscillator Select</b> — 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	<b>Low Power Select</b> — The LP bit controls whether the FLL is disabled in FLL bypassed modes. 1 FLL is disabled in bypass modes 0 FLL is not disabled in bypass mode
2 EREFS	<b>External Reference Select</b> — The EREFS bit selects the source for the external reference clock. 1 Oscillator requested 0 External Clock Source requested
1 ERCLKEN	<b>External Reference Enable</b> — The ERCLKEN bit enables the external reference clock for use as IC SERCLK. 1 IC SERCLK active 0 IC SERCLK inactive
0 EREFSTEN	<b>External Reference Stop Enable</b> — 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

### 9.3.3 ICS Trim Register (ICSTRM)

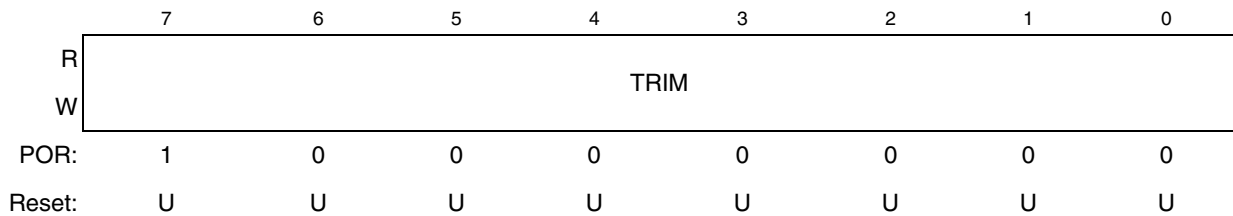


Figure 9-5. ICS Trim Register (ICSTRM)

Table 9-4. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	<p><b>ICS Trim Setting</b> — 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.</p> <p>An additional fine trim bit is available in ICSSC as the FTRIM bit.</p>

### 9.3.4 ICS Status and Control (ICSSC)

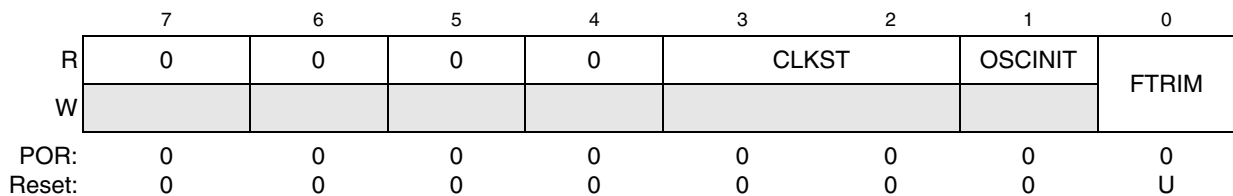


Figure 9-6. ICS Status and Control Register (ICSSC)

Table 9-5. ICS Status and Control Register Field Descriptions

Field	Description
7:2	Reserved, must be cleared.
1	<p><b>OSC Initialization</b> — 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.</p>
0	<p><b>ICS Fine Trim</b> — 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.</p>
3-2 CLKST	<p><b>Clock Mode Status</b> — 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.</p> <p>00 Output of FLL is selected.            01 FLL Bypassed, Internal reference clock is selected.            10 FLL Bypassed, External reference clock is selected.            11 Reserved.</p>



## 9.4 Functional Description

### 9.4.1 Operational Modes

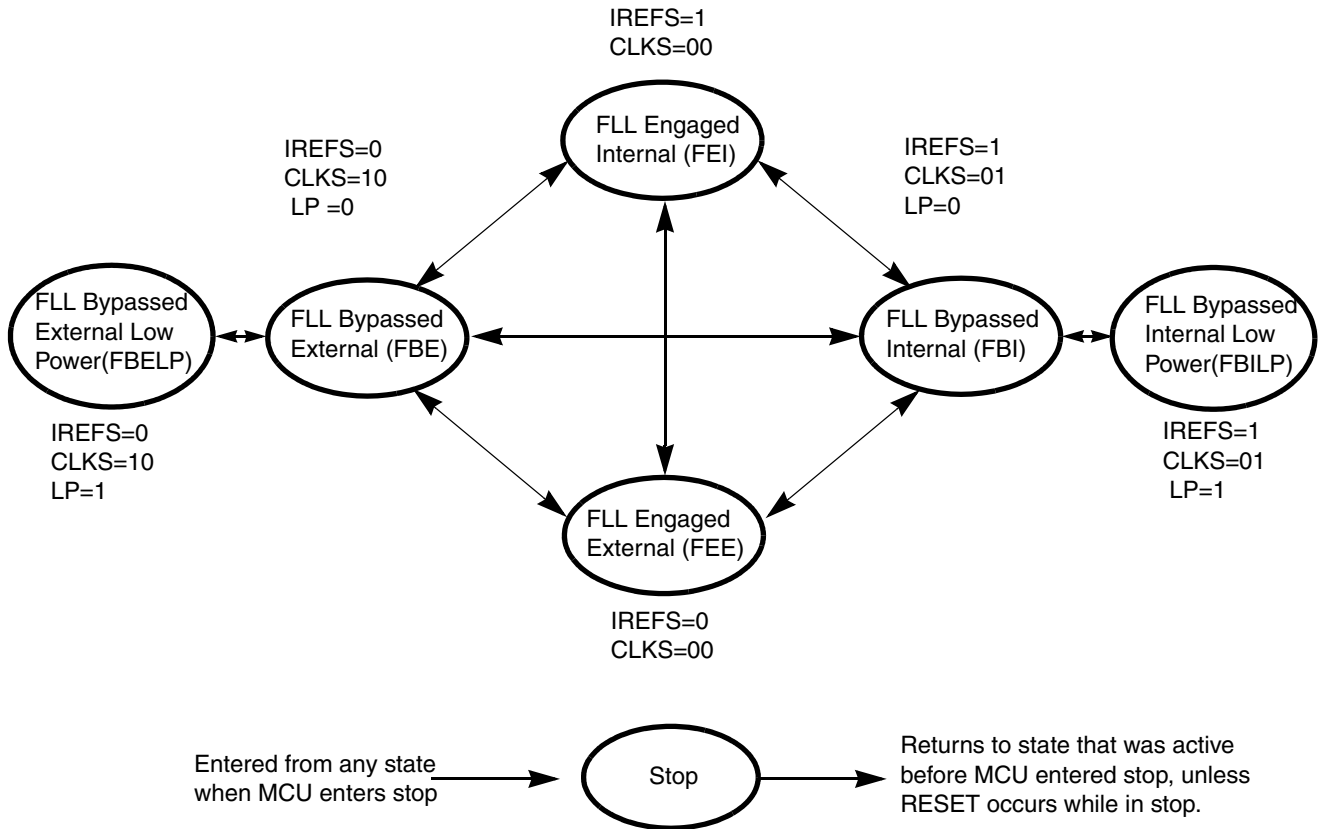


Figure 9-7. Clock Switching Modes

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

#### 9.4.1.1 FLL Engaged Internal (FEI)

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

- CLKS bits are written to 00
- IREFS bit is written to 1
- 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. and the internal reference clock is enabled.

### 9.4.1.2 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, and the external reference clock is enabled.

### 9.4.1.3 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.
- 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, and the internal reference clock is enabled.

### 9.4.1.4 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.
- 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, and the internal reference clock is enabled.

### 9.4.1.5 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.
- 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, and the external reference clock is enabled.

### 9.4.1.6 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.
- 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 external reference clock is enabled.

### 9.4.1.7 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, all ICS clock signals are static 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

## 9.4.2 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.

## 9.4.3 Bus Frequency Divider

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

## 9.4.4 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.

### 9.4.5 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 ICSIRCLK 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.

### 9.4.6 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 ICSECLK will keep running during stop mode in order to provide a fast recovery upon exiting stop.

## 9.4.7 Fixed Frequency Clock

The ICS presents 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:

- 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 10

## Liquid Crystal Display Module (S08LCDV2)

### 10.1 Introduction

In MC9RS08LA8 series, the liquid crystal display module (LCD) controls the 29 LCD pins to generate the waveforms necessary to drive a liquid crystal display. These LCD pins can be configured for 4 × 25 or 8 × 21 based on software configuration.

#### 10.1.1 LCD Clock Sources

The LCD module on MC9RS08LA8 can be clocked from ICSFFCLK or the external clock (ICSERCLK).

#### 10.1.2 LCD Modes of Operation

The LCD module can be configured to operate in stop modes by setting the LCDSTP bit. In stop mode, only ICSECLK is available for LCD module.

#### 10.1.3 LCD Power Supply Configurations

In MC9RS08LA8, the LCD power supply can be connected to  $V_{DD}$  internally. When connected to  $V_{LL2}$ , the internal charge pump will generate  $V_{LL1}$  and  $V_{LL3}$ . When connected to  $V_{LL3}$ , the internal charge pump will generate  $V_{LL1}$  and  $V_{LL2}$ . Both connections need corresponding software configurations.

The LCD power supply can be connected to an external power source through  $V_{LL3}$  pin to generate  $V_{LL1}$ ,  $V_{LL2}$ , and  $V_{LL3}$ .

Table 10-1 Lists typical LCD power supply configurations.

**Table 10-1. Typical LCD Power Supply Configurations**

$V_{DD}$	$V_{LL3}$	LCD Glass	Charge Pump	LCD Power Supply Configuration	$V_{SUPPLY}[1:0]$	CPSEL
3 V	3 V	3 V	Disabled	Use internal resistor network, Charge Pump disabled, $V_{LL1}$ and $V_{LL2}$ are generated by the resistor net work. $V_{LL3}$ is connected to $V_{DD}$ externally.	11	0
3 V	3 V	3 V	Disabled	Use internal resistor network, Charge Pump disabled, $V_{LL1}$ and $V_{LL2}$ are generated by the resistor net work. $V_{LL3}$ is connected to $V_{DD}$ internally.	01	0

Table 10-1. Typical LCD Power Supply Configurations (continued)

V <sub>DD</sub>	V <sub>LL3</sub>	LCD Glass	Charge Pump	LCD Power Supply Configuration	V <sub>SUPPLY</sub> [1:0]	CPSEL
3 V	3 V	3 V	Enabled	Use external power supply to V <sub>LL3</sub> (V <sub>LL3</sub> is connected to V <sub>DD</sub> externally). V <sub>LL1</sub> and V <sub>LL2</sub> are generated by the charge pump.	11	1
3 V	3 V	3 V	Enabled	Use internal V <sub>DD</sub> power supply to V <sub>LL3</sub> (V <sub>LL3</sub> is connected to V <sub>DD</sub> internally). V <sub>LL1</sub> and V <sub>LL2</sub> are generated by the charge pump.	01	1
3.3 V	5 V	5 V	Enabled	Use internal V <sub>DD</sub> power supply to V <sub>LL2</sub> . V <sub>LL1</sub> and V <sub>LL3</sub> are generated by charge pump.	00	1
5 V	5 V	5 V	Disabled	Use internal resistor network, the charge pump disabled, V <sub>LL1</sub> and V <sub>LL2</sub> are generated by the internal resistor network, V <sub>LL3</sub> is connected to V <sub>DD</sub> externally.	11	0
5 V	5 V	5 V	Disabled	Use internal resistor network, the charge pump disabled, V <sub>LL1</sub> and V <sub>LL2</sub> are generated by the internal resistor network, V <sub>LL3</sub> is connected to V <sub>DD</sub> internally.	01	0
5 V	5 V	5 V	Enabled	Use external power supply to V <sub>LL3</sub> (V <sub>LL3</sub> is connected to V <sub>DD</sub> externally) V <sub>LL1</sub> and V <sub>LL2</sub> are generated by the charge pump.	11	1
5 V	5 V	5 V	Enabled	Use internal V <sub>DD</sub> power supply to V <sub>LL3</sub> (V <sub>LL3</sub> is connected to V <sub>DD</sub> internally) V <sub>LL1</sub> and V <sub>LL2</sub> are generated by the charge pump.	01	1

**NOTE**

The V<sub>LCD</sub>, V<sub>I<sub>REG</sub></sub> and the voltage divider block are not available in MC9RS08LA8 series. The corresponding registers and bits are not visible to users.

**10.1.4 Open Drain Mode**

Some LCD driver pins are muxed with GPIO and SPI. These pins work in open drain mode when V<sub>LL3</sub> is not connected to V<sub>DD</sub> externally. An external pullup must be used when these pins are used as output pins and V<sub>LL3</sub> is not connected to V<sub>DD</sub>.

**10.1.5 Read Only Register Operations**

In MC9RS08LA8 MCU, all LCD associated registers are grouped into two parts, global control registers and pin control registers. The global control registers are read write ones including LCDCR0, LCDCR1, LCDSUPPLY, LCDBCTL, and LCDSR. The pin control registers are read only including LCDPENx, LCDBPENx, and LCDWFX. A user-specified buffer is recommend to record the contents that are written to LCDWFX and can be read back when necessary.

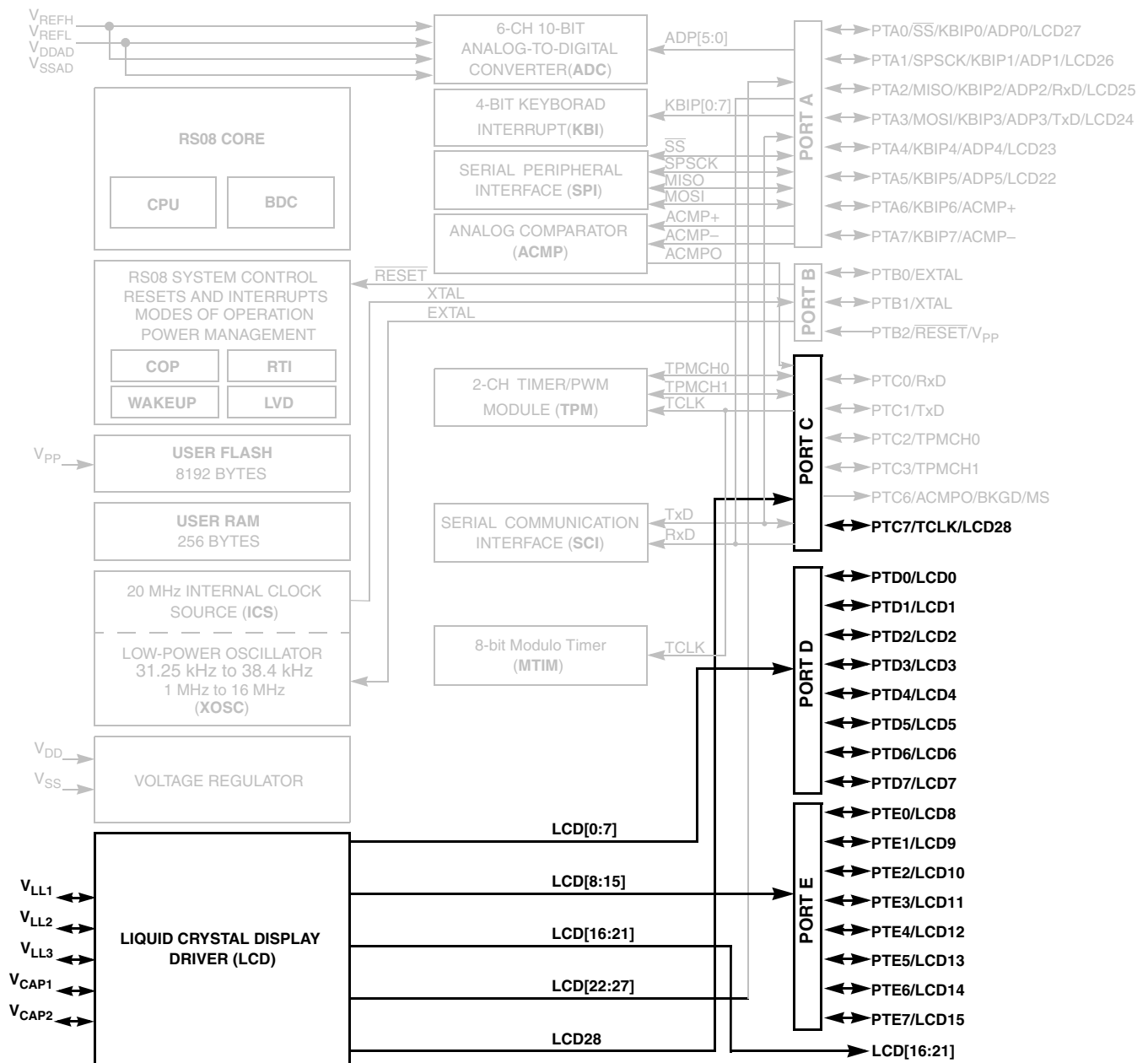


Table 10-2 Summaries all pins can be used in open drain mode.

**Table 10-2. Open Drain LCD Driver Pins**

LCD pins	Port	Alt
LCD[7:0]	PTD[7:0]	
LCD[15:8]	PTE[7:0]	
LCD22	PTA5	
LCD23	PTA4	
LCD24	PTA3	MOSI
LCD25	PTA2	MISO
LCD26	PTA1	SPSCK
LCD27	PTA0	$\overline{SS}$
LCD28	PTC7	

Figure 10-1 shows the MC9RS08LA8 block diagram highlighting the LCD block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 10-1. MC9RS08LA8 Series Block Diagram Highlighting LCD Block and Pins**

## 10.1.6 Features

The LCD module driver features include:

- LCD waveforms functional in LP<sub>Run</sub>, LP<sub>Wait</sub>, wait, stop low-power modes
- 29 LCD (LCD[28:0]) pins with selectable frontplane/backplane configuration
  - Generate up to 28 frontplane signals
  - Generate up to 8 backplanes signals
- Programmable LCD frame frequency
- Programmable blink modes and frequency
  - All segments blank during blink period
  - Alternate display for each LCD segment in x4 or less mode
  - Blink operation in low-power modes
- Programmable LCD power supply switch, making it an ideal solution for battery-powered and board-level applications
  - Charge pump requires only four external capacitors
  - Internal LCD power using V<sub>DD</sub> (2.7 to 5.5 V)
  - External V<sub>LL3</sub> power supply option (3V or 5V)
- Integrated charge pump for generating LCD bias voltages
  - Hardware configurable to drive 3-V or 5-V LCD panels
  - On-chip generation of bias voltages
- Waveform storage registers LCDWF
- 4  $\mu$ A operation in Stop mode- Design target
- GPIO Functionality on LCD[28:0]
  - Configurable to reference the digital input to V<sub>DD</sub> or V<sub>LL3</sub>
  - Open-drain operation with pull-up to V<sub>DD</sub>
  - Full complimentary drive when referenced to V<sub>LL3</sub> (V<sub>LL3</sub> must be connected to a low impedance power source)
- Backplane reassignment to assist in vertical scrolling on dot-matrix displays
- Software configurable LCD frame frequency interrupt
- Internal ADC channels are connected to V<sub>LL1</sub> to monitor their magnitudes. This feature allows software to adjust the contrast.

## 10.1.7 Modes of Operation

The LCD module supports the following operation modes:

Mode	Operation
Stop	<p>Depending on the state of the LCDSTP bit, the LCD module can operate an LCD panel in stop mode. If LCDSTP = 1, LCD module clock generation is turned off and the LCD module enters a power conservation state and is disabled. If LCDSTP = 0, the LCD module can operate an LCD panel in stop, and the LCD module continues displaying the current LCD panel contents based on the LCD operation prior to the stop event.</p> <p>If the LCD is enabled in stop, the selected LCD clock source, IC SERCLK or the Alternate Clock, must be enabled to operate in stop.</p> <p>In stop mode the LCD frame interrupt can cause the MCU to exit stop.</p>
Wait	<p>Depending on the configuration, the LCD module can operate an LCD panel in wait mode. If LCDWAI = 1, the LCD module clock generation is turned off and the LCD module enters a power-conservation state and is disabled. If LCDWAI = 0, the LCD module can operate an LCD panel in wait, and the LCD module continues displaying the current LCD panel contents base on the LCDWF registers.</p> <p>In wait mode, the LCD frame interrupt can cause the MCU to exit wait.</p>

Stop provides the lowest power consumption state where the LCD module is functional. To operate the LCD in stop mode, use an external reference clock.

## 10.1.8 Block Diagram

Figure 10-2 is a block diagram of the LCD module.

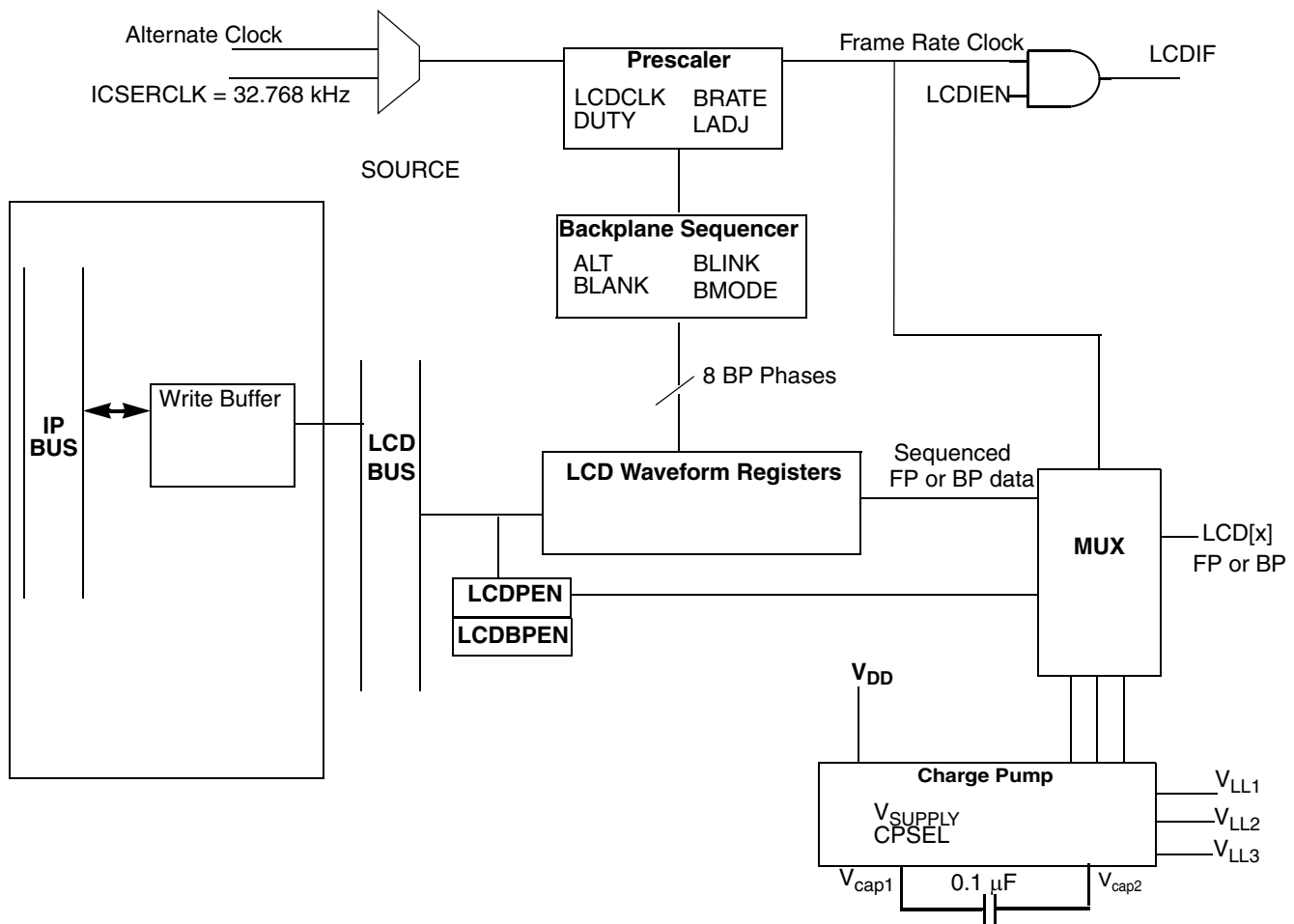


Figure 10-2. LCD Driver Block Diagram

## 10.2 External Signal Description

The LCD module has several external pins dedicated to power supply and LCD frontplane/backplane signaling. The LCD module can be configured to support eight backplane signals. The table below itemizes all the LCD external pins. See the [Pins and Connections](#) chapter for device-specific pin configurations.

Table 10-4. Signal Properties

Name	Port	Function	Reset State
29 LCD frontplane/backplane	LCD[28:0]	Switchable frontplane/backplane driver that connects directly to the display LCD[28:0] can operate as GPIO pins	High impedance

Table 10-4. Signal Properties (continued)

Name	Port	Function	Reset State
LCD bias voltages	$V_{LL1}$ , $V_{LL2}$ , $V_{LL3}$	LCD bias voltages	—
LCD charge pump capacitance	$V_{cap1}$ , $V_{cap2}$	Charge pump capacitor pins	—

### 10.2.1 LCD[29:0]

When LCD functionality is enabled by the PEN[28:0] bits in the LCDPEN registers, the corresponding LCD[28:0] pin will generate a frontplane or backplane waveform depending on the configuration of the backplane-enable bit field (BPEN[28:0]).

### 10.2.2 $V_{LL1}$ , $V_{LL2}$ , $V_{LL3}$

$V_{LL1}$ ,  $V_{LL2}$ , and  $V_{LL3}$  are bias voltages for the LCD module driver waveforms which can be internally generated using the internal charge pump (when enabled). The charge pump can also be configured to accept  $V_{LL3}$  as an input and generate  $V_{LL1}$  and  $V_{LL2}$  for 3 V glass operation.  $V_{LL3}$  should never be set to a voltage other than  $V_{DD}$ . Refer to VSUPPLY[1:0] bits explanation.

### 10.2.3 $V_{cap1}$ , $V_{cap2}$

The charge pump capacitor is used to transfer charge from the input supply to the regulated output. Use a ceramic capacitor.

## 10.3 Register Definition

This section consists of register descriptions. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

### 10.3.1 LCD Control Register 0 (LCDC0)

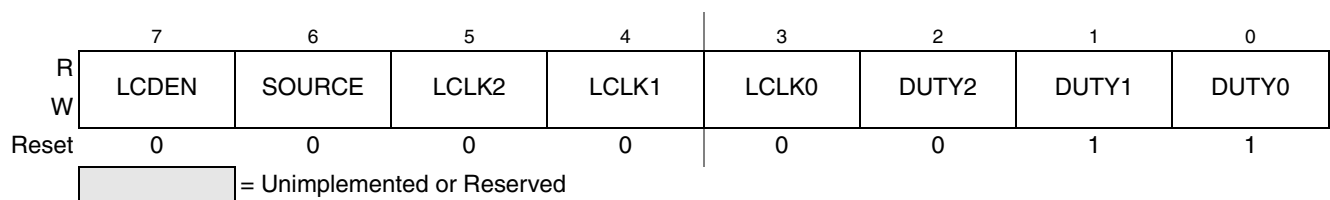


Figure 10-3. LCD Control Register 0 (LCDC0)

Read: anytime

Write: LCDEN anytime. Do not change SOURCE LCLK OR DUTY while LCDEN = 1.

Table 10-5. LCDC0 Field Descriptions

Field	Description
7 LCDEN	<p><b>LCD Driver Enable</b> — LCDEN starts LCD-module-waveform generator.</p> <p>0 All frontplane and backplane pins are disabled. The LCD module system is also disabled, and all LCD waveform generation clocks are stopped. <math>V_{LL3}</math> is connected to <math>V_{DD}</math> internally</p> <p>1 LCD module driver system is enabled and frontplane and backplane waveforms are generated. All LCD pins enabled using the LCD pin enable register (LCDPEN[x]) will output an LCD module driver waveform. The backplane pins will output an LCD module driver backplane waveform based on the settings of DUTY[2:0]. Chargepump or resistor bias is enabled.</p>
6 SOURCE	<p><b>LCD Clock Source Select</b> — The LCD module has two possible clock sources. This bit is used to select which clock source is the basis for LCDCLK.</p> <p>0 Selects the ICSECLK (external clock reference) as the LCD clock source.</p> <p>1 Selects the alternate clock as the LCD clock source.</p>
5:3 LCLK[2:0]	<p><b>LCD Clock Prescaler</b> — Used as a clock divider to generate the LCD module frame frequency as shown in Equation 10-1. LCD-module-duty-cycle configuration is used to determine the LCD module frame frequency. LCD module frame frequency calculations are provided in Table 10-14./p.139.</p> <p style="text-align: right;"><b>Eqn. 10-1</b></p> $\text{LCD Module Frame Frequency} = \frac{\text{LCDCLK}}{((\text{DUTY}+1) \times 8 \times (4 + \text{LCLK}[2:0]) \times Y)}$ <p style="text-align: right;">where <math>30 &lt; \text{LCDCLK} &lt; 39.063 \text{ kHz}</math> where <math>Y = 2,2,3,3,4,5,8,16</math> chosen by module duty cycle configuration</p>
2:0 DUTY[2:0]	<p><b>LCD Duty Select</b> — DUTY[2:0] bits select the duty cycle of the LCD module driver.</p> <p>000 Use 1 BP (1/1 duty cycle).</p> <p>001 Use 2 BP (1/2 duty cycle).</p> <p>010 Use 3 BP (1/3 duty cycle).</p> <p>011 Use 4 BP (1/4 duty cycle). (Default)</p> <p>100 Use 5 BP (1/5 duty cycle).</p> <p>101 Use 6 BP (1/6 duty cycle).</p> <p>110 Use 7 BP (1/7 duty cycle).</p> <p>111 Use 8 BP (1/8 duty cycle).</p>

### 10.3.2 LCD Control Register 1 (LCDC1)

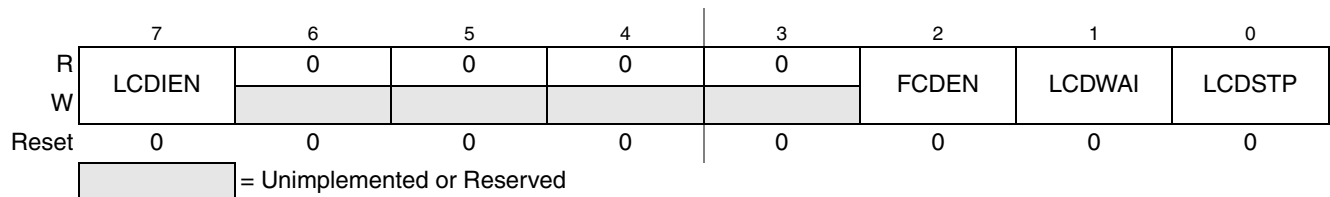


Figure 10-4. LCD Control Register 1 (LCDC1)

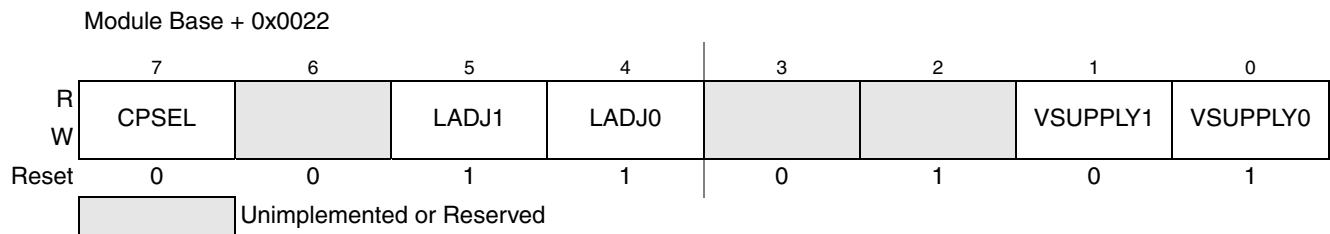
Read: anytime

Write: anytime

Table 10-6. LCDC1 Field Descriptions

Field	Description
7 LCDIEN	<b>LCD Module Frame Frequency Interrupt Enable</b> — Enables an LCD interrupt event that coincides with the LCD module frame frequency. 0 No interrupt request is generated by this event. 1 The start of the LCD module frame causes an LCD module frame frequency interrupt request.
2 FCDEN	<b>Full Complementary Drive Enable</b> — This bit allows GPIO that are shared with LCD pins to operate as full complementary if the other conditions necessary have been met. The other conditions are: VSUPPLY = 11 and RVEN = 0. 0 GPIO shared with LCD operate as open drain outputs, input levels and internal pullup resistors are referenced to $V_{DD}$ . 1 If VSUPPLY = 11 and RVEN = 0, GPIO shared with LCD operate as full complementary outputs. Input levels and internal pullup resistors are referenced to $V_{LL3}$ .
1 LCDWAI	<b>LCD Module Driver and Charge Pump Stop While in Wait Mode</b> 0 Allows the LCD driver and charge pump to continue running during wait mode. 1 Disables the LCD driver and charge pump when MCU goes into wait mode.
0 LCDSTP	<b>LCD Module Driver and Charge Pump Stop While in Stop2 or Stop3 Mode</b> 0 Allows LCD module driver and charge pump to continue running during stop. 1 Disables LCD module driver and charge pump when MCU goes into stop.

### 10.3.3 LCD Voltage Supply Register (LCDSUPPLY)



Read: anytime

Write: anytime.

For proper operation, do not modify VSUPPLY[1:0] while the LCDEN bit is asserted. VSUPPLY[1:0] must also be configured according to the external hardware power supply configuration.



Table 10-7. LCDSUPPLY Field Descriptions

Field	Description
7 CPSEL	<b>Charge Pump or Resistor Bias Select</b> — Selects LCD module charge pump or a resistor network to supply the LCD voltages $V_{LL1}$ , $V_{LL2}$ , and $V_{LL3}$ . See <a href="#">Figure 10-15</a> for more detail. 0 LCD charge pump is disabled. Resistor network selected (The internal 1/3-bias is forced.) 1 LCD charge pump is selected. Resistor network disabled (The internal 1/3-bias is forced.)
5:4 LADJ[1:0]	<b>LCD Module Load Adjust</b> — The LCD load adjust bits are used to configure the LCD module to handle different LCD glass capacitance.  For CPSEL = 1 Adjust the clock source for the charge pump. Higher loads require higher charge pump clock rates. 00 - Fastest clock source for charge pump (LCD glass capacitance 8000pf or lower) 01 - Intermediate clock source for charge pump (LCD glass capacitance 6000pf or lower) 10 - Intermediate clock source for charge pump (LCD glass capacitance 4000pf or lower) 11 - Slowest clock source for charge pump (LCD glass capacitance 2000pf or lower)  For CPSEL = 0 Adjust the resistor bias network for different LCD glass capacitance 00 - Low Load (LCD glass capacitance 2000pf or lower) 01 - Low Load (LCD glass capacitance 2000pf or lower) 10 - High Load (LCD glass capacitance 8000pf or lower) 11 - High Load (LCD glass capacitance 8000pf or lower)
1:0 VSUPPLY[1:0]	<b>Voltage Supply Control</b> — Configures whether the LCD module power supply is external or internal. Avoid modifying this bit field while the LCD module is enabled (e.g., LCDEN = 1). See <a href="#">Figure 10-15</a> for more detail. 00 Drive $V_{LL2}$ internally from $V_{DD}$ 01 Drive $V_{LL3}$ internally from $V_{DD}$ 10 Reserved 11 Drive $V_{LL3}$ externally

### 10.3.4 LCD Blink Control Register (LCDBCTL)

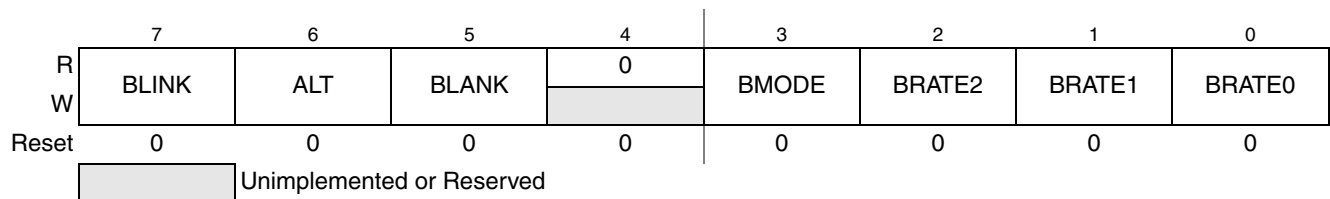


Figure 10-6. LCD Blink Control Register (LCDBCTL)

Read: anytime

Write: anytime

Table 10-8. LCDBCTL Field Descriptions

Field	Description
7 BLINK	<b>Blink Command</b> — Starts or stops LCD module blinking 0 Disables blinking 1 Starts blinking at blinking frequency specified by LCD blink rate calculation (see <a href="#">Equation 10-2</a> )
6 ALT	<b>Alternate Display Mode</b> — For four backplanes or less the LCD backplane sequencer changes to output an alternate display. ALT bit is ignored if Duty is 5 or greater. 0 Normal Display 1 Alternate display mode
5 BLANK	<b>Blank Display Mode</b> — Asserting this bit clears all segments in the LCD display. 0 Normal or Alternate Display 1 Blank Display Mode
3 BMODE	<b>Blink Mode</b> — Selects the blink mode displayed during the blink period. See <a href="#">Table 10-8</a> for more information on how BMODE affects the LCD display. 0 Display blank during the blink period 1 Display alternate display during blink period (Ignored if duty is 5 or greater)
2:0 BRATE[2:0]	<b>Blink-Rate Configuration</b> — Selects frequency at which the LCD display blinks when the BLINK is asserted. <a href="#">Equation 10-2</a> shows how BRATE[2:0] bit field is used in the LCD blink-rate calculation.  <a href="#">Equation 10-2</a> provides an expression for the LCD module blink rate  $\text{LCD module blink rate} = \frac{\text{LCDCLK}}{2^{(12 + \text{BRATE}[2:0])}}$ <i>Eqn. 10-2</i>  LCD module blink rate calculations are provided in <a href="#">10.4.3.2/p.145</a> .

### 10.3.5 LCD Status Register (LCDS)

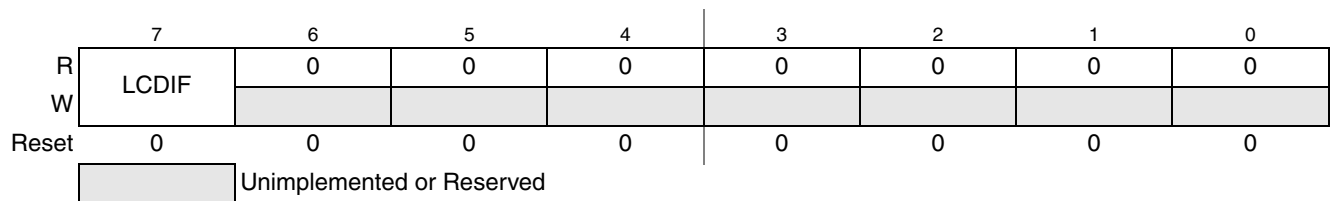


Figure 10-7. LCD Status Register (LCDS)

Read: anytime

Write: anytime

Table 10-9. LCDS Field Descriptions

Field	Description
7 LCDIF	<b>LCD Interrupt Flag</b> — LCDIF indicates an interrupt condition occurred. To clear the interrupt write a 1 to LCDIF. 0 interrupt condition has not occurred. 1 interrupt condition has occurred.

### 10.3.6 LCD Pin Enable Registers 0–3 (LCDPEN0–LCDPEN37)

When LCDEN = 1, these bits enable the corresponding LCD pin for LCD operation.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register. Initialize these registers before enabling the LCD module. Exiting Stop2 mode does not require reinitializing the LCDPEN registers.

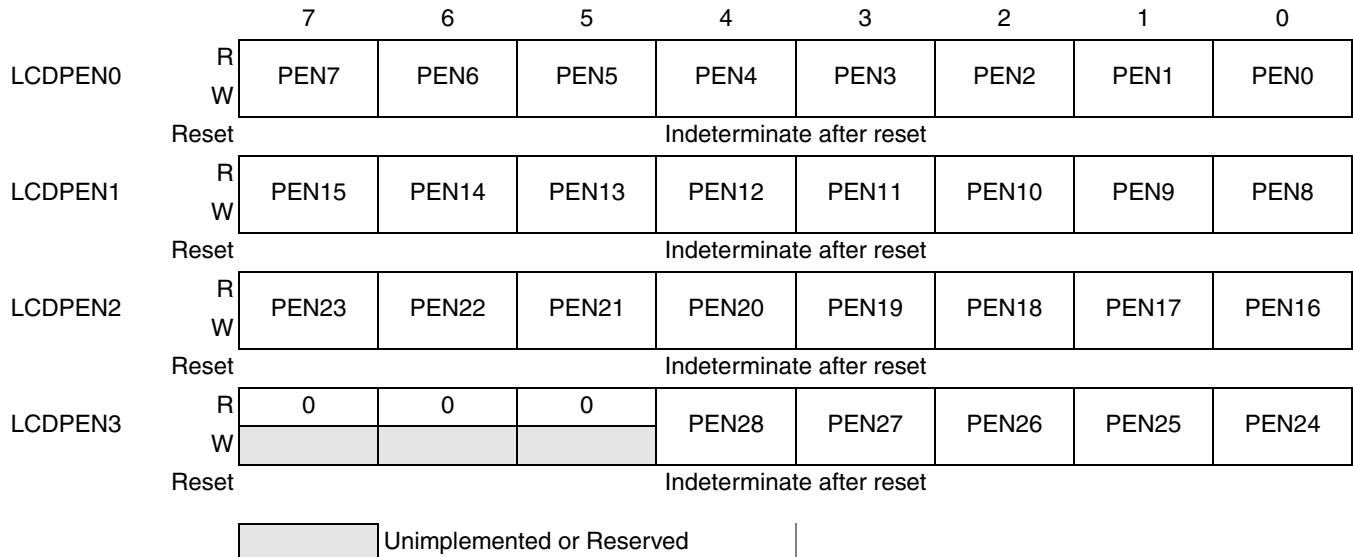


Figure 10-8. LCD Pin Enable Registers 0–3 (LCDPEN0–LCDPEN3)

Read: anytime

Write: anytime

Table 10-10. LCDPEN0–LCDPEN7 Field Descriptions

Field	Description
28:0 PEN[28:0]	<p><b>LCD Pin Enable</b> — The PEN[28:0] bit enables the LCD[28:0] pin for LCD operation. Each LCD[28:0] pin can be configured as a backplane or a frontplane based on the corresponding BPEN[n] bit in the Backplane Enable Register (LCDBPEN[3:0]). If LCDEN = 0, these bits have no effect on the state of the I/O pins. Set PEN[28:0] bits before LCDEN is set.</p> <p>0 LCD operation disabled on LCDn. 1 LCD operation enabled on LCDn.</p>

### 10.3.7 Backplane Enable Registers 0–3 (BPEN0–BPEN3)

When LCDPEN[n] = 1, these bits configure the corresponding LCD pin to operate as an LCD backplane or an LCD frontplane. Most applications set a maximum of eight of these bits. Initialize these registers before enabling the LCD module. Exiting Stop2 mode does not require reinitializing the LCDBPEN registers.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register.

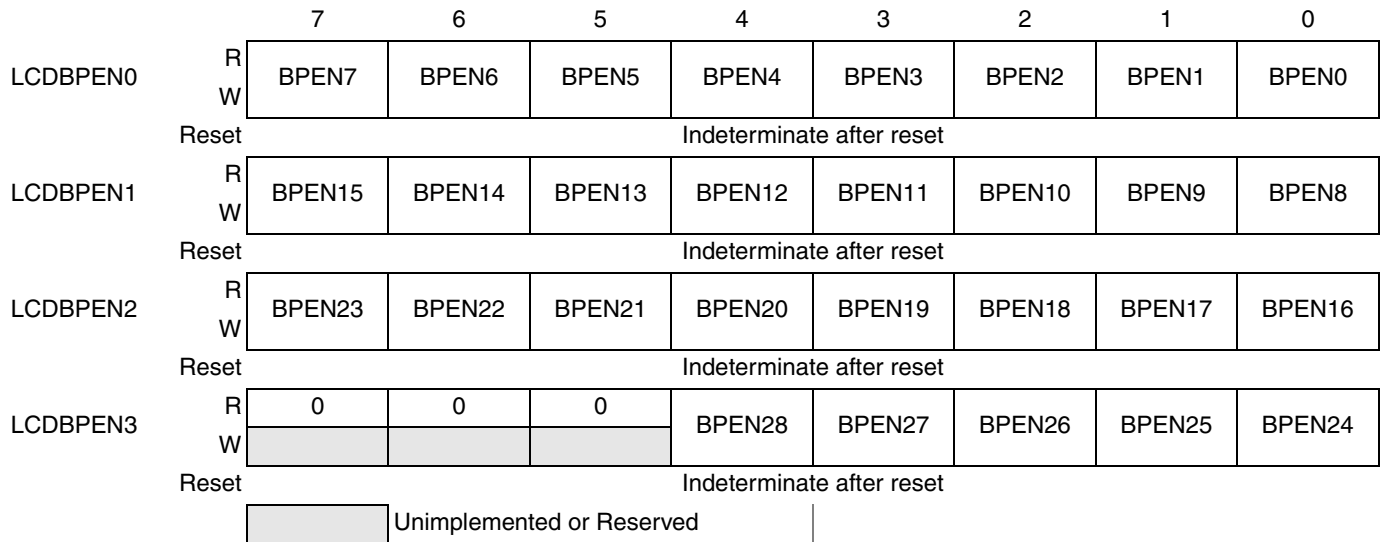


Figure 10-9. Backplane Enable Registers 0–3 (BPEN0–BPEN3)

Read: anytime

Write: anytime

Table 10-11. LCDBPEN0–LCDBPEN3 Field Descriptions

Field	Description
28:0 BPEN[28:0]	<p><b>Backplane Enable</b> — The BPEN[28:0] bit configures the LCD[28:0] pin to operate as an LCD backplane or LCD frontplane. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set BPEN[28:0] bits before LCDEN is set.</p> <p>0 Frontplane operation enabled on LCD[n]. 1 Backplane operation enabled on LCD[n].</p>

### 10.3.8 LCD Waveform Registers (LCDWF[28:0])

Each frontplane segment is associated with a backplane phase (A-H). For an LCD pin configured as a frontplane the LCDWF registers control the on/off state for frontplane segments.

For an LCD pin configured as a backplane the LCDWF registers controls the phase (A-H) in which the associated backplane pin is active.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register.

After reset, the LCDWF contents are indeterminate as indicated by [Figure 10-10](#). Exiting Stop2 mode does not require reinitializing the LCDWF registers.

		7	6	5	4	3	2	1	0
<b>LCDWF0</b>	R	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
	W	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
	Reset	Indeterminate after reset							
<b>LCDWF1</b>	R	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
	W	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
	Reset	Indeterminate after reset							
<b>LCDWF2</b>	R	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
	W	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
	Reset	Indeterminate after reset							
<b>LCDWF3</b>	R	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
	W	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
	Reset	Indeterminate after reset							
<b>LCDWF4</b>	R	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
	W	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
	Reset	Indeterminate after reset							
<b>LCDWF5</b>	R	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
	W	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
	Reset	Indeterminate after reset							
<b>LCDWF6</b>	R	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
	W	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
	Reset	Indeterminate after reset							
<b>LCDWF7</b>	R	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
	W	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
	Reset	Indeterminate after reset							
<b>LCDWF8</b>	R	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
	W	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
	Reset	Indeterminate after reset							
<b>LCDWF9</b>	R	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
	W	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
	Reset	Indeterminate after reset							
<b>LCDWF10</b>	R	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
	W	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
	Reset	Indeterminate after reset							
<b>LCDWF11</b>	R	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
	W	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
	Reset	Indeterminate after reset							
<b>LCDWF12</b>	R	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
	W	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
	Reset	Indeterminate after reset							

Figure 10-10. LCD Waveform Registers (LCDWF[28:0])

<b>LCDWF13</b>	R	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
	W								
	Reset	Indeterminate after reset							
<b>LCDWF14</b>	R	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
	W								
	Reset	Indeterminate after reset							
<b>LCDWF15</b>	R	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
	W								
	Reset	Indeterminate after reset							
<b>LCDWF16</b>	R	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
	W								
	Reset	Indeterminate after reset							
<b>LCDWF17</b>	R	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
	W								
	Reset	Indeterminate after reset							
<b>LCDWF18</b>	R	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
	W								
	Reset	Indeterminate after reset							
<b>LCDWF19</b>	R	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
	W								
	Reset	Indeterminate after reset							
<b>LCDWF20</b>	R	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
	W								
	Reset	Indeterminate after reset							
<b>LCDWF21</b>	R	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21
	W								
	Reset	Indeterminate after reset							
<b>LCDWF22</b>	R	BPHLCD22	BPGLCD22	BPFLCD22	BPELCD22	BPDLCD22	BPCLCD22	BPBLCD22	BPALCD22
	W								
	Reset	Indeterminate after reset							
<b>LCDWF23</b>	R	BPHLCD23	BPGLCD23	BPFLCD23	BPELCD23	BPDLCD23	BPCLCD23	BPBLCD23	BPALCD23
	W								
	Reset	Indeterminate after reset							
<b>LCDWF24</b>	R	BPHLCD24	BPGLCD24	BPFLCD24	BPELCD24	BPDLCD24	BPCLCD24	BPBLCD24	BPALCD24
	W								
	Reset	Indeterminate after reset							
<b>LCDWF25</b>	R	BPHLCD25	BPGLCD25	BPFLCD25	BPELCD25	BPDLCD25	BPCLCD25	BPBLCD25	BPALCD25
	W								
	Reset	Indeterminate after reset							
<b>LCDWF26</b>	R	BPHLCD26	BPGLCD26	BPFLCD26	BPELCD26	BPDLCD26	BPCLCD26	BPBLCD26	BPALCD26
	W								

**Figure 10-10. LCD Waveform Registers (LCDWF[28:0]) (continued)**

	Reset	Indeterminate after reset							
LCDWF27	R	BPHLCD27	BPGLCD27	BPFLCD27	BPELCD27	BPDLCD27	BPCLCD27	BPBLCD27	BPALCD27
	W								
	Reset	Indeterminate after reset							
LCDWF28	R	BPHLCD28	BPGLCD28	BPFLCD28	BPELCD28	BPDLCD28	BPCLCD28	BPBLCD28	BPALCD28
	W								
	Reset	Indeterminate after reset							

Figure 10-10. LCD Waveform Registers (LCDWF[28:0]) (continued)

Table 10-12. LCDWF Field Descriptions

Field	Description
BP[y]LCD[x]	<p><b>Segment-on-Frontplane Operation</b> — If the LCD[x] pin is enabled and configured to operate as a frontplane, the BP[y]LCD[x] bit in the LCDWF registers controls the on/off state for the LCD segment connected between LCD[x] and BP[y].BP[y] corresponds to an LCD[28:0] pin enabled and configured to operate as a backplane that is active in phase [y]. Asserting BP[y]LCD[x] displays (turns on) the LCD segment connected between LCD[x] and BP[y].</p> <p>0 LCD segment off 1 LCD segment on</p> <p><b>Segment-on-Backplane Operation</b> — If the LCD[x] pin is enabled and configured to operate as a backplane, the BP[y] LCD[x] bit in the LCDWF registers controls the phase (A-H) in which the LCD[x] pin is active.Backplane phase assignment is done using this method.</p> <p>0 LCD backplane inactive for phase[y] 1 LCD backplane active for phase[y].</p>

## 10.4 Functional Description

This section provides a complete functional description of the LCD block, detailing the operation of the design from the end-user perspective.

Before enabling the LCD module by asserting the LCDEN bit in the LCDC0 register, configure the LCD module based on the end application requirements. Out of reset, the LCD module is configured with default settings, but these settings are not optimal for every application. The LCD module provides several versatile configuration settings and options to support varied implementation requirements, including:

- Frame frequency
- Duty cycle (number of backplanes)
- Backplane assignment (which LCD[28:0] pins operate as backplanes)
- Frame frequency interrupt enable
- Blinking frequency and options
- Power-supply configurations

The LCD module also provides an LCD pin enable control. Setting the LCD pin enable bit (PEN[x] in the LCDPEN[y] register) for a particular LCD[y] pin enables the LCD module functionality of that pin once

the LCDEN bit is set. When the BPEN[x] bit in the LCDBPEN[y] is set, the associated pin operates as a backplane. The LCDWF registers can then activate (display) the corresponding LCD segments on an LCD panel.

The LCDWF registers control the on/off state for the segments controlled by the LCD pins defined as front planes and the active phase for the backplanes. Blank display modes do not use the data from the LCDWF registers. When using the LCDWF register for frontplane operation, writing a 0 turns the segment off.

For pins enabled as backplane, the phase of the backplane (A-H) is assigned by the LCDWF register for the corresponding backplane pin.

## 10.4.1 LCD Driver Description

The LCD module driver has 8 modes of operation:

- 1/1 duty (1 backplane) (Phase A), 1/3 bias (4 voltage levels)
- 1/2 duty (2 backplanes) (Phase A, B), 1/3 bias (4 voltage levels)
- 1/3 duty (3 backplanes) (Phase A, B, C), 1/3 bias (4 voltage levels)
- 1/4 duty (4 backplanes) (Phase A, B, C, D), 1/3 bias (4 voltage levels)
- 1/5 duty (5 backplanes) (Phase A, B, C, D, E), 1/3 bias (4 voltage levels)
- 1/6 duty (6 backplanes) (Phase A, B, C, D, E, F), 1/3 bias (4 voltage levels)
- 1/7 duty (7 backplanes) (Phase A, B, C, D, E, F, G), 1/3 bias (4 voltage levels)
- 1/8 duty (8 backplanes) (Phase A, B, C, D, E, F, G, H), 1/3 bias (4 voltage levels)

All modes are 1/3 bias. These modes of operation are described in more detail in the following sections.

### 10.4.1.1 LCD Duty Cycle

The denominator of the duty cycle indicates the number of LCD panel segments capable of being driven by each individual frontplane output driver. Depending on the duty cycle, the LCD waveform drive can be categorized as static or multiplexed.

In static-driving method, the LCD is driven with two square waveforms. The static-driving method is the most basic method to drive an LCD panel, but because each frontplane driver can drive only one LCD segment, static driving limits the LCD segments that can be driven with a given number of frontplane pins. In static mode, only one backplane is required.

In multiplexed mode, the LCD waveforms are multi-level and depend on the bias mode. Multiplex mode, depending on the number of backplanes, can drive multiple LCD segments with a single frontplane driver. This reduces the number of driver circuits and connections to LCD segments. For multiplex mode operation, at least two backplane drivers are needed. The LCD module is optimized for multiplex mode.

The duty cycle indicates the amount of time the LCD panel segment is energized during each LCD module frame cycle. The denominator of the duty cycle indicates the number of backplanes that are being used to drive an LCD panel.

The duty cycle is used by the backplane phase generator to set the phase outputs. The phase outputs A-H are driven according to the sequence shown below. The sequence is repeated at the LCD frame frequency.



The duty cycle is configured using the DUTY[2:0] bit field in the LCDC0 register, as shown in [Table 10-13](#).

**Table 10-13. LCD Module Duty Cycle Modes**

Duty	LCDC0 Register			Number of Backplanes	Phase Sequence
	DUTY2	DUTY1	DUTY0		
1/1	0	0	0	1	A
1/2	0	0	1	2	A B
1/3	0	1	0	3	A B C
1/4	0	1	1	4	A B C D
1/5	1	0	0	5	A B C D E
1/6	1	0	1	6	A B C D E F
1/7	1	1	0	7	A B C D E F G
1/8	1	1	1	8	A B C D E F G H

#### 10.4.1.2 LCD Bias

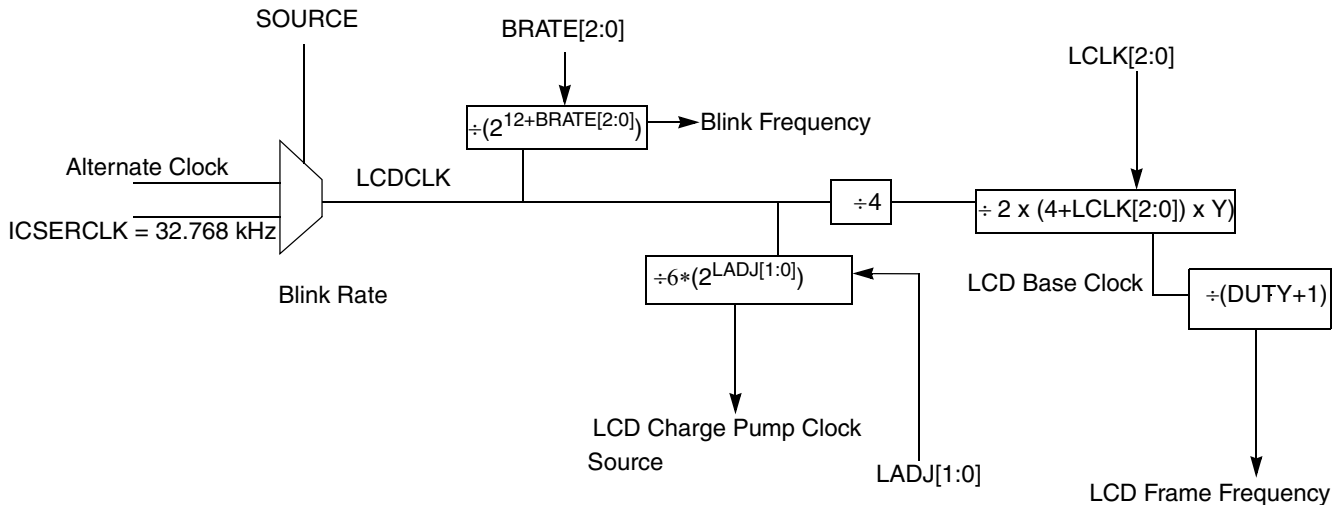
Because a single frontplane driver is configured to drive more and more individual LCD segments, 3 voltage levels are required to generate the appropriate waveforms to drive the segment. The LCD module is designed to operate using the 1/3 bias mode.

#### 10.4.1.3 LCD Module Base Clock and Frame Frequency

The LCD module is optimized to operate using a 32.768-kHz clock input. Two clock sources are available to the LCD module, which are selectable by configuring the SOURCE bit in the LCDC0 register. The two clock sources include:

- External crystal —ICSERCLK (SOURCE = 0)
- Alternate clock (SOURCE = 1)

[Figure 10-11](#) shows the LCD clock tree. The clock tree shows the two possible clock sources and the LCD frame frequency and blink frequency clock source. The LCD blink frequency is discussed in [Section 10.4.3.2, “Blink Frequency.”](#)



**Figure 10-11. LCD Clock Tree**

An external 32.768-kHz clock input is required to achieve lowest power consumption.

The value of LCDCLK is important because it is used to generate the LCD module frame frequency. [Equation 10-1](#) provides an expression for the LCD module frame frequency calculation.

The LCD module frame frequency is a function of the LCD module duty cycle as shown in [Equation 10-1](#). [Table 10-15](#) and [Table 10-14](#) show LCD module frame frequency calculations that consider several possible LCD module configurations of LCLK[2:0] and DUTY[2:0].

The LCD module frame frequency is defined as the number of times the LCD segments are energized per second. The LCD module frame frequency must be selected to prevent the LCD display from flickering (LCD module frame frequency is too low) or ghosting (LCD module frame frequency is too high). To avoid these issues, an LCD module frame frequency in the range of 28 to 58 Hz is required. LCD module frame frequencies less than 28 Hz or greater than 58 Hz are out of specification, and so are invalid. Selecting lower values for the LCD base and frame frequency results in lower current consumption for the LCD module.

The LCD module base clock frequency is the LCD module frame frequency multiplied by the number of backplane phases that are being generated. The number of backplane phases is selected using the DUTY[2:0] bits. The LCD module base clock is used by the backplane sequencer to generate the LCD waveform data for the enabled phases (A-H).

**Table 10-14. LCD Module Frame Frequency Calculations<sup>1</sup>**

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	16	8	5	4	3	3	2	2
LCLK[2:0]								
0	64	64	68.3	64	68.3	56.9	73.1	64
1	51.2	51.2	54.6	51.2	54.6	45.5	58.5	51.2
2	42.7	42.7	45.5	42.7	45.5	37.9	48.8	42.7
3	36.6	36.6	39	36.6	39	32.5	41.8	36.6
4	32	32	34.1	32	34.1	28.4	36.6	32
5	28.4	28.4	30.3	28.4	30.3	25.3	32.5	28.4
6	25.6	25.6	27.3	25.6	27.3	22.8	29.3	25.6
7	23.3	23.3	24.8	23.3	24.8	20.7	26.6	23.3

<sup>1</sup> LCD clock input ~ 32.768 kHz

Shaded table entries are out of specification and invalid.

**Table 10-15. LCD Module Frame Frequency Calculations<sup>1</sup>**

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	16	8	5	4	3	3	2	2
LCLK[2:0]								
0	76.3	76.3	81.4	76.3	81.4	67.8	87.2	76.3
1	61	61	65.1	61	65.1	54.3	69.8	61
2	50.9	50.9	54.3	50.9	54.3	45.2	58.1	50.9
3	43.6	43.6	46.5	43.6	46.5	38.8	49.8	43.6
4	38.1	38.1	40.7	38.1	40.7	33.9	43.6	38.1
5	33.9	33.9	36.2	33.9	36.2	30.1	38.8	33.9
6	30.5	30.5	32.6	30.5	32.6	27.1	34.9	30.5
7	27.7	27.7	29.6	27.7	29.6	24.7	31.7	27.7

<sup>1</sup> LCD clock input ~ 39.063 kHz

Shaded table entries are out of specification and invalid.

#### 10.4.1.4 LCD Waveform Examples

This section shows the timing examples of the LCD output waveforms for the several modes of operation. As shown in [Table 10-16](#), all examples use 1/3 bias mode.

**Table 10-16. Configurations for Example LCD Waveforms**

	Bias Mode	DUTY[2:0]	Duty Cycle
Example 1	1/3	001	1/2
Example 2		011	1/4
Example 3		111	1/8

### 10.4.1.4.1 1/2 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty=1/2:DUTY[2:0] = 001

LCD pin 0 (LCD[0])and LCD pin 1, LCD[1] enabled as backplanes:

BPEN0 =1 and BPEN1 =1 in the LCDBPEN0

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

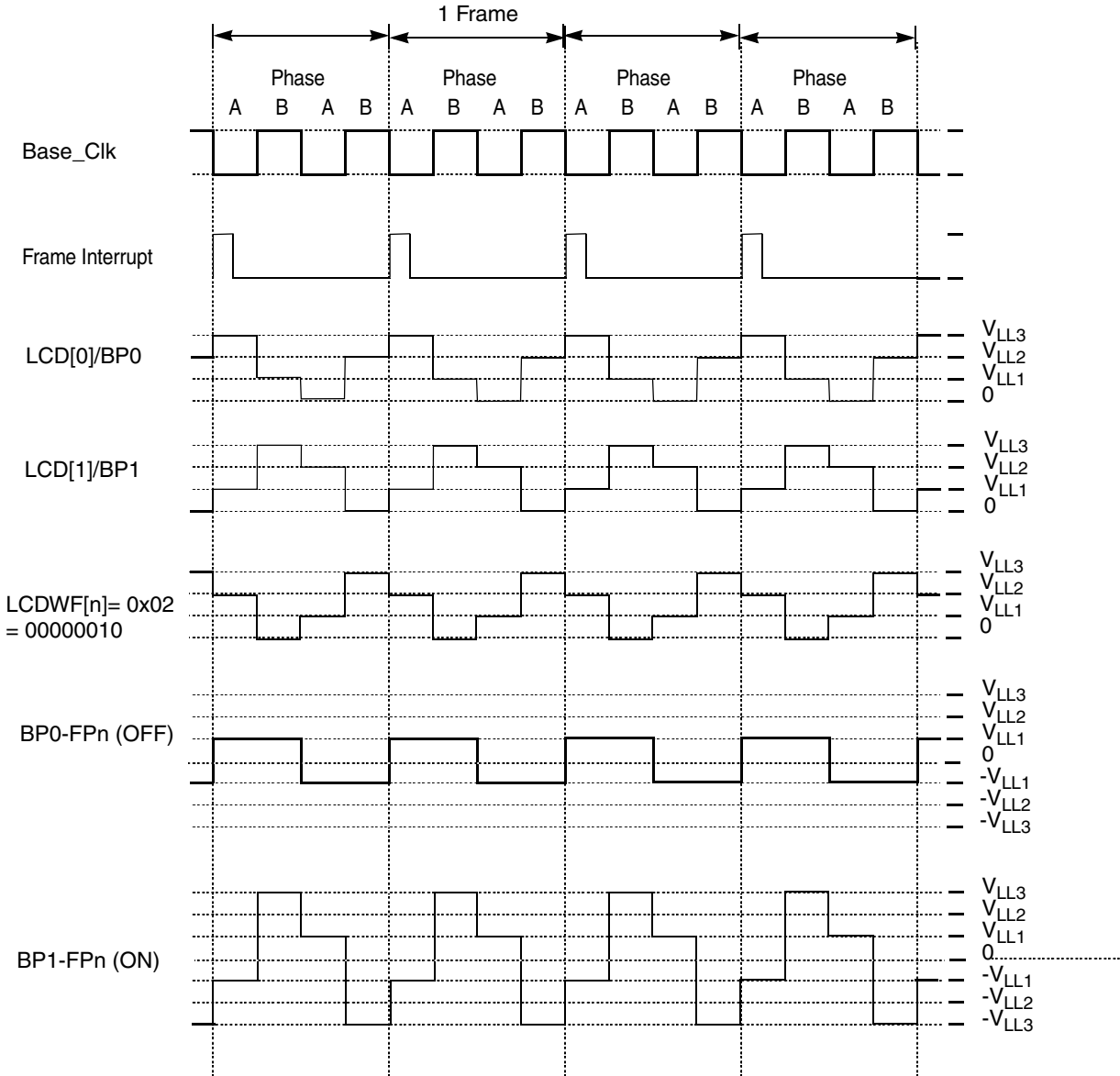


Figure 10-12. 1/2 Duty and 1/3 Bias (Low-Power Waveform)

### 10.4.1.4.2 1/4 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty = 1/4: DUTY[2:0] = 011

LCD pins 0 – 3 enabled as backplanes: LCDBPEN0 = 0x0F

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

LCD[2] assigned to Phase C: LCDWF2 = 0x04

LCD[3] assigned to Phase D: LCDWF3 = 0x08

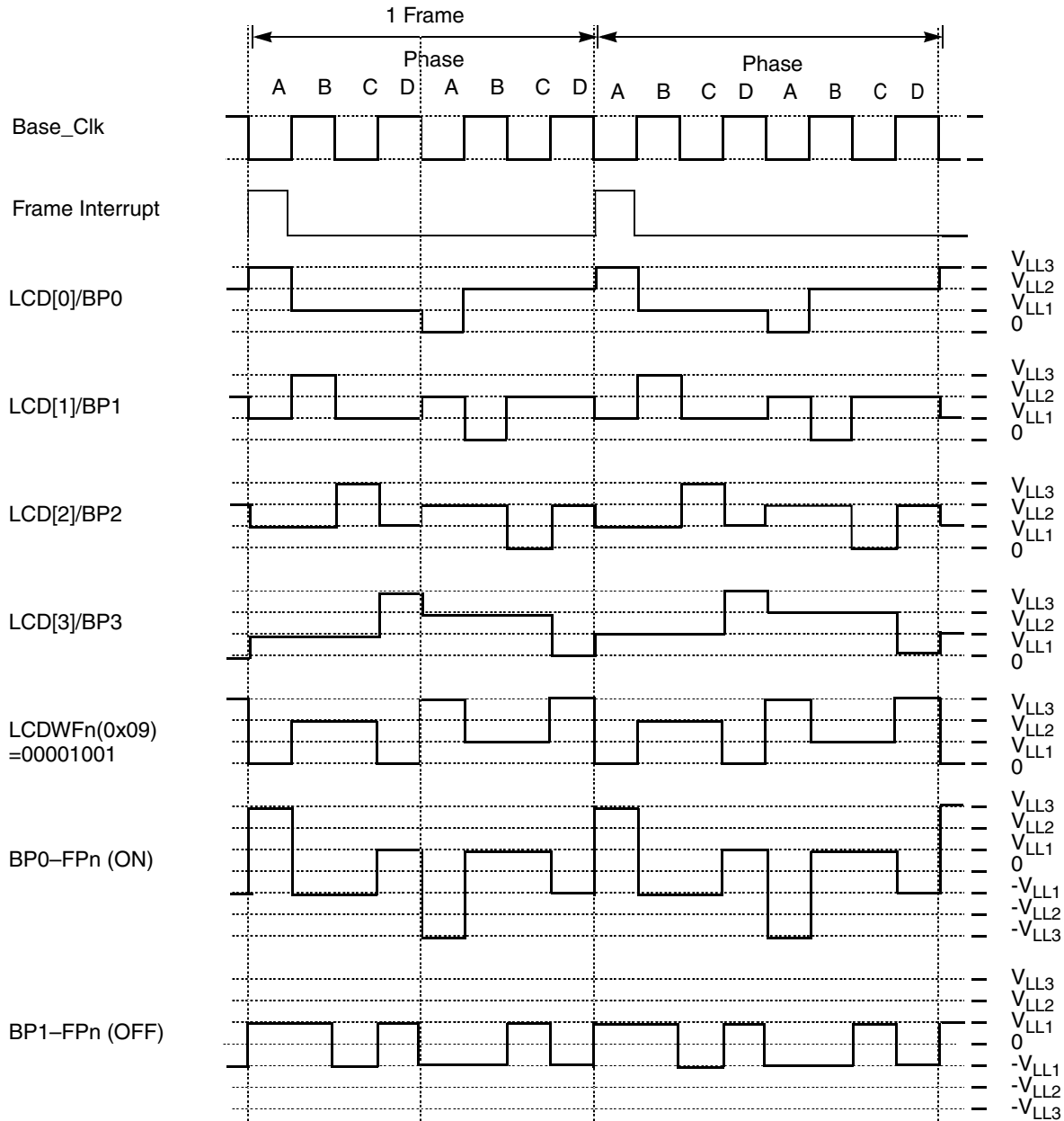


Figure 10-13. 1/4 Duty and 1/3 Bias (Low-Power Waveform)

### 10.4.1.4.3 1/8 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty = 1/8:DUTY[2:0] = 111

LCD pins 0 – 7 enabled as backplanes: LCDBPEN0 = 0xFF

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

LCD[2] assigned to Phase C: LCDWF2 = 0x04

LCD[3] assigned to Phase D: LCDWF3 = 0x08

LCD[4] assigned to Phase E: LCDWF4 = 0x10

LCD[5] assigned to Phase F: LCDWF5 = 0x20

LCD[6] assigned to Phase G: LCDWF6 = 0x40

LCD[7] assigned to Phase H: LCDWF7 = 0x80

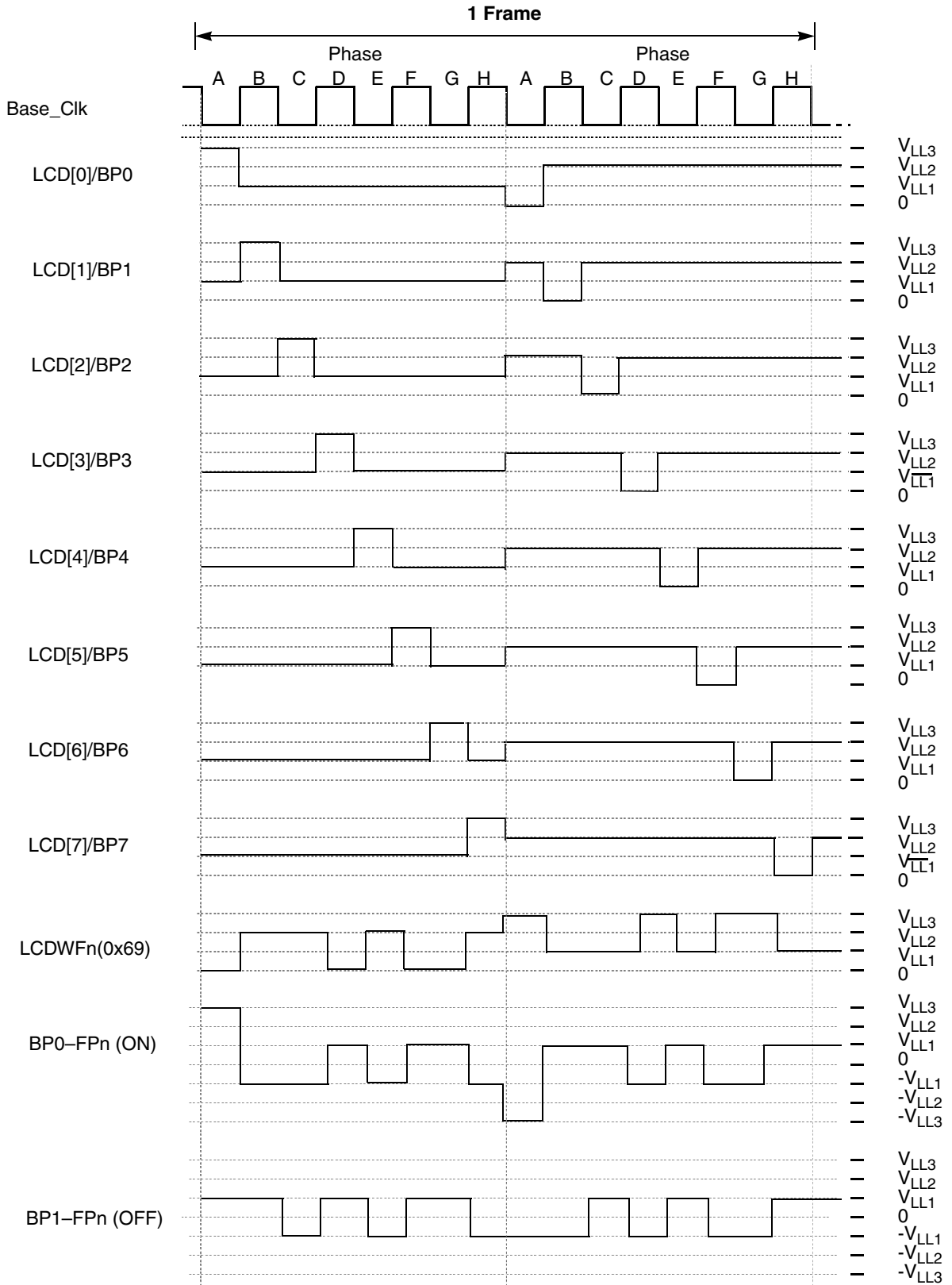


Figure 10-14. 1/8 Duty and 1/3 Bias (Low-power Waveform)

## 10.4.2 LCDWF Registers

For a segment on the LCD panel to be displayed, data must be written to the LCDWF registers. For LCD pins enabled as frontplanes, each bit in the LCDWF registers corresponds to a segment on an LCD panel. The different phases A-H represent the different backplanes of the LCD panel. The selected LCD duty cycle controls the number of implemented phases. Refer to [Table 10-13](#) for normal LCD operation the phases follow the sequence shown.

For LCD pins enabled as a backplane, the LCDWF assigns the phase in which the backplane pin is active. This is how backplane assignment is done.

An example of normal operation follows: enable LCD pin 0 to operate as backplane 0. Enable the LCD pin 0 by setting PEN0 bit in the LCDPEN0 register. Configure LCD pin 0 as a backplane pin by setting the BPEN0 bit in the LCDBPEN0 register. Finally, the BPALCD0 bit in the LCDWF0 is set to associate LCD pin 0 with backplane phase A. This will configure LCD0 to operate as a backplane that is active in Phase A.

For LCD pins enabled as a frontplane, writing a 1 to a given LCDWF location results in the corresponding display segment being driven with the differential root mean square (RMS) voltage necessary to turn the segment on during the phase selected. Writing a 0 to a given location results in the corresponding display segment being driven with the differential RMS voltage necessary to turn the segment off during the phase selected.

## 10.4.3 LCD Display Modes

The LCD module can be configured to implement several different display modes. The bits ALT and BLANK in the LCD-blink-control register (LCDBCTL) configure the different display modes. In normal display mode (default), LCD segments are controlled by the data placed in the LCDWF registers, as described in [Section 10.4.2, “LCDWF Registers.”](#) For blank-display mode, the LCDWF data is bypassed and the frontplane and backplane pins are configured to clear all segments.

For alternate-display mode, the backplane sequence is modified for duty cycles of 1/4, 1/3, 1/2, and 1/1. For four backplanes or less, the backplane sequence is modified as shown below. The altered sequence allows two complete displays to be placed in the LCDWF registers. The first display is placed in phases A-D and the second in phases E-H in the case of four backplanes. If the LCD duty cycle is five backplanes or greater, the ALT bit is ignored and creates a blank display. Refer to [Table 10-18](#) for additional information.

Using the alternate display function an inverse display can be accomplished for x4 mode and less by placing inverse data in the alternate phases of the LCDWF registers.

**Table 10-17. Alternate Display Backplane Sequence**

Duty	Backplane Sequence	Alt. Backplane Sequence
1/1	A	E
1/2	A B	E F



**Table 10-17. Alternate Display Backplane Sequence (continued)**

Duty	Backplane Sequence	Alt. Backplane Sequence
1/3	A B C	E F G
1/4	A B C D	E F G H

### 10.4.3.1 LCD Blink Modes

The blink mode is used as a means of alternating among different LCD display modes at a defined frequency. The LCD module can be configured to implement two blink modes. The BMODE bit in the LCD-blink-control register (LCDBCTL) configures the different blink modes. Blink modes are activated by setting the BLINK bit in the LCDBCTL register. If BLINK = 0, the LCD module operates normally as described [Section 10.4.3, “LCD Display Modes”](#). If BLINK = 1, BMODE bit configures the blinking operation. During a blink, the display data driven by the LCD module changes to the mode selected by the BMODE bit. The BMODE bit selects two different blink modes, blank and alternate modes operate in the same way, as defined in [Section 10.4.3, “LCD Display Modes.”](#) The table below shows the interaction between display modes and blink modes. If the LCD duty cycle is five backplanes or greater, BMODE = 1 is ignored and will revert to create a blank display during the blink period.

**Table 10-18. Display Mode Interaction**

BLANK	ALT	BMODE	LCD Duty	BLINK = 1	
				Normal Period	Blink Period
0	0	0	1-4	Normal Display	Blank Display
0	0	1	1-4	Normal Display	Alternate display
0	1	0	1-4	Alternate display	Blank Display
0	1	1	1-4	Alternate Display	Alternate display
1	X	0	1-4	Blank Display	Blank Display
1	X	1	1-4	Blank Display	Alternate display
0	X	X	5-8	Normal Display	Blank Display
1	X	X	5-8	Blank Display	Blank Display

### 10.4.3.2 Blink Frequency

The LCD clock is the basis for the calculation of the LCD module blink frequency. The LCD module blink frequency is equal to the LCD clock (LCDCLK) divided by the factor selected by the BRATE[2:0] bits. [Table 10-19](#) shows LCD module blink frequency calculations for all values of BRATE[2:0] at a few common LCDCLK selections.

**Table 10-19. Blink Frequency Calculations**  
(Blink Rate = LCD Clock(Hz) ÷ Blink Divider)

BRATE[2:0]	0	1	2	3	4	5	6	7
LCD Clock	Blink Frequency (Hz)							
30 khz	7.32	3.66	1.831	.916	.46	.23	.11	.06

**Table 10-19. Blink Frequency Calculations**  
 (Blink Rate = LCD Clock(Hz) ÷ Blink Divider)

BRATE[2:0]	0	1	2	3	4	5	6	7
LCD Clock	Blink Frequency (Hz)							
32.768 khz	8	4	2	1	.5	.25	.13	.06
39.063 khz	9.54	4.77	2.38	1.19	.6	.30	.15	.075

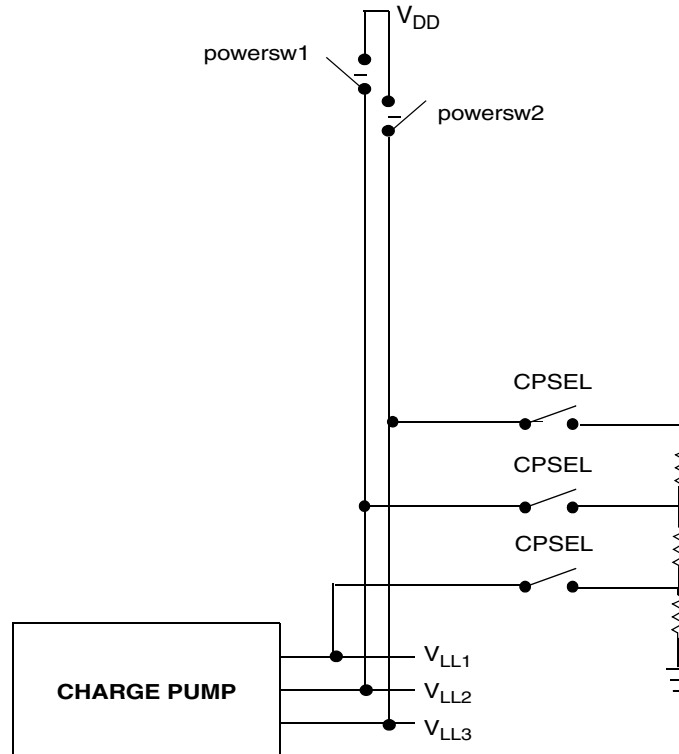
#### 10.4.4 LCD Charge Pump, Voltage Divider, and Power Supply Operation

This section describes the LCD charge pump, voltage divider, and LCD power supply configuration options. [Figure 10-15](#) provides a block diagram for the LCD charge pump and the resistor divider network.

The LCD bias voltages ( $V_{LL1}$ ,  $V_{LL2}$  and  $V_{LL3}$ ) can be generated by the LCD charge pump or a resistor divider network that is connected using the CPSEL bit. The input source to the LCD charge pump is controlled by the VSUPPLY[1:0] bit field.

VSUPPLY[1:0] indicates the state of internal signals used to configure power switches as shown in the table in [Figure 10-15](#). The block diagram in [Figure 10-15](#) illustrates several potential operational modes for the LCD module including configuration of the LCD module power supply source using  $V_{DD}$ , or an external supply on the  $V_{LL3}$ .  $V_{LL3}$  should never exceed  $V_{DD}$ .

Upon Reset the VSUPPLY[1:0] bits are configured to connect  $V_{LL3}$  to  $V_{DD}$ . This configuration should be changed to match the application requirements before the LCD module is enabled.



$V_{SUPPLY}[1:0]$	Configuration	powersw1	powersw2	CPSEL
00	Drive $V_{LL2}$ internally from $V_{DD}$ for 5V glass	1	0	1
01	Drive $V_{LL3}$ internally from $V_{DD}$ for 5V glass	0	1	1
10	Reserved	0	0	1
11	Drive $V_{LL3}$ externally from $V_{DD}$	0	0	1
11	Drive $V_{LL3}$ Externally to $V_{DD}$ for 3V or 5V LCD glass, charge pump disabled, Resistor network creates LCD bias voltages	0	0	0

Figure 10-15. LCD Charge Pump and  $V_{LCD}$  Voltage Divider Block Diagram

**NOTE:**

The charge pump is optimized for 1/3 bias mode operation only.

During the first 16 timebase clock cycles after the LCDCPEN bit is set, all the LCD frontplane and backplane outputs are disabled, regardless of the state of the LCDEN bit.

The charge pump requires external capacitance for its operation. To provide this external capacitance, the  $V_{cap1}$  and  $V_{cap2}$  external pins are provided. It is recommended that a ceramic capacitor be used. Proper orientation is imperative when using a polarized capacitor. The recommended value for the external capacitor is 0.1  $\mu$ F.

**10.4.4.1 LCD Power Supply Configuration**

The LCD bias voltages can be internally derived from  $V_{DD}$ , internally derived from a voltage source (must not exceed  $V_{DD}$ ) connected to  $V_{LL3}$ . [Table 10-21](#) provides a more detailed description of the power state of the LCD module which depends on the configuration of the VSUPPLY[1:0], CPSEL and RVEN bits.

[Table 10-21](#) shows all possible configurations of the LCD Power Supply. All other combinations of the configuration bits above are not permissible LCD power supply modes and should be avoided.

**Table 10-20. LCD Power Supply Options**

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	CPSEL	RVEN
$V_{LL2}$ connected to $V_{DD}$ internally for 5 V glass operation.	For 5 V glass operation $V_{DD}$ must equal 3.33 V Charge pump is used to generate $V_{LL1}$ and $V_{LL3}$	00	1	0
$V_{LL3}$ connected to $V_{DD}$ internally for 3 V or 5 V glass operation	For 3 V glass operation $V_{DD}$ must equal 3 V. For 5 V glass operation $V_{DD}$ must equal 5 V. Charge pump is used to generate $V_{LL1}$ and $V_{LL2}$	01	1	0
$V_{LL3}$ is driven externally for 3 V LCD Glass operation.	For 3 V glass operation $V_{LL3}$ must equal 3 V. Charge pump is used to generate $V_{LL1}$ and $V_{LL2}$	11	1	0
$V_{LL3}$ is driven externally for 5 V LCD Glass operation. $V_{LL3}$ must equal $V_{DD}$	For 5 V glass operation $V_{LL3}$ must equal 5 V. Charge pump is used to generate $V_{LL1}$ and $V_{LL2}$	11	1	0

Table 10-20. LCD Power Supply Options (continued)

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	CPSEL	RVEN
V <sub>LL3</sub> is driven externally for 3 V LCD Glass operation. Resistor Bias Network enabled.	For 3 V glass operation V <sub>LL3</sub> must equal 3 V. Charge pump is disabled. Resistor Bias network is used to create V <sub>LL1</sub> and V <sub>LL2</sub>	11	0	0
V <sub>LL3</sub> is driven externally for 5 V LCD Glass operation. Resistor Bias network enabled. V <sub>LL3</sub> must equal V <sub>DD</sub>	For 5 V glass operation V <sub>LL3</sub> must equal 5 V. Charge pump is disabled. Resistor Bias network is used to create V <sub>LL1</sub> and V <sub>LL2</sub> .	11	0	0

#### 10.4.4.1.1 LCD External Power Supply, VSUPPLY[1:0] = 11

When VSUPPLY[1:0] = 11, powersw1, powersw2, are deasserted. V<sub>DD</sub> is not available to power the LCD module internally, so the LCD module requires an external power source for V<sub>LL1</sub>, V<sub>LL2</sub>, and V<sub>LL3</sub> when the charge pump is disabled.

If the charge pump is enabled, external power must be applied to V<sub>LL3</sub>. With this configuration, the charge pump will generate the other LCD bias voltages V<sub>LL1</sub> and V<sub>LL2</sub>.

#### 10.4.4.1.2 LCD Internal Power Supply, VSUPPLY[1:0] = 00 or 01

V<sub>DD</sub> is used as the LCD module power supply when VSUPPLY[1:0] = 00 or 11 (Table 10-22). When powering the LCD module using V<sub>DD</sub>, the charge pump must be enabled (CPSEL = 1). Table 10-22 provides recommendations regarding configuration of the VSUPPLY[1:0] bit field when using both 3-V and 5-V LCD panels.

Table 10-22. V<sub>DD</sub> Switch Option

VSUPPLY[1:0]	V <sub>DD</sub> Switch Option	Recommend Use for 3-V LCD Panels	Recommend Use for 5-V LCD Panels
00	V <sub>LL2</sub> is generated from V <sub>DD</sub>	<ul style="list-style-type: none"> <li>Invalid LCD configuration</li> </ul>	<ul style="list-style-type: none"> <li>V<sub>LL1</sub> = 1.67 V</li> <li><b>V<sub>DD</sub> = V<sub>LL2</sub> = 3.3 V</b></li> <li>V<sub>LL3</sub> = 5 V</li> </ul>
01	V <sub>LL3</sub> is generated from V <sub>DD</sub>	<ul style="list-style-type: none"> <li>V<sub>LL1</sub> = 1 V</li> <li>V<sub>LL2</sub> = 2 V</li> <li><b>V<sub>DD</sub> = V<sub>LL3</sub> = 3 V</b></li> </ul>	<ul style="list-style-type: none"> <li>V<sub>LL1</sub> = 1.67 V</li> <li>V<sub>LL2</sub> = 3.33 V</li> <li><b>V<sub>DD</sub> = V<sub>LL3</sub> = 5 V</b></li> </ul>

## 10.4.5 Resets

During a reset, the LCD module system is configured in the default mode. The default mode includes the following settings:

- LCDEN is cleared, thereby forcing all frontplane and backplane driver outputs to the high impedance state.
- 1/4 duty
- 1/3 bias
- LCLK[2:0], VSUPPLY[1:0], CPSEL, and BRATE[2:0] revert to their reset values

## 10.4.6 Interrupts

When an LCD module frame-frequency interrupt event occurs, the LCDIF bit in the LCDS register is asserted. The LCDIF bit remains asserted until software clears the LCD-module-frame-frequency interrupt. The interrupt can be cleared by software writing a 1 to the LCDIF bit.

If both the LCDIF bit in the LCDS register and the LCDIEN bit in the LCDC1 register are set, an LCD interrupt signal asserts.

## 10.5 Initialization Section

This section provides a recommended initialization sequence for the LCD module and also includes initialization examples for several LCD application scenarios.

### 10.5.1 Initialization Sequence

The below list provides a recommended initialization sequence for the LCD module.

You must write to all LCDPEN, LCDBPEN, and LCDWF registers to initialize their values after a reset.

1. LCDC0 register
  - a) Configure LCD clock source (SOURCE bit).
2. LCDSUPPLY register
  - a) Enable charge pump (CPSEL bit).
  - b) Configure charge pump clock (LADJ[1:0]).
  - c) Configure LCD power supply (VSUPPLY[1:0]).
3. LCDC1 register
  - a) Configure LCD frame frequency interrupt (LCDIEN bit).
  - b) Configure LCD behavior in low power mode (LCDWAI and LCDSTP bits).
4. LCDC0 register
  - a) Configure LCD duty cycle (DUTY[2:0]).
  - b) Select and configure LCD frame frequency (LCLK[2:0]).
5. LCDBCTL register
  - a) Configure display mode (ALT and BLANK bits).
  - b) Configure blink mode (BMODE).
  - c) Configure blink frequency (BRATE[2:0]).

6. LCDPEN[7:0] register
  - a) Enable LCD module pins (PEN[28:0] bits).
7. LCDBPEN[7:0]
  - a) Enable LCD pins to operate as an LCD backplane (BPEN[28:0]).
8. LCDC0 register
  - a) Enable LCD module (LCDEN bit).

## 10.5.2 Initialization Examples

This section provides initialization information for LCD configuration. Each example details the register and bit-field values required to achieve the appropriate LCD configuration for a given LCD application scenario. The table below lists each example and the setup requirements.

**Table 10-23. LCD Application Scenario**

Example	Operating Voltage, V <sub>DD</sub>	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in WAIT/STOP modes	LCD Power Input
1	2.7 V	External 32.768 kHz	3 V	128	30 Hz	None	WAIT: on STOP: on	Power via External Resistor Bias Network
2	5.5 V	Internal 39.063 kHz	5 V	100	50 Hz	Alternate 0.5 Hz	WAIT: on STOP: off	Power via V <sub>DD</sub>
3	5 V	External 32.768 kHz	5 V	168	30 Hz	Blank 2.0 Hz	WAIT: off STOP: off	Power via V <sub>LL3</sub>

These examples illustrate the flexibility of the LCD module to be configured to meet a wide range of application requirements including:

- clock inputs/sources
- LCD power supply
- LCD glass operating voltage
- LCD segment count
- varied blink modes/frequencies
- LCD frame rate

### 10.5.2.1 Initialization Example 1

**Table 10-24. LCD Setup Requirements for Example 1**

Example	Operating Voltage, V <sub>DD</sub>	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in STOP and WAIT modes	LCD Power Input

Table 10-24. LCD Setup Requirements for Example 1

1	2.7-V	External 32.768 kHz	3-V	128	30 Hz	None	WAIT: on STOP: on	Power via External Resistor Bias Network
---	-------	------------------------	-----	-----	-------	------	----------------------	---

The table below lists the setup values required to initialize the LCD as specified by Example 1:

Table 10-25. Initialization Register Values for Example 1

Register	bit or bit field	Binary Value	Comment
LCDSUPPLY 0-XX--11	LCDCPEN	0	Disable charge pump
	LADJ[1:0]	XX	Configure LCD charge pump clock source
	VSUPPLY[1:0]	11	When VSUPPLY[1:0] = 11, the LCD must be externally powered via External resistor bias network (see <a href="#">Table 10-20</a> ).
LCDC1 0-----11	LCDIEN	0	LCD frame interrupts disabled
	LCDWAI	1	LCD is "on" in WAIT mode
	LCDSTP	1	LCD is "on" in STOP mode
LCDC0 00101111	LCDEN	0	Initialization is done before initializing the LCD module
	SOURCE	0	Selects the external clock reference as the LCD clock input (OSCOU <sub>T</sub> )
	LCLK[2:0]	101	For 1/8 duty cycle, select closest value to the desired 30 Hz LCD frame frequency (see <a href="#">Table 10-14</a> )
	DUTY[2:0]	111	For 128 segments (8x16), select 1/8 duty cycle
LCDBCTL 0XX-XXXX	BLINK	0	No blinking
	ALT	X	Alternate bit is configured during LCD operation
	BLANK	X	Blank bit is configured during LCD operation
	BMODE	X	N/A; Blink Blank = 0; Blink Alternate = 1
	BRATE[2:0]	XXX	N/A
LCDPEN[3:0]	LCDPEN0 LCDPEN1 LCDPEN2 LCDPEN3	11111111 11111111 11111111 00000000	Only 24 LCD pins need to be enabled.  <b>Note:</b> Any of the 28 LCD pins can be used, this allows flexibility in the hardware design.



Table 10-25. Initialization Register Values for Example 1

Register	bit or bit field	Binary Value	Comment
LCDBPEN[3:0]	LCDBPEN0 LCDBPEN1 LCDBPEN2 LCDBPEN3	11111111 00000000 00000000 00000000	Eight backplane pins needed.  <b>Note:</b> Any enabled LCD pin can be enabled to operate as a backplane.
LCDWF[28:0]	LCDWF0 LCDWF1 LCDWF2 LCDWF3 LCDWF4 LCDWF5 LCDWF6 LCDWF7	00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000	Configure which phase the eight backplane pins will be active in. This configuration sets LCD[0] to be active in Phase A, LCD[1] to be active in Phase B...etc This configuration sets LCD pins 0-7 to represent backplane 1-8.  <b>Note:</b> Any backplane pin can be active in any phase.

### 10.5.2.2 Initialization Example 2

Example 2 LCD setup requirements are reiterated in the table below:

Example	Operating Voltage, V <sub>DD</sub>	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in STOP3 and WAIT modes	LCD Power Input
2	5.5 V	Internal 39.063 kHz	3 V	100	50 Hz	Alternate 0.5 Hz	WAIT: on STOP: off	Power via V <sub>DD</sub>

The table below lists the required setup values required to initialize the LCD as specified by Example 2:

Table 10-26. Initialization Register Values for Example 2

Register	bit or bit field	Binary Value	Comment
LCDSUPPLY 1-00--01	LCDCPEN	1	Enable charge pump
	LADJ[1:0]	00	Configure LCD charge pump clock source
	VSUPPLY[1:0]	01	Generate V <sub>LL3</sub> from V <sub>DD</sub> (See <a href="#">Table 10-20</a> )
LCDC1 0-----10	LCDIEN	0	LCD Frame Interrupts disabled
	LCDWAI	1	LCD is "on" in WAIT mode
	LCDSTP	0	LCD is "off" in STOP mode
LCDC0 01010011	LCDEN	0	Initialization is done before initializing the LCD module
	SOURCE	1	Selects the alternate-clock reference as the LCD clock input (ALTCLK) This clock source is configured by the ICS TRIM bits to be 39.063Khz.
	LCLK[2:0]	010	For 1/4 duty cycle, select closest value to the desired 50 Hz LCD frame frequency ( <a href="#">Table 10-15</a> )
	DUTY[2:0]	011	For 100 segments (4x25), select 1/4 duty cycle

Table 10-26. Initialization Register Values for Example 2 (continued)

Register	bit or bit field	Binary Value	Comment
LCDBCTL 1XX-1100	BLINK	1	Blinking is turned on or off during LCD operation
	ALT	X	Alternate bit is configured during LCD operation
	BLANK	X	Blank bit is configured during LCD operation
	BMODE	1	Blink Alternate = 1
	BRATE[2:0]	100	Select 5 Hz blink frequency using <a href="#">Table 10-19</a>
LCDPEN[3:0]	LCDPEN0 LCDPEN1 LCDPEN2 LCDPEN3	11111111 11111111 11111111 00011111	29 LCD pins need to be enabled.
LCDBPEN[3:0]	LCDBPEN0 LCDBPEN1 LCDBPEN2 LCDBPEN3	00001111 00000000 00000000 00000000	Four backplane pins needed. <b>Note:</b> Any enabled LCD pin can be enabled to operate as a backplane.
LCDWF[28:0]	LCDWF0 LCDWF1 LCDWF2 LCDWF3	00000001 00000010 00000100 00001000	Configure which phase the four backplane pins will be active in. This configuration sets LCD[0] to be active in Phase A, LCD[1] to be active in Phase B etc.  This configuration sets LCD pins 0-3 to represent backplane 1-4. <b>Note:</b> Any backplane pin can be active in any of the phases.

### 10.5.2.3 Initialization Example 3

Example 3 LCD setup requirements are reiterated in the table below:

Example	Operating Voltage, V <sub>DD</sub>	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in STOP3 and WAIT modes	LCD Power Input
3	5.0 V	External 32.768 kHz	5 V	168	30 Hz	Blank 2.0 Hz	WAIT: off STOP: off	Power via V <sub>LL3</sub>

The table below lists the required setup values required to initialize the LCD as specified by Example 3:

Table 10-27. Initialization Register Values for Example 3

Register	bit or bit field	Binary Value	Comment
LCDSUPPLY 1X00-X00	LCDCPEN	1	Enable charge pump
	LADJ[1:0]	00	Configure LCD charge pump clock source
	VSUPPLY[1:0]	11	When VSUPPLY[1:0] = 11, the LCD can be powered via V <sub>LL3</sub> (see <a href="#">Table 10-20</a> ).

Table 10-27. Initialization Register Values for Example 3 (continued)

Register	bit or bit field	Binary Value	Comment
LCDC1 0-----00	LCDIEN	0	LCD Frame Interrupts disabled
	LCDWAI	0	LCD is "off" in WAIT mode
	LCDSTP	0	LCD is "off" in STOP mode
LCDC0 00101111	LCDEN	0	Initialization is done before initializing the LCD module
	SOURCE	0	Selects OSCOUT as the LCD clock source (32.768 Khz crystal)
	LCLK[2:0]	101	For 1/8 duty cycle, select closest value to the desired 30 Hz LCD frame frequency (see Table 10-14)
	DUTY[2:0]	111	For 168 segments (8x21), select 1/8 duty cycle
LCDBCTL XXX-0010	BLINK	X	Blinking is turned on or off during LCD operation
	ALT	X	Alternate bit is configured during LCD operation
	BLANK	X	Blank bit is configured during LCD operation
	BMODE	0	Blink to a blank mode
	BRATE[2:0]	010	Select 2 Hz blink frequency using Table 10-19
LCDPEN[3:0]	LCDPEN0	11111111	29 LCD pins need to be enabled.
	LCDPEN1	11111111	
	LCDPEN2	11111111	
	LCDPEN3	00011111	
LCDBPEN[3:0]	LCDBPEN0	11111111	Eight backplane pins needed.  <b>Note:</b> Any enabled LCD pin can be enabled to operate as a backplane
	LCDBPEN1	00000000	
	LCDBPEN2	00000000	
	LCDBPEN3	00000000	
LCDWF[28:0]	LCDWF0	00000001	Configure which phase the eight backplane pins will be active in. This configuration sets LCD[0] to be active in Phase A, LCD[1] to be active in Phase B... etc.
	LCDWF1	00000010	
	LCDWF2	00000100	
	LCDWF3	00001000	
	LCDWF4	00010000	This configuration sets LCD pins 0-7 to represent backplane 1-8.  <b>Note:</b> Any backplane pin can be active in any phase
	LCDWF5	00100000	
	LCDWF6	01000000	
	LCDWF7	10000000	

## 10.6 Application Information

Figure 10-16 is a programmer's model of the LCD module. The programmer's model groups the LCD module register bit and bit field into functional groups. The model is a high-level illustration of the LCD module showing the module's functional hierarchy including initialization and runtime control.

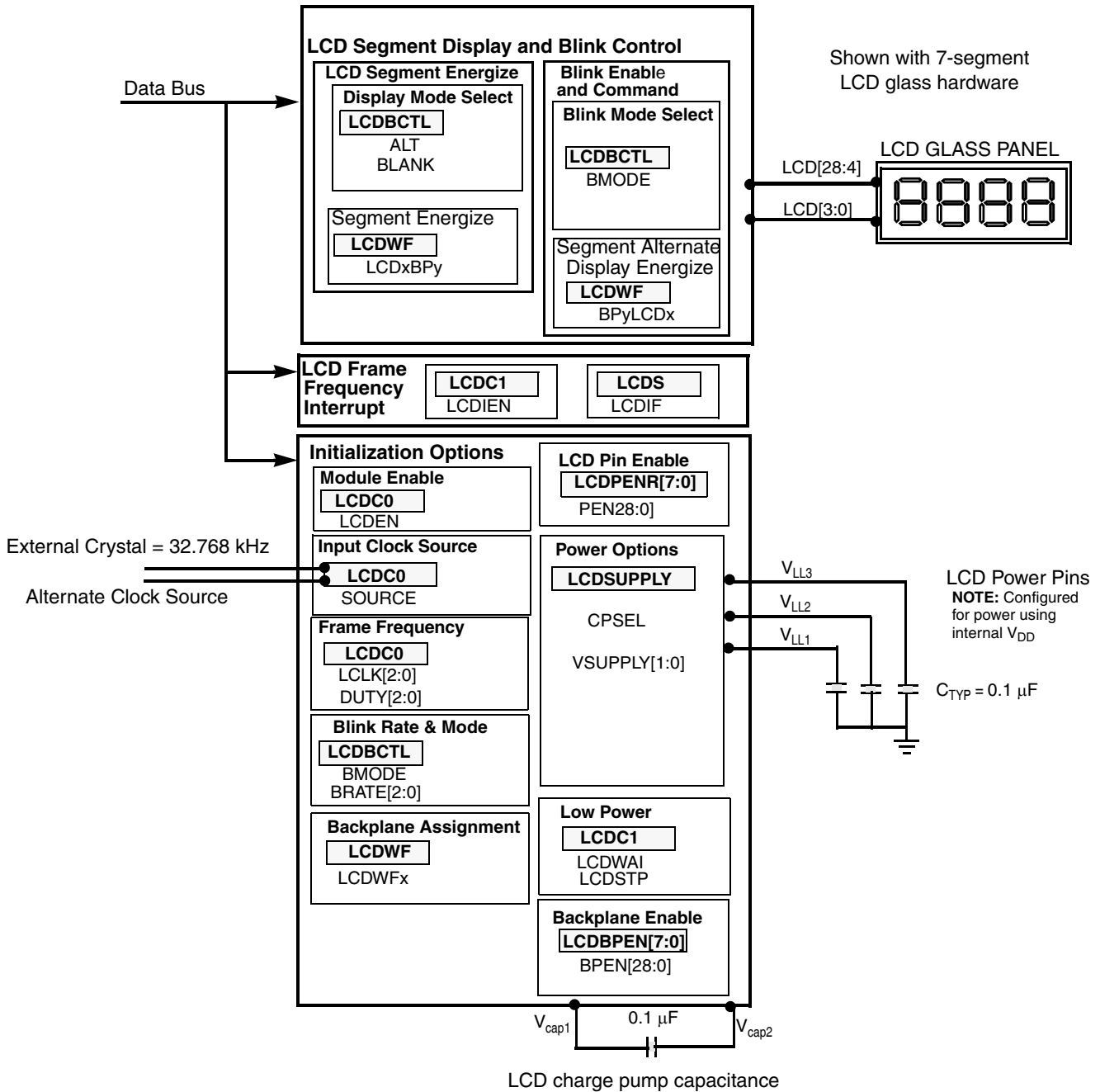
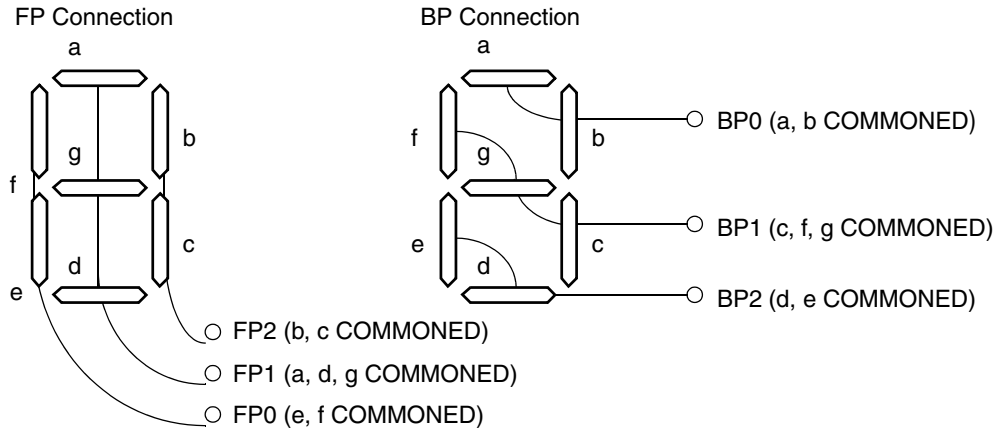


Figure 10-16. LCD Programmer's Model Diagram

### 10.6.1 LCD Seven Segment Example Description

A description of the connection between the LCD module and a seven segment LCD character is illustrated below to provide a basic example for a 1/3 duty cycle LCD implementation. The example uses three backplane pins (LCD[3], LCD[4] and LCD[5] and 3 frontplane pins (LCD[0], LCD[1], and LCD[2]). LCDWF contents and output waveforms are also shown. Output waveforms are illustrated in Figure 10-17 and Figure 10-18.



The above segment assignments are provided by the specification for the LCD glass for this example. For this LCD Module any of the LCD pins can be configured to be Frontplane 0-2 or Backplane 0-2. For this example we will set LCD[0] as FP0, LCD[1] as FP1, and LCD[2] as FP2. For this example we will set LCD[3] as BP0, LCD[4] as BP1 and LCD[5] as BP2.

Backplane assignment is done in the LCDWF register as shown below:

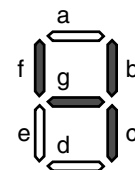
LCDWF3	0	0	0	0	0	0	0	1
LCDWF4	0	0	0	0	0	0	1	0
LCDWF5	0	0	0	0	0	1	0	0

With the above conditions the segment assignment is shown below:

LCDWF0	-	-	-	-	-	e	f	-
LCDWF1	-	-	-	-	-	d	g	a
LCDWF2	-	-	-	-	-	-	c	b

To display the character "4": LCDWF0 = XXXXX01X, LCDWF1 = XXXXX010, LCDWF2 = XXXXXX11

LCDWF0	X	X	X	X	X	0	1	X
LCDWF1	X	X	X	X	X	0	1	0
LCDWF2	X	X	X	X	X	X	1	1



X = don't care

**Figure 10-17. Waveform Output from LCDWF Registers**

### 10.6.1.1 LCD Module Waveforms

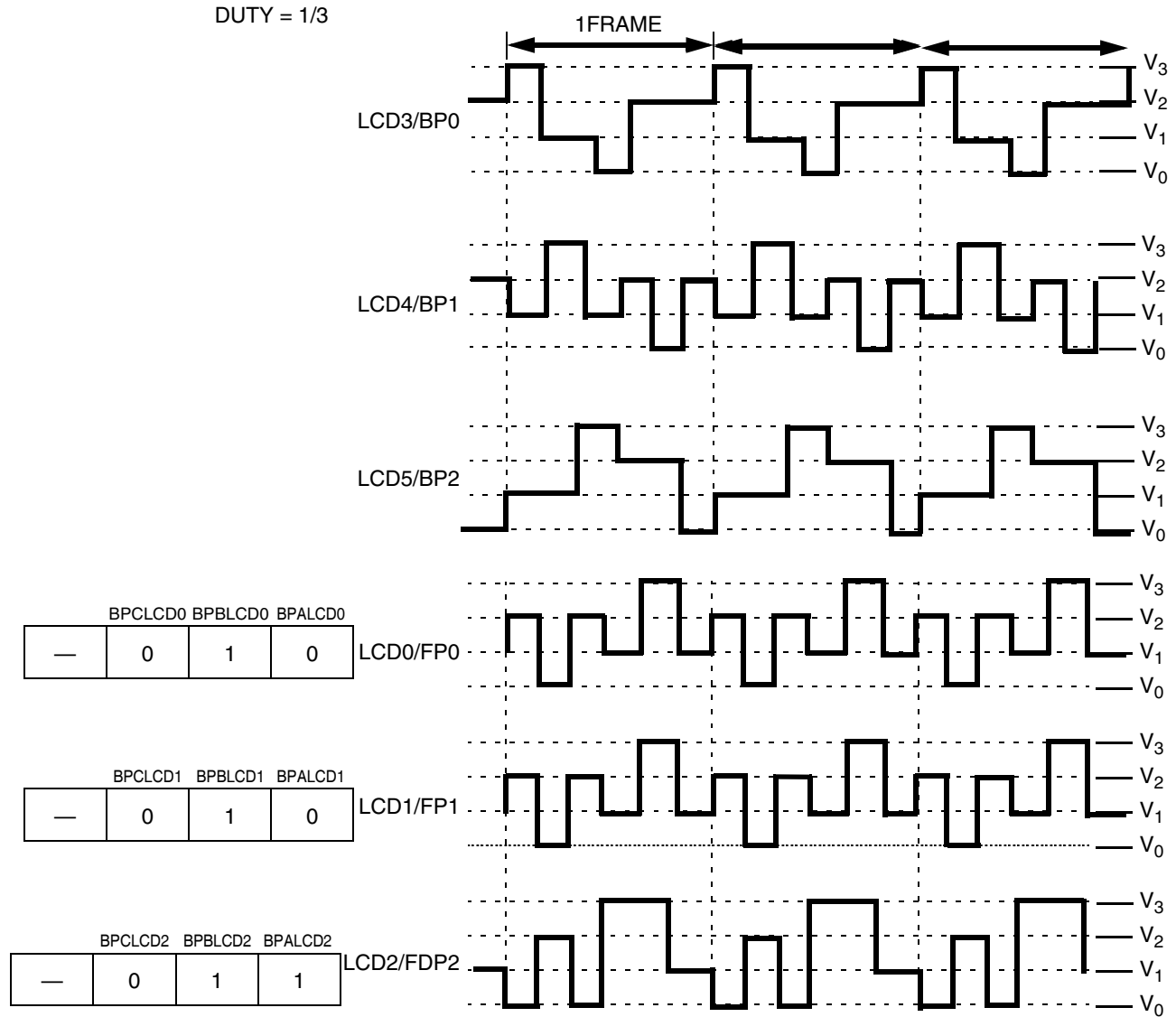


Figure 10-18. LCD Waveforms

### 10.6.1.2 Segment On Driving Waveform

The voltage waveform across the “f” segment of the LCD (between LCD[4]/BP1 and LCD[0]/FP0) is illustrated in Figure 10-19. As shown in the waveform, the voltage level reaches the value  $V_3$  therefore the segment will be on.

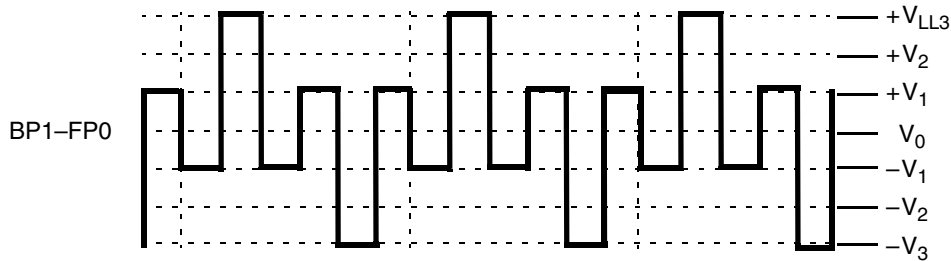


Figure 10-19. “f” Segment Voltage Waveform

### 10.6.1.3 Segment Off Driving Waveform

The voltage waveform across the “e” segment of the LCD (between LCD[5]/BP2 and LCD[0]/FP0) is illustrated in Figure 10-20. As shown in the waveform, the voltage does not reach the voltage  $V_3$  threshold therefore the segment will be off.

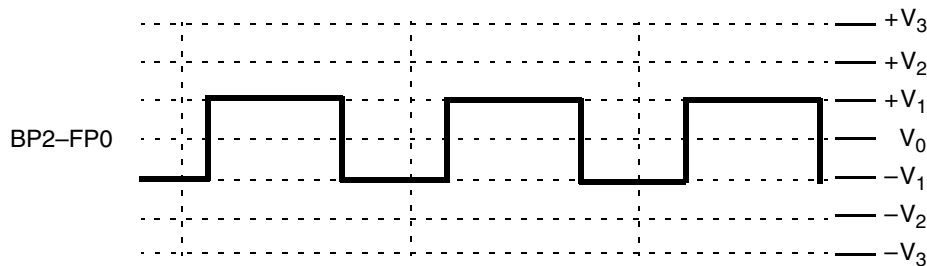


Figure 10-20. “e” Segment Voltage Waveform

## 10.6.2 LCD Contrast Control

Contrast control for the LCD module is achieved when the LCD power supply is adjusted above and below the LCD threshold voltage. The LCD threshold voltage is the nominal voltage required to energize the LCD segments. For 3-V LCD glass, the LCD threshold voltage is 3 V; while for 5-V LCD glass it is 5 V. By increasing the value of the LCD voltage, the energized segments on the LCD glass will become more opaque. Decreasing the value of the LCD voltage makes the energized segments on the LCD glass become more transparent. The LCD power supply can be adjusted to facilitate contrast control by using external components like a variable resistor.

**NOTE:**  
 Contrast control configuration  
 when LCD is powered using  
 internal  $V_{DD}$

$V_{DD}$  is specified between 2.7  
 and 5.5V.

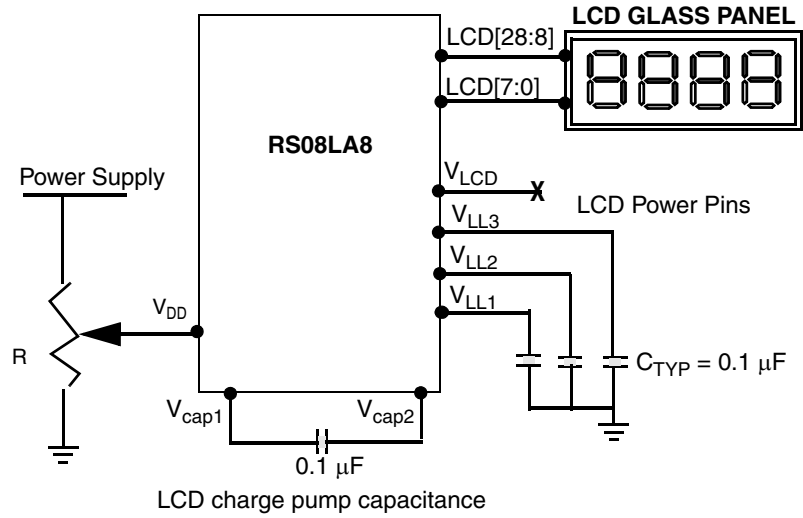


Figure 10-21. Power Connections for Contrast Control



---

# Chapter 11

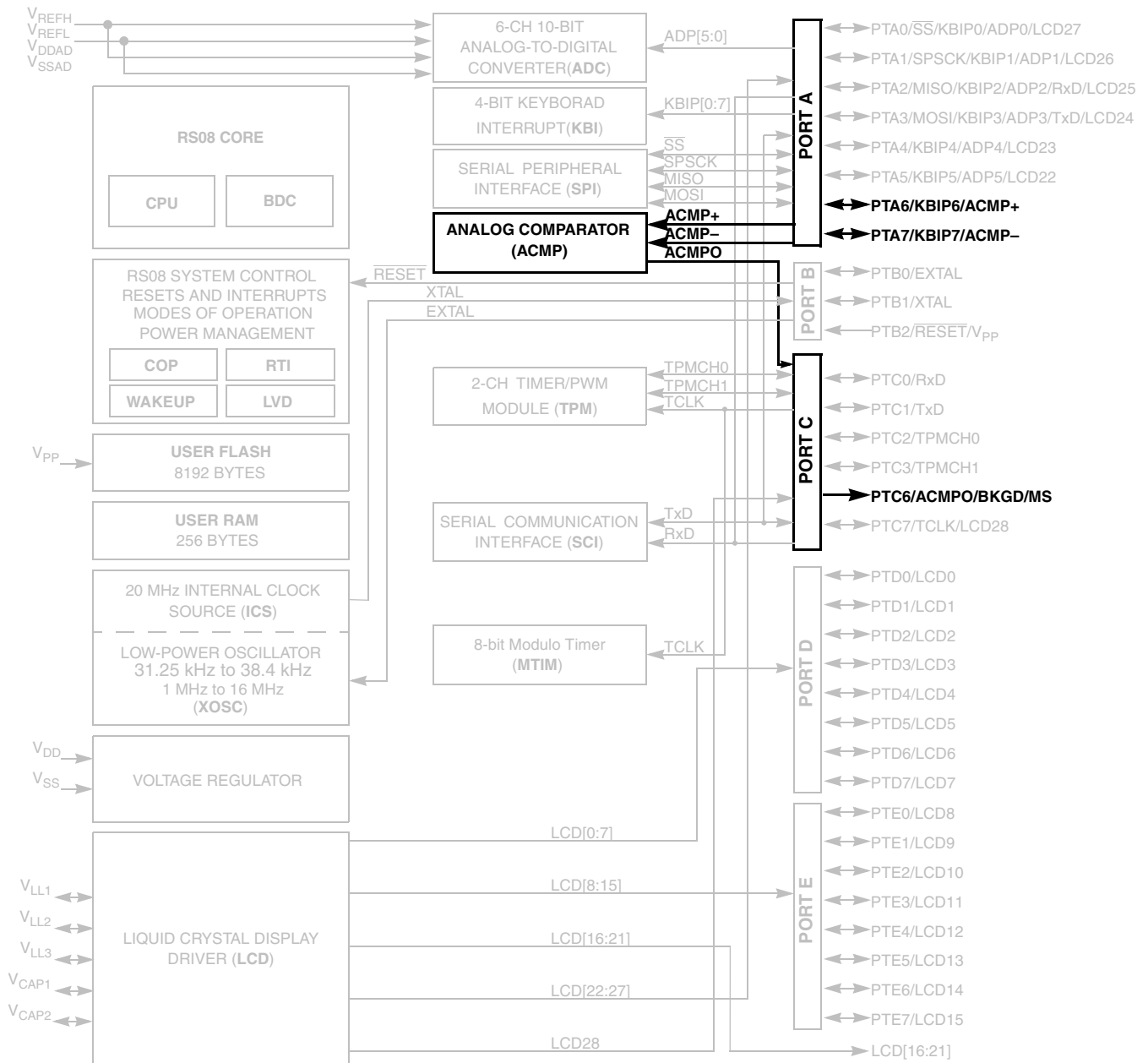
## Analog Comparator (RS08ACMPV1)

### 11.1 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).

The MC9RS08LA8 supports all features of the RS08ACMPV1.

[Figure 11-1](#) shows the MC9RS08LA8 block diagram highlighting the ACMP block and pins.



**NOTES:**

1. PTB2/RESET/ $V_{PP}$  is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 11-1. MC9RS08LA8 Series Block Diagram Highlighting ACMP Block and Pins**

## 11.1.1 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
- Remains operational in stop mode

## 11.1.2 Modes of Operation

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

### 11.1.2.1 Operation in Wait Mode

The ACMP continues to operate 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 is enabled (ACIE = 1). For lowest possible current consumption, the ACMP must be disabled by software if not required as an interrupt source during wait mode.

### 11.1.2.2 Operation in Stop Mode

The ACMP continues to operate in stop mode if enabled and compare operation remains active. If ACOPE is enabled, comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. The MCU is brought out of stop when a compare event occurs and ACIE is enabled; ACF flag sets accordingly.

If stop is exited with a reset, the ACMP will be put into its reset state.

### 11.1.2.3 Operation in Active Background Mode

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

## 11.1.3 Block Diagram

The block diagram for the analog comparator module is shown in [Figure 11-2](#).

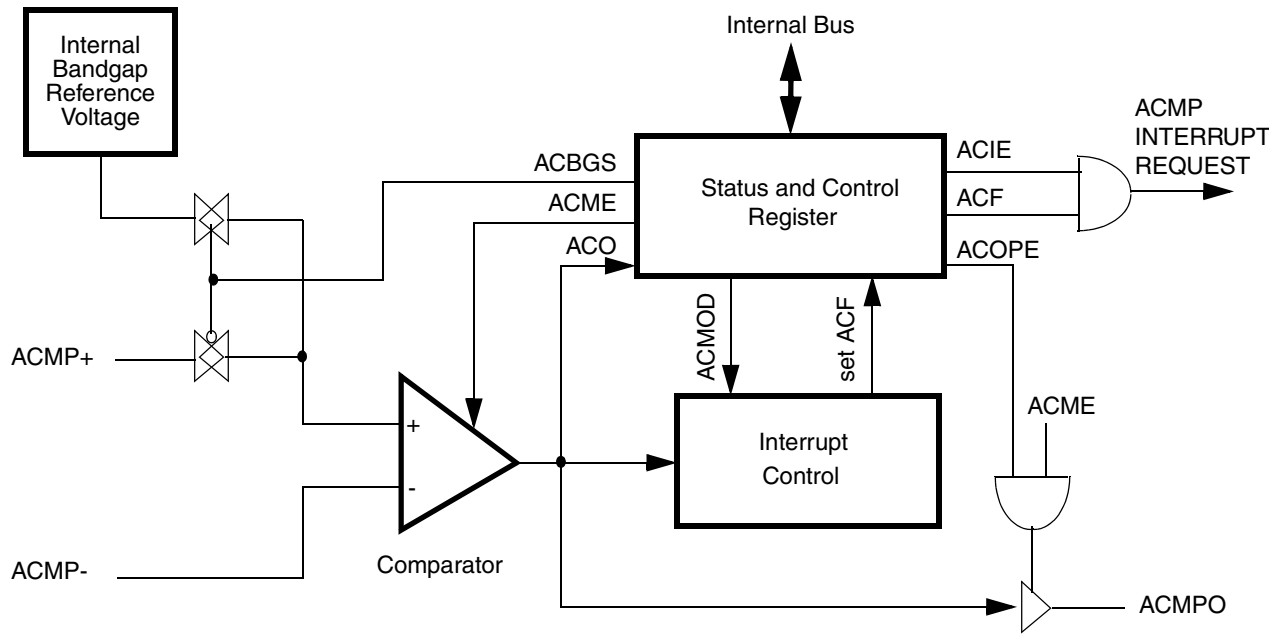


Figure 11-2. Analog Comparator (ACMP) Block Diagram

## 11.2 External Signal Description

The ACMP has two analog input pins, ACMP+ and ACMP–, and one digital output pin, ACMPO. Each of the input pins can accept an input voltage that varies across the full operating voltage range of the MCU. As shown in Figure 11-2, the ACMP– pin is connected to the inverting input of the comparator, and the ACMP+ pin is connected to the non-inverting input of the comparator if ACBGS=0. As shown in Figure 11-2, the ACMPO pin can be enabled to drive an external pin.

The signal properties of ACMP are shown in Table 11-1.

Table 11-1. 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

## 11.3 Register Definition

The ACMP includes one register:

- An 8-bit status and control register

Refer to the direct-page register summary in the memory chapter of this data sheet for the absolute address assignments for all ACMP registers.

## 11.3.1 ACMP Status and Control Register (ACMPSC)

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

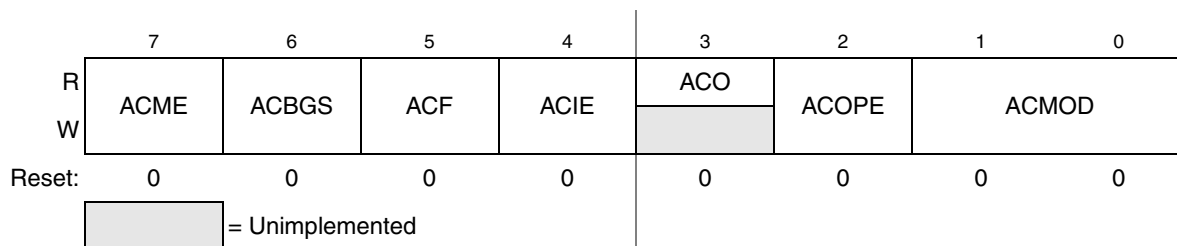


Figure 11-3. ACMP Status and Control Register (ACMPSC)

Table 11-2. ACMPSC Field Descriptions

Field	Description
7 ACME	<b>Analog Comparator Module Enable</b> — ACME enables the ACMP module. 0 ACMP not enabled. 1 ACMP is enabled.
6 ACBGS	<b>Analog Comparator Bandgap Select</b> — ACBGS is used to select between the internal bandgap reference voltage or the ACMP+ pin as the non-inverting input of the analog comparator. 0 External pin ACMP+ selected as non-inverting input to comparator. 1 Internal bandgap reference voltage selected as non-inverting input to comparator.
5 ACF	<b>Analog Comparator Flag</b> — 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	<b>Analog Comparator Interrupt Enable</b> — ACIE enables the interrupt for the ACMP. When ACIE is set, an interrupt will be asserted when ACF is set. 0 Interrupt disabled. 1 Interrupt enabled.
3 ACO	<b>Analog Comparator Output</b> — 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	<b>Analog Comparator Output Pin Enable</b> — ACOPE is used to enable the comparator output to be placed onto the external pin, ACMPO. ACOPE will only control the pin if the ACMP is active (ACME=1). 0 Analog comparator output not available on ACMPO. 1 Analog comparator output is driven out on ACMPO.
1:0 ACMOD	<b>Analog Comparator Mode</b> — 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.

## 11.4 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 it 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 also be driven onto the ACMPO pin using ACOPE.

#### **NOTE**

Comparator inputs are high impedance analog pins which are sensitive to noise. Noisy VDD and/or pin toggling adjacent to the analog inputs may cause the comparator offset/hysteresis performance to exceed the specified values. Maximum source impedance is restricted to the value specified in the *MC9RS08LA8 Data Sheet*. To achieve maximum performance device is recommended to enter WAIT/STOP mode for ACMP measurement and adjacent pin toggling must be avoided.

# Chapter 12

## 10-Bit Analog-to-Digital Converter (RS08ADC10V1)

### 12.1 Introduction

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

All ADC pins are shared with LCD pins. ADC functionality must be disabled for these pins to operate as LCD pins.

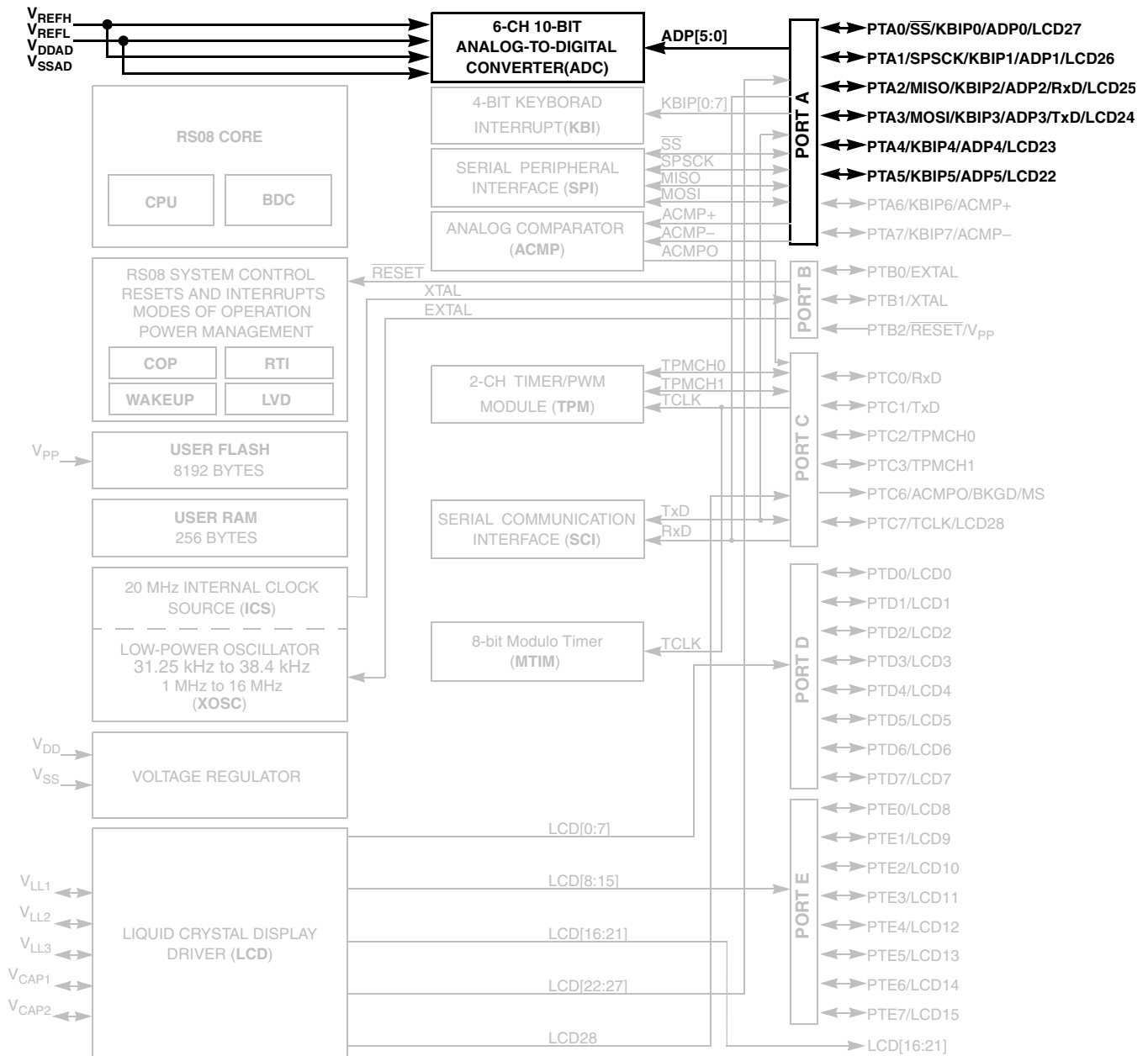
#### 12.1.1 ADC Channel Assignments

Figure 12-1 shows the ADC channel assignments. Reserved channels convert to an unknown value.

ADCH	Input Select	ADCH	Input Select
00000	AD0	10000	V <sub>REFL</sub>
00001	AD1	10001	Reserved
00010	AD2	10010	Reserved
00011	AD3	10011	Reserved
00100	AD4	10100	Reserved
00101	AD5	10101	V <sub>LL1</sub>
00110	V <sub>REFL</sub>	10110	Reserved
00111	V <sub>REFL</sub>	10111	Reserved
01000	V <sub>REFL</sub>	11000	Reserved
01001	V <sub>REFL</sub>	11001	Reserved
01010	V <sub>REFL</sub>	11010	Temperature Sensor
01011	V <sub>REFL</sub>	11011	Bandgap
01100	V <sub>REFL</sub>	11100	V <sub>REFH</sub>
01101	V <sub>REFL</sub>	11101	V <sub>REFH</sub>
01110	V <sub>REFL</sub>	11110	V <sub>REFL</sub>
01111	V <sub>REFL</sub>	11111	Module disabled

Figure 12-1. ADC Channel Assignment

Figure 12-2 shows the MC9RS08LA8 block diagram highlighting the ADC block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 12-2. MC9RS08LA8 Series Block Diagram Highlighting ADC Block and Pins**

### 12.1.2 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock (ALTCLK). The



ALTCLK on the MC9RS08LA8 is connected to the IC SERCLK. See [Chapter 9, “Internal Clock Source \(RS08ICSV1\)”](#), for more information.

### 12.1.3 Hardware Trigger

The ADC hardware trigger is connected RTI module. When enabled, RTI overflow initiates an ADC conversion if ADC is configured to use hardware trigger.

### 12.1.4 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. [Equation 12-1](#) provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{\text{TEMP}} - V_{\text{TEMP25}}) \div m) \quad \text{Eqn. 12-1}$$

where:

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

For temperature calculations, use the  $V_{\text{TEMP25}}$  and  $m$  values in the data sheet.

In application code, the user reads the temperature sensor channel, calculates  $V_{\text{TEMP}}$ , and compares it to  $V_{\text{TEMP25}}$ . If  $V_{\text{TEMP}}$  is greater than  $V_{\text{TEMP25}}$  the cold slope value is applied in [Equation 12-1](#). If  $V_{\text{TEMP}}$  is less than  $V_{\text{TEMP25}}$  the hot slope value is applied in [Equation 12-1](#).

## 12.1.5 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 stop 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.

## 12.1.6 Block Diagram

Figure 12-3 provides a block diagram of the ADC module

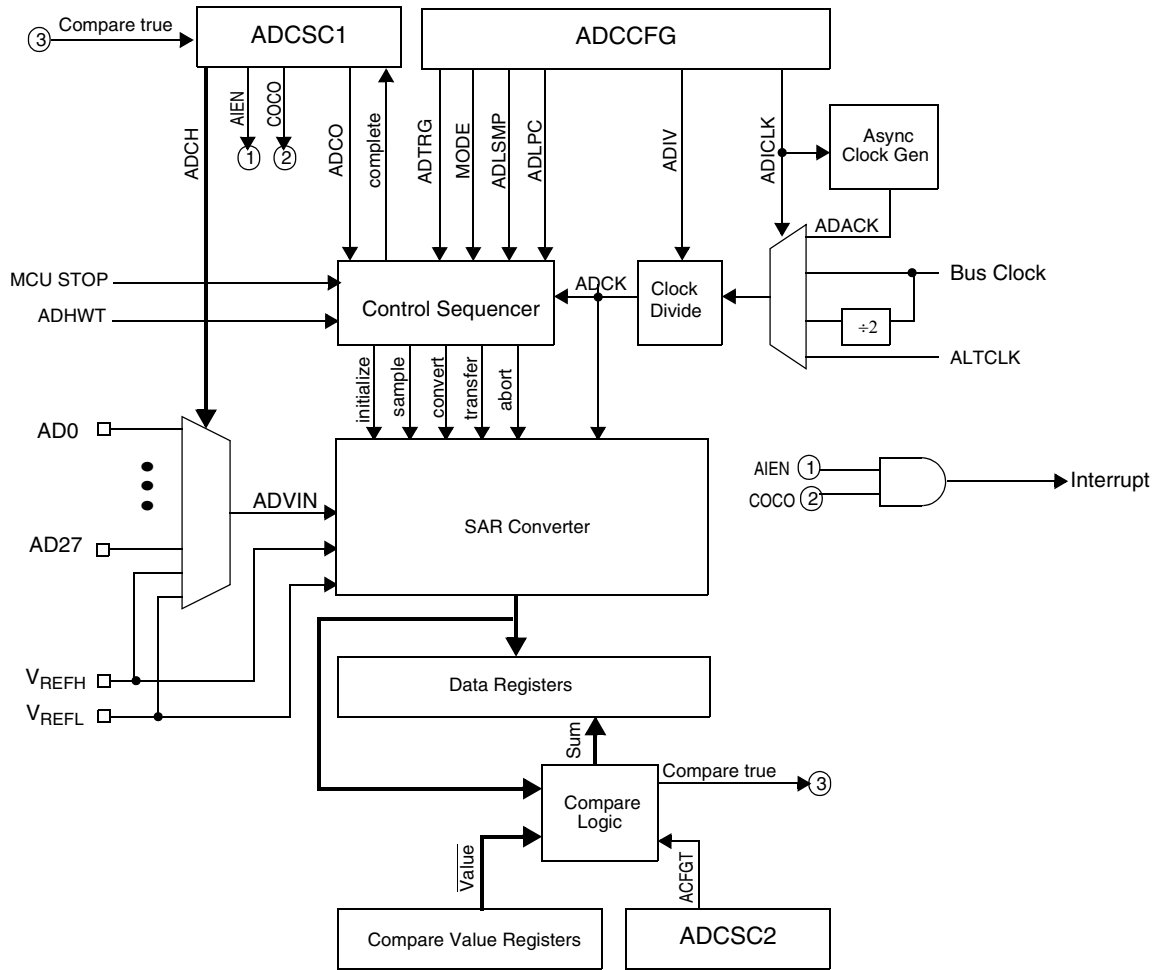


Figure 12-3. ADC Block Diagram

## 12.2 External Signal Description

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

Table 12-1. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V <sub>REFH</sub>	High reference voltage
V <sub>REFL</sub>	Low reference voltage
V <sub>DDAD</sub>	Analog power supply
V <sub>SSAD</sub>	Analog ground

### 12.2.1 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 might be necessary to ensure clean  $V_{DDAD}$  for good results.

### 12.2.2 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  $V_{SS}$ .

### 12.2.3 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,  $V_{REFH}$  can be connected to the same potential as  $V_{DDAD}$ , or can be driven by an external source that is between the minimum  $V_{DDAD}$  spec and the  $V_{DDAD}$  potential ( $V_{REFH}$  must never exceed  $V_{DDAD}$ ).

### 12.2.4 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  $V_{SSAD}$ .

### 12.2.5 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.

## 12.3 Register Definition

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

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

### 12.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register 1 (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).

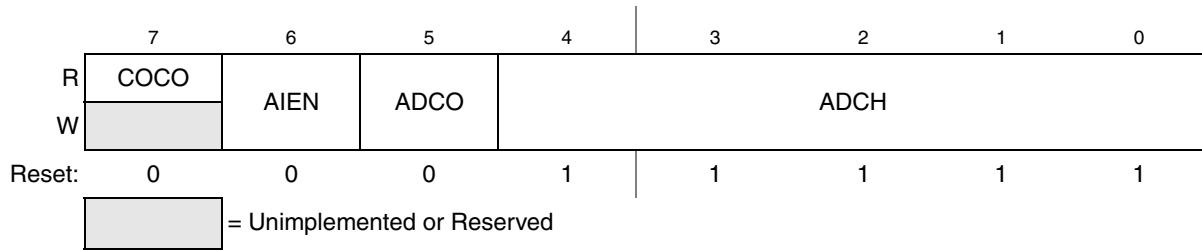


Figure 12-4. Status and Control Register (ADCSC1)

Table 12-2. ADCSC1 Register Field Descriptions

Field	Description
7 COCO	<p><b>Conversion Complete Flag</b> — 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.</p> <p>0 Conversion not completed 1 Conversion completed</p>
6 AIEN	<p><b>Interrupt Enable</b> — AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted.</p> <p>0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled</p>
5 ADCO	<p><b>Continuous Conversion Enable</b> — ADCO is used to enable continuous conversions.</p> <p>0 One conversion following a triggered operation<sup>1</sup> 1 Continuous conversions initiated following a triggered operation is selected.</p>
4:0 ADCH	<p><b>Input Channel Select</b> — The ADCH bits form a 5-bit field which selects one of the input channels. The input channels are detailed in <a href="#">Table 12-3</a>.</p> <p>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 prevents 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.</p>

<sup>1</sup> See [Table 12-4](#) for how to specify a hardware or software trigger type (ADTRG).

Table 12-3. 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
00111	AD7	10111	AD23

Table 12-3. Input Channel Select (continued)

ADCH	Input Select	ADCH	Input Select
01000	AD8	11000	AD24
01001	AD9	11001	AD25
01010	AD10	11010	AD26
01011	AD11	11011	AD27
01100	AD12	11100	Reserved
01101	AD13	11101	V <sub>REFH</sub>
01110	AD14	11110	V <sub>REFL</sub>
01111	AD15	11111	Module disabled

### 12.3.2 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.



<sup>1</sup> Bits 1 and 0 are reserved bits that must always be written to 0.

Figure 12-5. Status and Control Register 2 (ADCSC2)

Table 12-4. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	<b>Conversion Active</b> — 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	<b>Conversion Trigger Select</b> — ADTRG selects the type of trigger to be used for initiating a conversion. 0 Software trigger selected.(initiates a conversion following a write to ADCSC1). 1 Hardware trigger selected.(initiates a conversion following the assertion of the ADHWT input).
5 ACFE	<b>Compare Function Enable</b> — ACFE is used to enable the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	<b>Compare Function Greater Than Enable</b> — ACFGT configures the compare function to trigger upon conversion of the input being monitored. 0 Compare triggered when input is less than compare level 1 Compare triggered when input is greater than or equal to compare level

### 12.3.3 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.

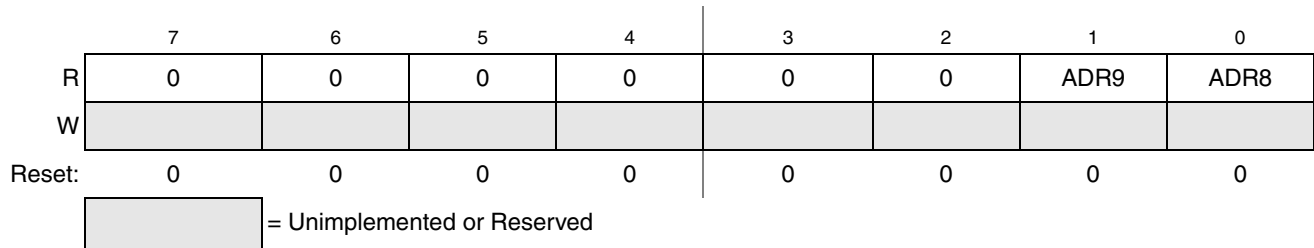


Figure 12-6. Data Result High Register (ADCRH)

### 12.3.4 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 updates each time a conversion completes except when an 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 completes, then the intermediate conversion results are lost. In 8-bit mode, there is no interlocking with ADCRH. When MODE bits are changed, data in ADCRL becomes invalid.



Figure 12-7. Data Result Low Register (ADCRL)

### 12.3.5 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.

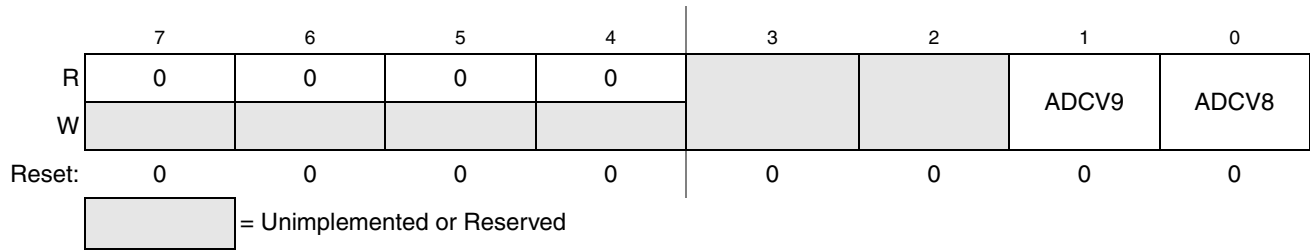


Figure 12-8. Compare Value High Register (ADCCVH)

### 12.3.6 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 10-bit or 8-bit mode.

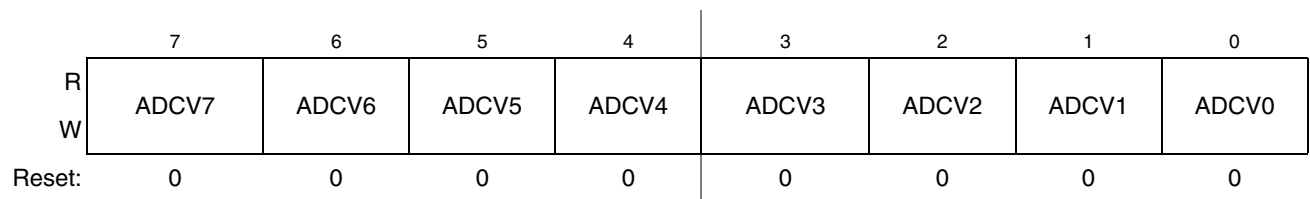


Figure 12-9. Compare Value Low Register(ADCCVL)

### 12.3.7 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.

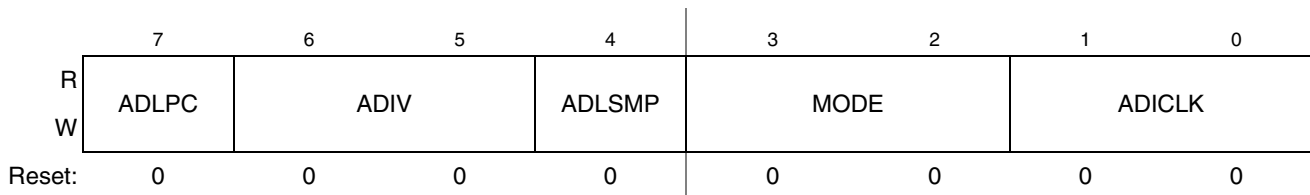


Figure 12-10. Configuration Register (ADCCFG)

Table 12-5. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	<b>Low Power Configuration</b> — ADLPC controls the speed and power configuration of the successive approximation converter. This optimize power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: {FC31} power is reduced at the expense of maximum clock speed.
6:5 ADIV	<b>Clock Divide Select</b> — ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. <a href="#">Table 12-6</a> shows the available clock configurations.



Table 12-5. ADCCFG Register Field Descriptions (continued)

Field	Description
4 ADLSMP	<b>Long Sample Time Configuration</b> — ADLSMP selects between long and short sample periods. 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	<b>Conversion Mode Selection</b> — MODE bits are used to select between 10- or 8-bit operation. See <a href="#">Table 12-7</a> .
1:0 ADICLK	<b>Input Clock Select</b> — ADICLK bits select the input clock source to generate the internal clock ADCK. See <a href="#">Table 12-8</a> .

Table 12-6. 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 12-7. Conversion Modes

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

Table 12-8. Input Clock Select

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

### 12.3.8 Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1

controls the pins associated with channels 0–7 of the ADC module.

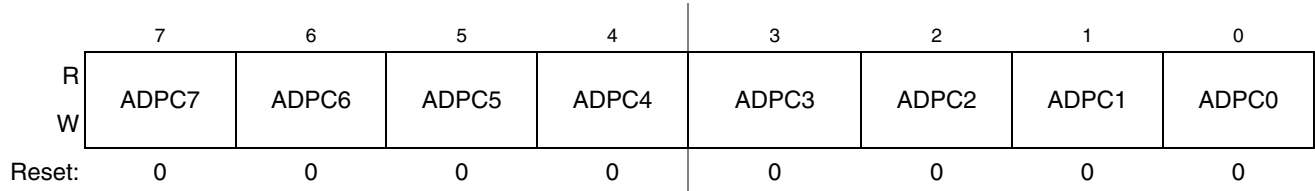


Figure 12-11. Pin Control 1 Register (APCTL1)

Table 12-9. APCTL1 Register Field Descriptions

Field	Description
7 ADPC7	<b>ADC Pin Control 7</b> — 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	<b>ADC Pin Control 6</b> — 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	<b>ADC Pin Control 5</b> — 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	<b>ADC Pin Control 4</b> — 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	<b>ADC Pin Control 3</b> — 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	<b>ADC Pin Control 2</b> — 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
1 ADPC1	<b>ADC Pin Control 1</b> — 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	<b>ADC Pin Control 0</b> — 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

### 12.3.9 Pin Control 2 Register (APCTL2)

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

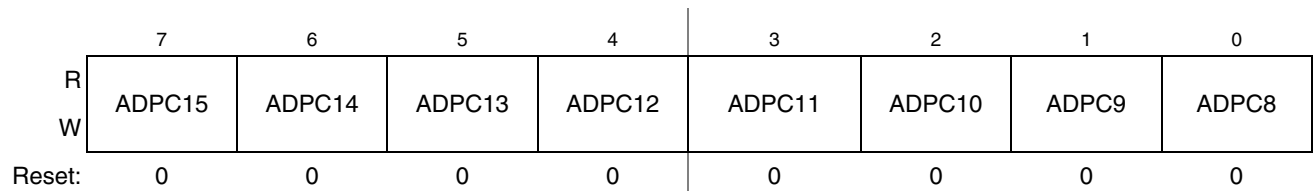


Figure 12-12. Pin Control 2 Register (APCTL2)

**Table 12-10. APCTL2 Register Field Descriptions**

Field	Description
7 ADPC15	<b>ADC Pin Control 15</b> — 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	<b>ADC Pin Control 14</b> — 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	<b>ADC Pin Control 13</b> — 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	<b>ADC Pin Control 12</b> — 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	<b>ADC Pin Control 11</b> — 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	<b>ADC Pin Control 10</b> — 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
1 ADPC9	<b>ADC Pin Control 9</b> — 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	<b>ADC Pin Control 8</b> — 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

### 12.3.10 Pin Control 3 Register (APCTL3)

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

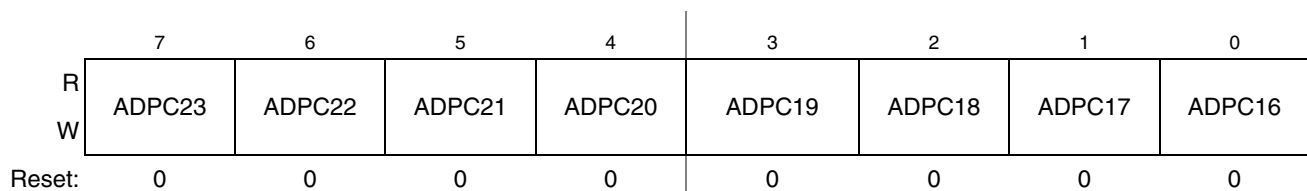
**Figure 12-13. Pin Control 3 Register (APCTL3)**

Table 12-11. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	<b>ADC Pin Control 23</b> — 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	<b>ADC Pin Control 22</b> — 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	<b>ADC Pin Control 21</b> — 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	<b>ADC Pin Control 20</b> — 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	<b>ADC Pin Control 19</b> — 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	<b>ADC Pin Control 18</b> — 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
1 ADPC17	<b>ADC Pin Control 17</b> — 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	<b>ADC Pin Control 16</b> — 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

## 12.4 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 into a 9-bit digital result by a successive approximation algorithm.

When the conversion completes, 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 can automatically compare a conversion result with the contents of its compare registers. Set the ACFE bit to enable the compare function and operate in conjunction with any of the conversion modes and configurations.

### 12.4.1 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, equal to the frequency at which the 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 stop mode and allows conversions in these modes for lower noise operation.

The selected clock's frequency must fall within the range specified for ADCK. If the available clocks are too slow, the ADC will not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divided by 1, 2, 4, or 8.

### 12.4.2 Input Select and Pin Control

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

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

### 12.4.3 Hardware Trigger

The ADC module enables ADHWT, a selectable asynchronous hardware conversion trigger, 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 initiates 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 with any of the conversion modes and configurations.

### 12.4.4 Conversion Control

Conversions can be performed in 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.

#### 12.4.4.1 Initiating Conversions

A conversion is initiated:

- A write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- A hardware trigger (ADHWT) event if hardware triggered operation is selected.
- 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.

#### 12.4.4.2 Completing Conversions

A conversion completes when the result transferred into the data-result registers, ADCRH and ADCRL. The setting of COCO. An interrupt is generated if AIEN is high when 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; 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 terminates. In all other cases, when a data transfer is blocked, another conversion initiates 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, the data registers must not be read after initiating a single conversion until the conversion completes.

#### 12.4.4.3 Aborting Conversions

Any conversion in progress aborts when:

- A write to ADCSC1 occurs (the current conversion is aborted and a new conversion is 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 ADCRH and ADCRL, data registers is not altered but continues to be the values transferred after the completion of the last successful conversion. If a reset aborts the conversion, ADCRH and ADCRL return to their reset states.

#### 12.4.4.4 Power Control

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

Setting ADLPC can reduce power consumption, resulting in a lower maximum value for  $f_{ADCK}$  (see the electrical specifications).

#### 12.4.4.5 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 completes, 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 conversion result transfers 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 12-12](#).

**Table 12-12. 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 $\mu$ s + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	0	5 $\mu$ s + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 $\mu$ s + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	1	5 $\mu$ 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 chosen clock source and the divide ratio determine the maximum total conversion time. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits.

For example, you could use the following equation conversion time for a single conversion in 10-bit mode:

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

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

where:

- **Mode:** 10-Bit
- **Input Clock Source:** Bus Clock
- **Input Clock Ratio:** Divide by 1
- **Bus Frequency:** 8 MHz
- **Conversion Time:** 3.5  $\mu\text{s}$

#### NOTE

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

### 12.4.5 Automatic Compare Function

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

Following a conversion where the compare function is enabled, and 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 wait or stop mode. The ADC interrupt wakes the MCU when the compare condition is met.

### 12.4.6 MCU Wait Mode Operation

The WAIT instruction puts the MCU in a lower power-consumption standby mode. 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.



While in wait mode, the bus clock, bus clock divided by two, and ADACK are available as conversion clock sources. Using 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).

## 12.4.7 MCU Stop Mode Operation

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

### 12.4.7.1 Stop Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not 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 stop mode. After exiting from stop mode, a software or hardware trigger is required to resume conversions.

### 12.4.7.2 Stop Mode With ADACK Enabled

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

If a conversion is in progress when the MCU enters stop mode, it continues until completion. Conversions can be initiated while the MCU is in stop 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 stop mode if the ADC interrupt is enabled (AIEN = 1).

#### NOTE

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, your application must ensure that the data transfer blocking mechanism (discussed in [Section 12.4.4.2, "Completing Conversions"](#)) is cleared when entering stop and continuing ADC conversions.

## 12.5 Initialization Information

This section gives basic direction on how to initialize and configure the ADC module. Among other options, you can configure the module for:

- 8-bit or 10-bit resolution
- single or continuous conversion

- polled or interrupt approach

Refer to [Table 12-6](#), [Table 12-7](#), and [Table 12-8](#) for information used in the following example.

### NOTE

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

## 12.5.1 ADC Module Initialization Example

Before the ADC module completes conversions, an initialization must be performed.

### 12.5.1.1 Initialization Sequence

A typical initialization 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. 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 continuous or once-only conversions and to enable or disable conversion complete interrupts. Also, use this register to select the input channel on which conversions are to be performed.

### 12.5.1.2 Pseudo-Code Example

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

**ADCCFG = 0x98 (%10011000)**

Bit 7	ADLPC	1	Configures for low power (lowers maximum clock speed)
Bit 6:5	ADIV	00	Sets ADCK to 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	Indicates if a conversion is in progress
Bit 6	ADTRG	0	Selects software trigger
Bit 5	ACFE	0	Disables compare function
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 internal use; always write zero

**ADCSC1 = 0x41 (%01000001)**

Bit 7	COCO	0	Indicates when a conversion completes
Bit 6	AIEN	1	Enable conversion complete interrupt
Bit 5	ADCO	0	Specifies one conversion only (continuous conversions disabled)
Bit 4:0	ADCH	00001	Selects 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

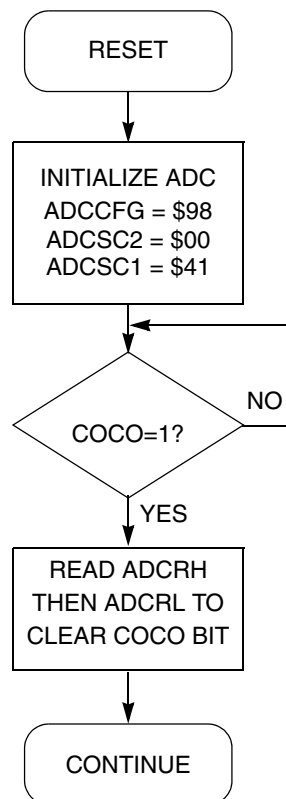


Figure 12-14. Initialization Flowchart Example

## 12.6 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.

## 12.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how to use them for best results.

### 12.6.1.1 Analog Supply Pins

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

When available on a separate pin,  $V_{DDAD}$  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 with 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. Typically this must be the only ground connection between these supplies if possible. The  $V_{SSAD}$  pin makes a good, single-point ground location.

### 12.6.1.2 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 can be shared on the same pin as  $V_{DDAD}$  on some devices. The low reference is  $V_{REFL}$ , which can be shared on the same pin as  $V_{SSAD}$  on some devices.

When available on a separate pin,  $V_{REFH}$  can be connected to the same potential as  $V_{DDAD}$ , or can 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  $\mu\text{F}$  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).

### 12.6.1.3 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 must 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  $\mu\text{F}$  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 must not be transitioning during conversions.

## 12.6.2 Sources of Error

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

### 12.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7k $\Omega$  and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (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 below 5 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.

### 12.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause a 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 1/4LSB leakage error ( $N = 8$  in 8-bit mode or 10 in 10-bit mode).

### 12.6.2.3 Noise-Induced Errors

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

- There is a 0.1  $\mu\text{F}$  low-ESR capacitor from  $V_{REFH}$  to  $V_{REFL}$ .
- There is a 0.1  $\mu\text{F}$  low-ESR capacitor from  $V_{DDAD}$  to  $V_{SSAD}$ .
- If inductive isolation is used from the primary supply, an additional 1  $\mu\text{F}$  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 stop 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 stop mode operation, select ADACK as the clock source. Operation in stop 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.

In 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 stop or I/O activity can not be halted, try the following recommended actions to reduce the effect of noise on the accuracy:

- Place a 0.01  $\mu\text{F}$  capacitor ( $C_{AS}$ ) on the selected input channel to  $V_{REFL}$  or  $V_{SSAD}$  (this improves noise issues, but can affect sample rate based upon 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.

#### 12.6.2.4 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:

$$1\text{LSB} = (V_{REFH} - V_{REFL}) / 2^N \quad \text{Eqn. 12-2}$$

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

#### 12.6.2.5 Linearity Errors

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

- Zero-scale error ( $E_{ZS}$ ) (sometimes called offset) — 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 ( $1\text{LSB}$ ) is used.
- Full-scale error ( $E_{FS}$ ) — The difference between the actual code width of the last conversion and the ideal code width ( $1.5\text{LSB}$ ). Note, if the last conversion is \$3FE, then the difference between the actual \$3FE code width and its ideal ( $1\text{LSB}$ ) is used.

- Differential non-linearity (DNL) — The worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — The highest-value (the absolute value) of the running sum 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) — The difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

### 12.6.2.6 Code Jitter, Non-Monotonicity and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error;

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 increases with noise. Repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 12.6.2.3](#) 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 values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is monotonic and has no missing codes.





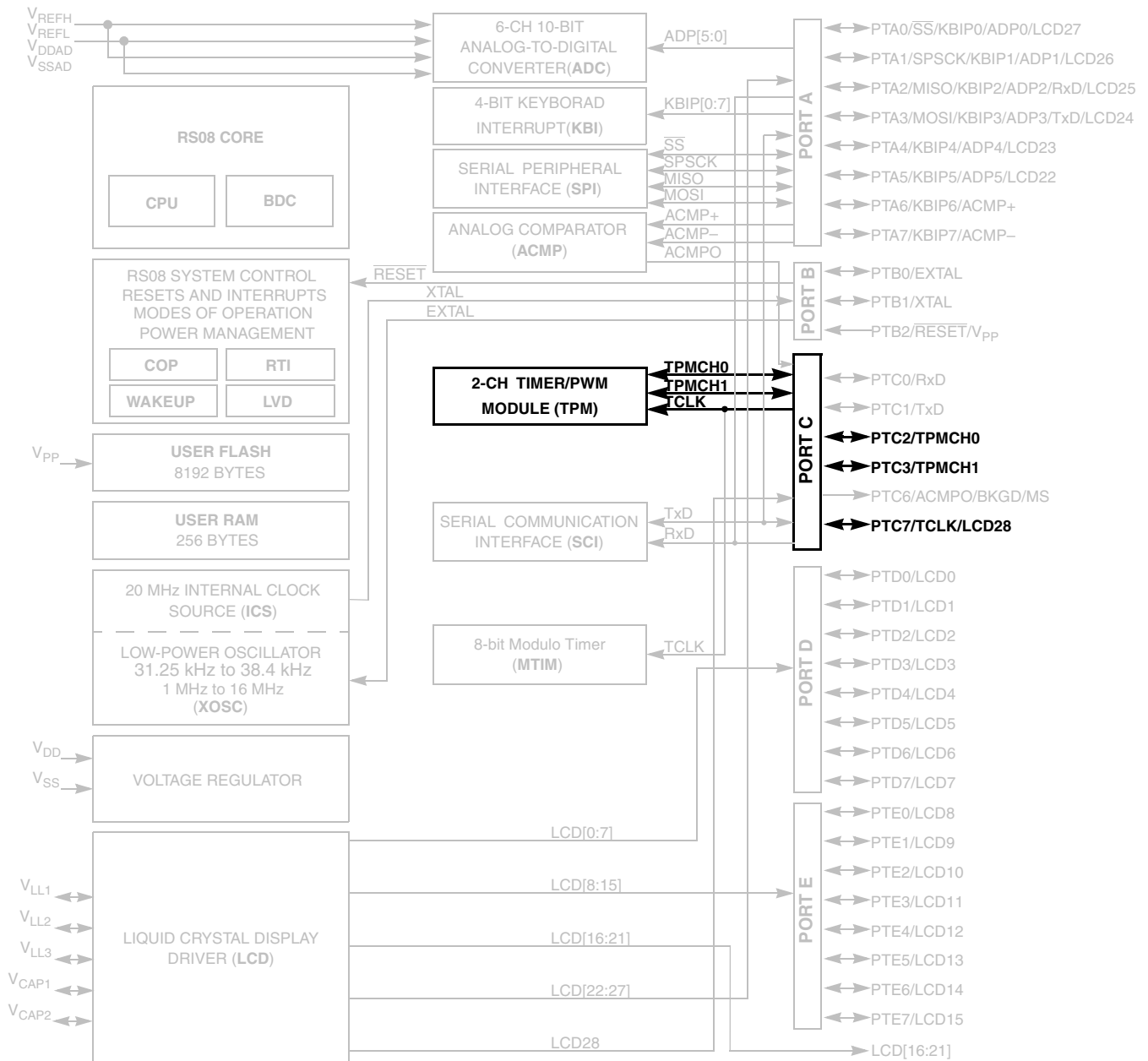
---

## Chapter 13

### 16-Bit Timer/PWM (RS08TPMV2)

#### 13.1 Introduction

[Figure 13-1](#) shows the MC9RS08LA8 block diagram highlighting the TPM block and pins.



**NOTES:**

1. PTB2/RESET/ $V_{PP}$  is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 13-1. MC9RS08LA8 Series Block Diagram Highlighting TPM Block and Pins**

### 13.1.1 Features

The TPM includes these distinctive features:

- One to eight channels:
  - Each channel may be input capture, output compare, or edge-aligned PWM
  - Rising-Edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
  - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
  - Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
  - External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

### 13.1.2 Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- Input capture mode
 

When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- Output compare mode
 

When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

- Edge-aligned PWM mode

The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.

- Center-aligned PWM mode

Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

### 13.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMCH<sub>n</sub> (timer channel *n*) where *n* is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 13-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMMODH:TPMMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMCNT counter resets the counter, regardless of the data value written.

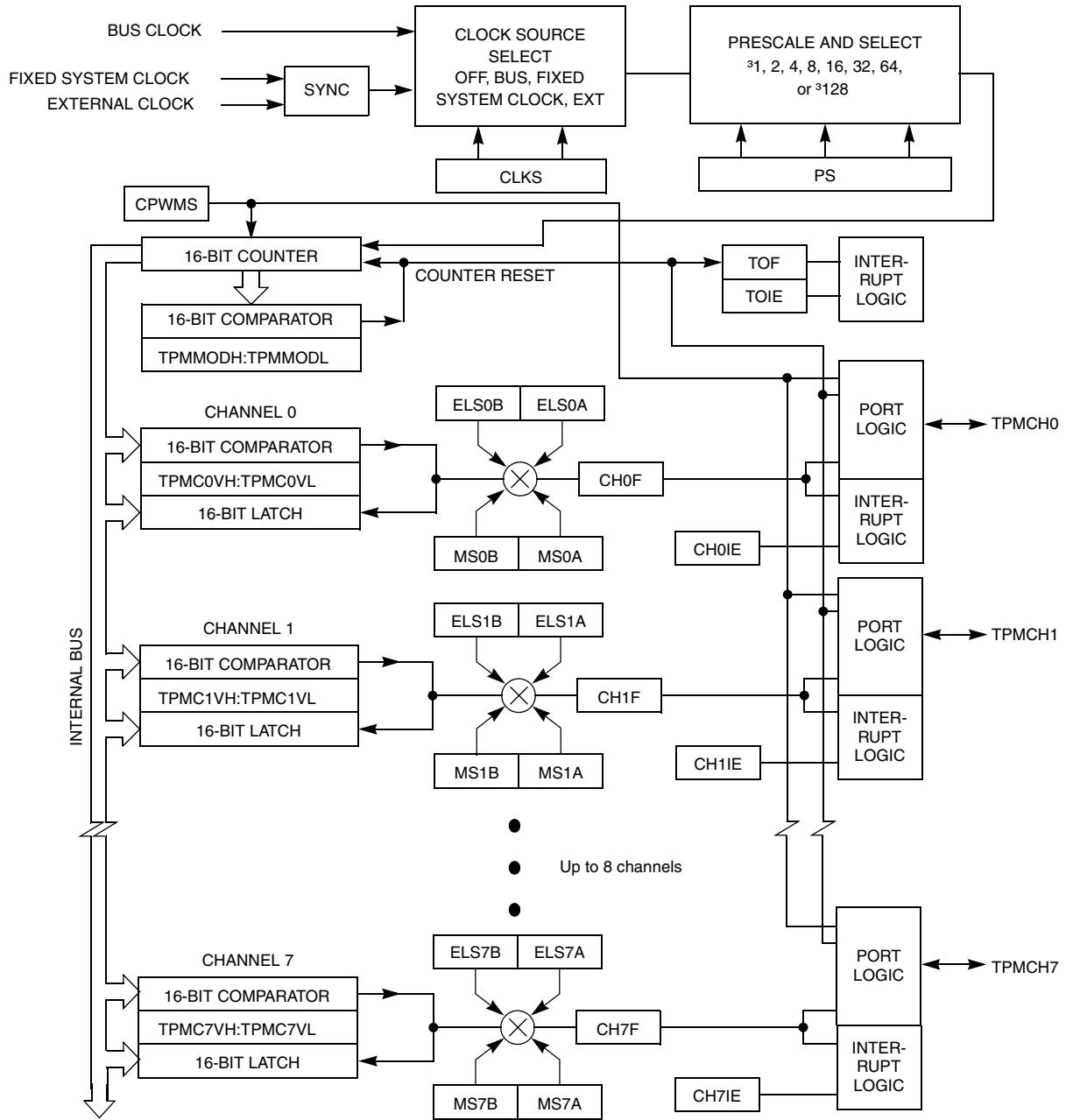


Figure 13-2. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

## 13.2 Signal Description

Table 13-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

**Table 13-1. Signal Properties**

Name	Function
EXTCLK <sup>1</sup>	External clock source which may be selected to drive the TPM counter.
TPMCHn <sup>2</sup>	I/O pin associated with TPM channel n

<sup>1</sup> When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

<sup>2</sup> n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

### 13.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 13-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

### 13.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

### 13.2.1.2 TPMCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKS = 00) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

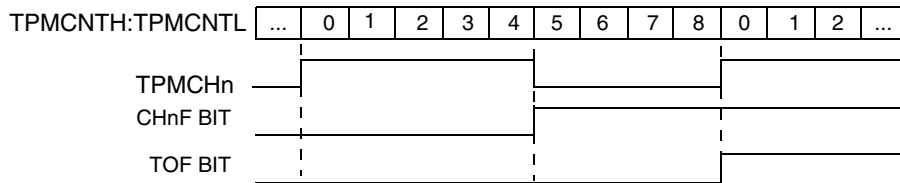
When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

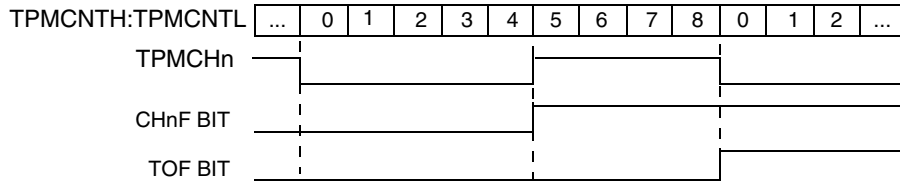
When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMCHn pin is forced high at the start of each new period (TPMCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMCHn pin is forced low at the start of each new period (TPMCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

TPMMODH:TPMMODL = 0x0008  
 TPMMODH:TPMMODL = 0x0005



**Figure 13-3. High-True Pulse of an Edge-Aligned PWM**

TPMMODH:TPMMODL = 0x0008  
 TPMMODH:TPMMODL = 0x0005



**Figure 13-4. Low-True Pulse of an Edge-Aligned PWM**



When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

TPMMODH:TPMMODL = 0x0008  
 TPMMODH:TPMMODL = 0x0005

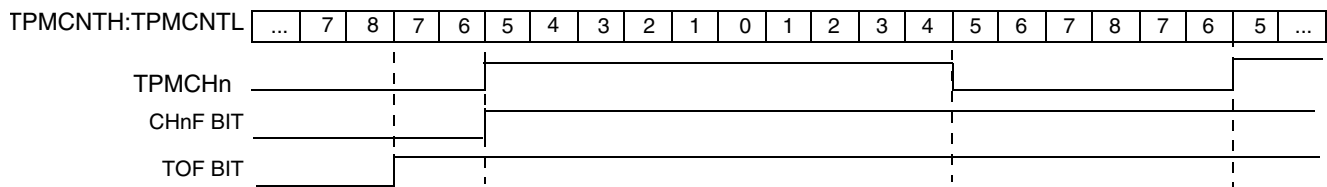


Figure 13-5. High-True Pulse of a Center-Aligned PWM

TPMMODH:TPMMODL = 0x0008  
 TPMMODH:TPMMODL = 0x0005

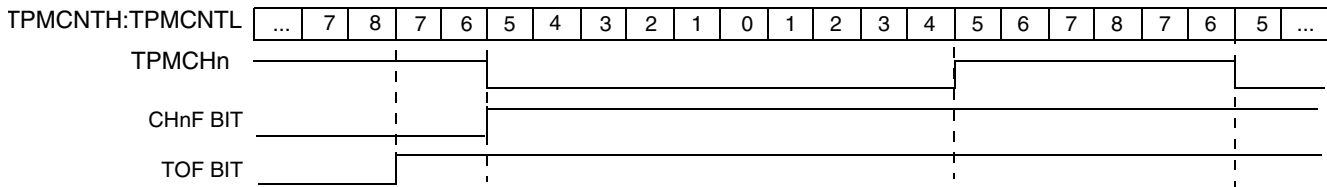


Figure 13-6. Low-True Pulse of a Center-Aligned PWM

## 13.3 Register Definition

This section consists of register descriptions in address order.

### 13.3.1 TPM Status and Control Register (TPMSC)

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

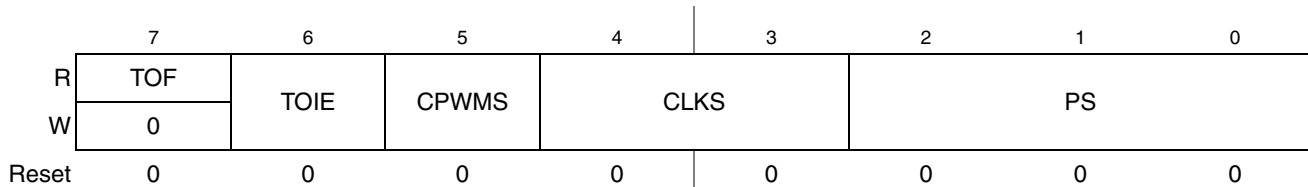


Figure 13-7. TPM Status and Control Register (TPMSC)

Table 13-2. TPMSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 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 one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.
4–3 CLKS	Clock source selects. As shown in <a href="#">Table 13-3</a> , this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based system clock. When there is no PLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in <a href="#">Table 13-4</a> . This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.

**Table 13-3. TPM-Clock-Source Selection**

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

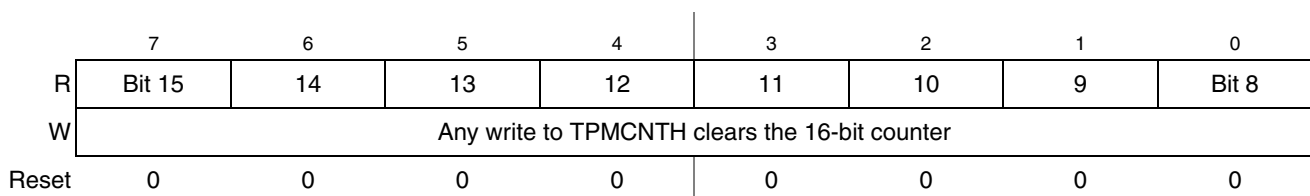
**Table 13-4. Prescale Factor Selection**

PS	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

### 13.3.2 TPM-Counter Registers (TPMCNTH:TPMCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMCNTH or TPMCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMSC).

Reset clears the TPM counter registers. Writing any value to TPMCNTH or TPMCNTL also clears the TPM counter (TPMCNTH:TPMCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

**Figure 13-8. TPM Counter Register High (TPMCNTH)**

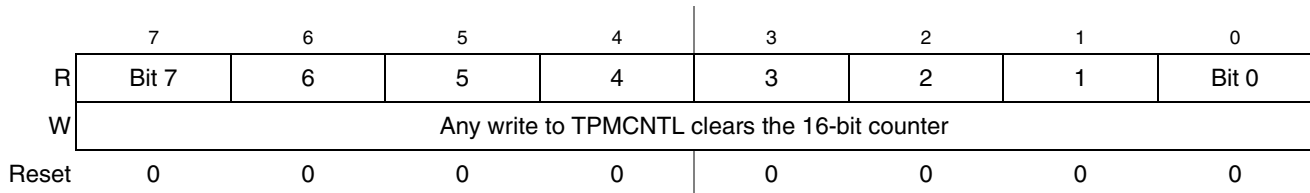


Figure 13-9. TPM Counter Register Low (TPMCNTL)

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

### 13.3.3 TPM Counter Modulo Registers (TPMMODH:TPMMODL)

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

Writing to either byte (TPMMODH or TPMMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), then the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMSA address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

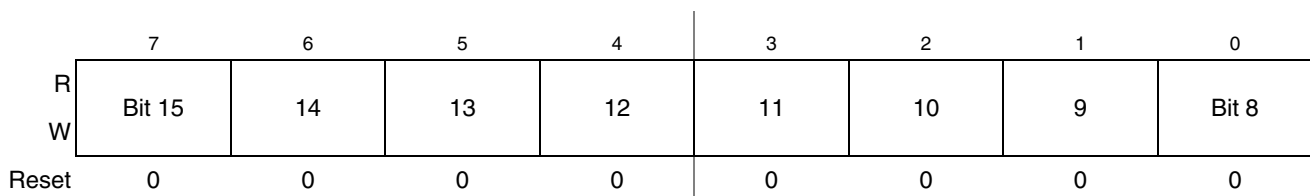
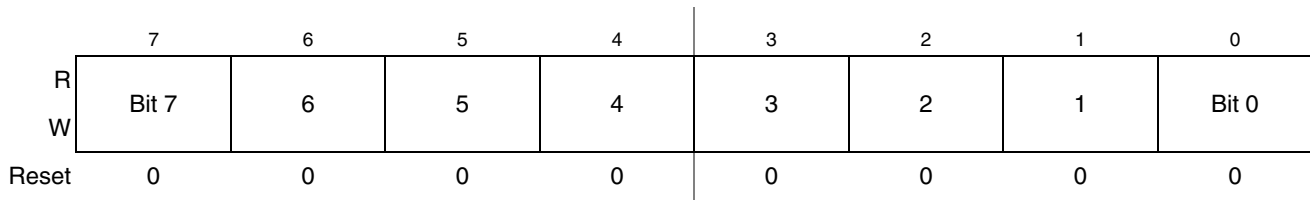


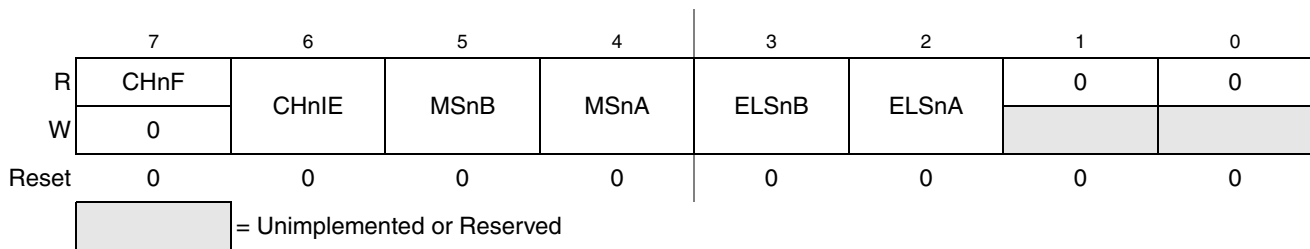
Figure 13-10. TPM Counter Modulo Register High (TPMMODH)



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 TPM Channel n Status and Control Register (TPMCnSC)

TPMCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.



**Figure 13-12. TPM Channel n Status and Control Register (TPMCnSC)**

**Table 13-5. TPMCnSC Field Descriptions**

Field	Description
7 CHnF	Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF. Reset clears the CHnF bit. Writing a logic 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 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 for 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. Refer to the summary of channel mode and setup controls in <a href="#">Table 13-6</a> .

Table 13-5. TPMCnSC Field Descriptions (continued)

Field	Description
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 13-6 for a summary of channel mode and setup controls. <b>Note:</b> If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 13-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output. Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 13-6. Mode, Edge, and Level Selection

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

### 13.3.5 TPM Channel Value Registers (TPMCnVH:TPMCnVL)

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

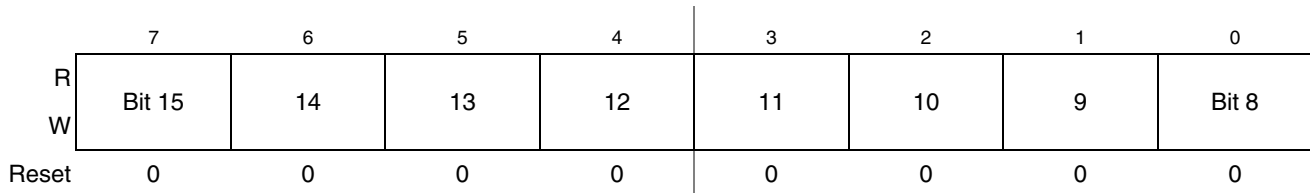


Figure 13-13. TPM Channel Value Register High (TPMCnVH)

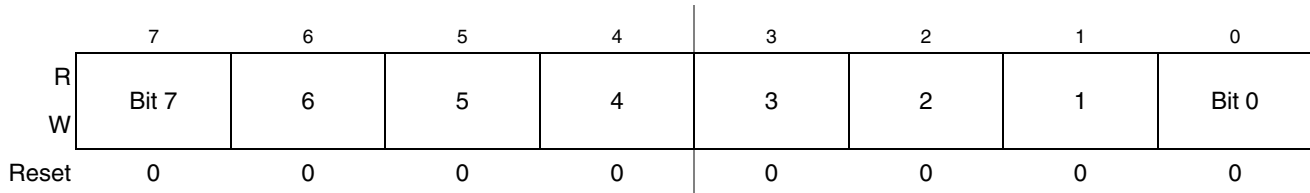


Figure 13-14. TPM Channel Value Register Low (TPMCnVL)

In input capture mode, reading either byte (TPMCnVH or TPMCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMCnVH and TPMCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMCnVH or TPMCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKS bits and the selected mode, so:

- If (CLKS[1:0] = 00), then the registers are updated when the second byte is written.
- If (CLKS[1:0] not = 00 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKS[1:0] not = 00 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel

registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

## 13.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

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

### 13.4.1 Counter

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

#### 13.4.1.1 Counter Clock Source

The 2-bit field, CLKS, in the timer status and control register (TPMSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See [Table 13-3](#). After any MCU reset, CLKS=00 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKS field) does not affect the values in the counter or other timer registers.



**Table 13-7. TPM Clock Source Selection**

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL or the PLL is not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin must not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

### 13.4.1.2 Counter Overflow and Modulo Reset

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

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

### 13.4.1.3 Counting Modes

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

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

### 13.4.1.4 Manual Counter Reset

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

## 13.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

### 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 (TPMCnVH:TPMCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

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

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

### 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 registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

An output compare event sets a flag bit (CHnF) which may 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 value of the modulus register (TPMMODH:TPMMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMCnVH:TPMCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 13-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

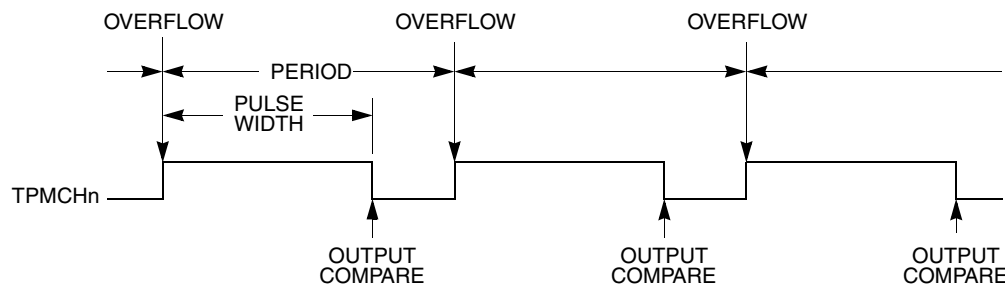


Figure 13-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMCnVH:TPMCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMCnVH and TPMCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFFF.

### 13.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMCnVH:TPMCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMMODH:TPMMODL. TPMMODH:TPMMODL must 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.

$$\text{pulse width} = 2 \times (\text{TPMCnVH:TPMCnVL})$$

$$\text{period} = 2 \times (\text{TPMMODH:TPMMODL}); \text{TPMMODH:TPMMODL}=0\text{x}0001\text{-}0\text{x}7\text{FFF}$$

If the channel-value register TPMCnVH:TPMCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMCnVH:TPMCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

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

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 13-16). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMMODH:TPMMODL, then counts down until it reaches zero. This sets the period equal to two times TPMMODH:TPMMODL.

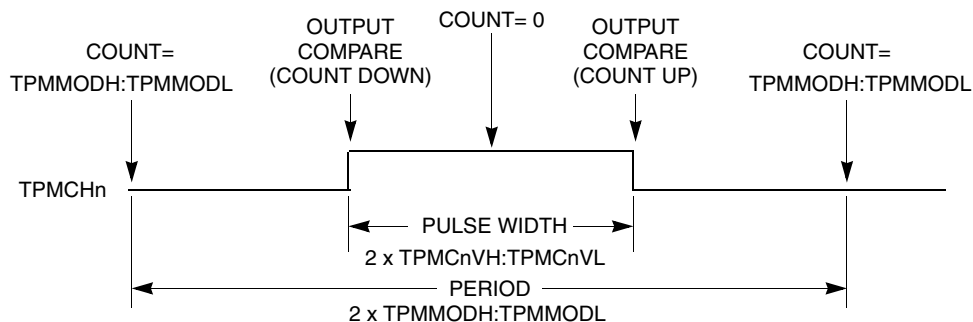


Figure 13-16. CPWM Period and Pulse Width (ELSnA=0)

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

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMMODH, TPMMODL, TPMCnVH, and TPMCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMCnVH:L registers are updated with the value of their write buffer according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

When TPMCNTH:TPMCNTL=TPMMODH:TPMMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to TPMSC cancels any values written to TPMMODH and/or TPMMODL and resets the coherency mechanism for the modulo registers. Writing to TPMCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMCnVH:TPMCnVL.

## 13.5 Reset Overview

### 13.5.1 General

The TPM is reset whenever any MCU reset occurs.

### 13.5.2 Description of Reset Operation

Reset clears the TPMSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

## 13.6 Interrupts

### 13.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in [Table 13-8](#) which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

**Table 13-8. Interrupt Summary**

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user's guide for the interrupt module or to the chip's complete documentation for details.

## 13.6.2 Description of Interrupt Operation

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

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

### 13.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

#### 13.6.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

#### 13.6.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

## 13.6.2.2 Channel Event Interrupt Description

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

### 13.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in [Section 13.6.2, "Description of Interrupt Operation."](#)

### 13.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described [Section 13.6.2, "Description of Interrupt Operation."](#)

### 13.6.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described [Section 13.6.2, "Description of Interrupt Operation."](#)





---

## Chapter 14

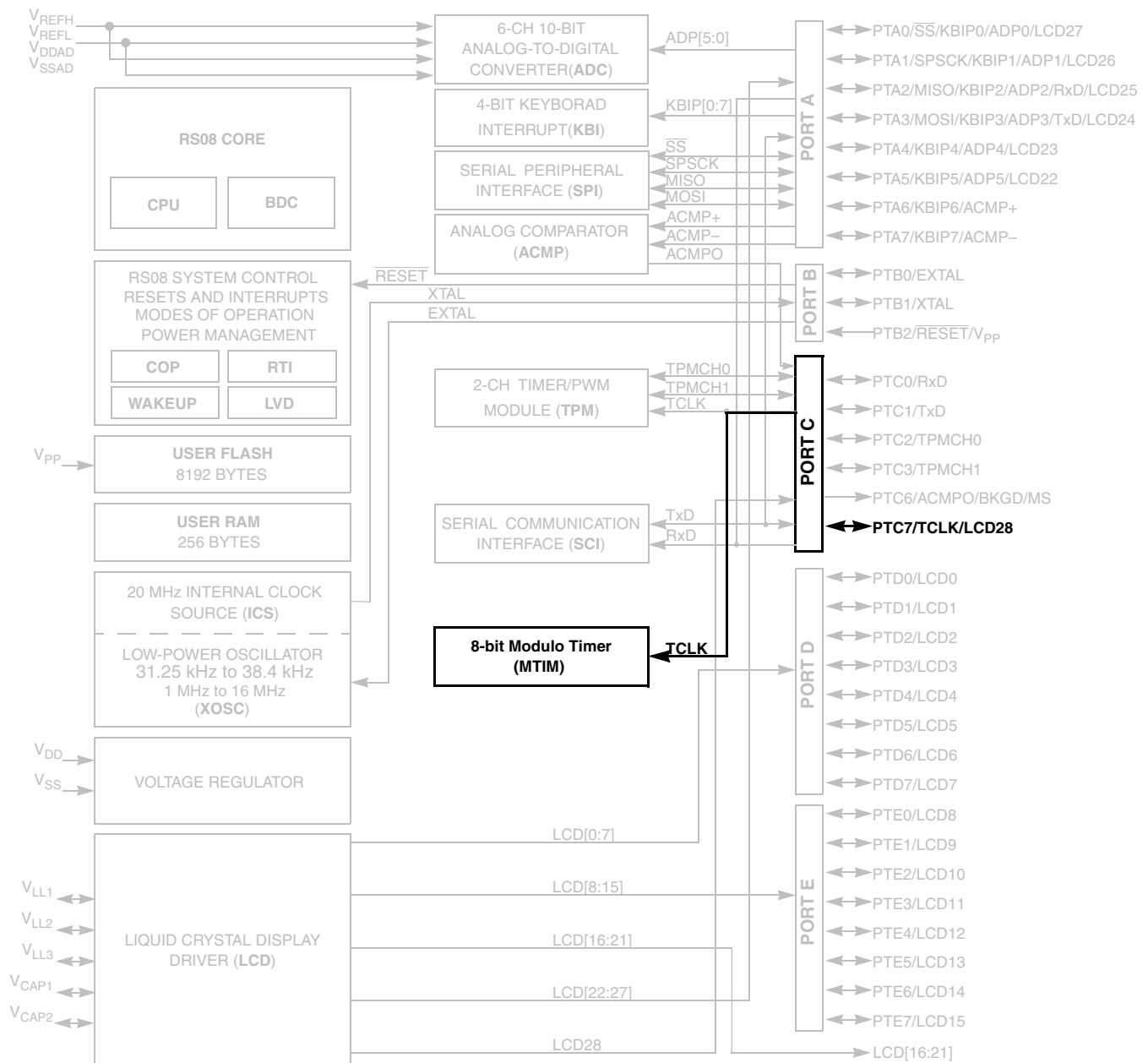
# Modulo Timer (RS08MTIMV1)

### 14.1 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 that 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 14-1](#) shows the MC9RS08LA8 block diagram highlighting the MTIM block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 14-1. MC9RS08LA8 Series Block Diagram Highlighting MTIM Block and Pin**

## 14.1.1 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

## 14.1.2 Modes of Operation

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

### 14.1.2.1 Operation 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 must be disabled by software if not needed as an interrupt source during wait mode.

### 14.1.2.2 Operation 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 mode.

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

### 14.1.2.3 Operation in Active Background Mode

The MTIM suspends all counting until the MCU 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 any value is written to the MTIMMOD register).

### 14.1.3 Block Diagram

The block diagram for the modulo timer module is shown [Figure 14-2](#).

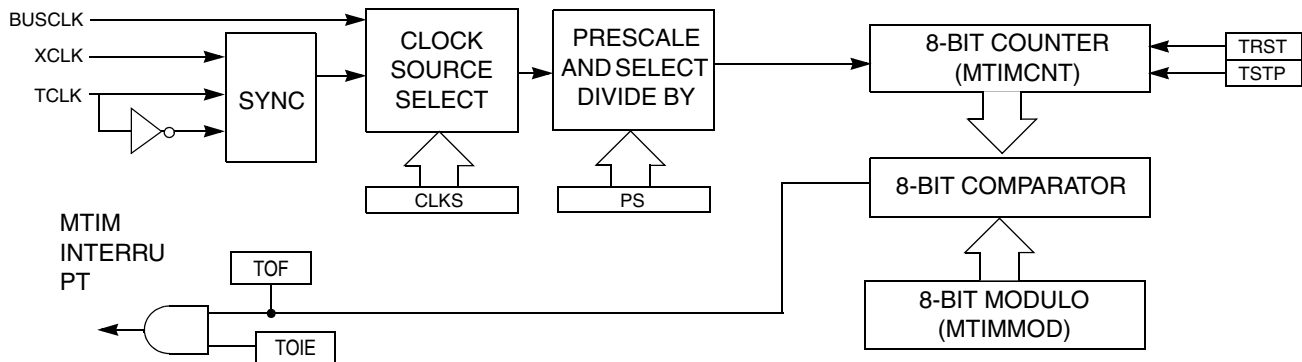


Figure 14-2. Modulo Timer (MTIM) Block Diagram

## 14.2 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 14-1](#).

Table 14-1. 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.

## 14.3 Register Definition

Each MTIM includes four registers, which are summarized in [Table 14-2](#):

- 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 section 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.

Table 14-2. MTIM Register Summary

Name		7	6	5	4	3	2	1	0
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMCLK	R	0	0	CLKS		PS			
	W								
MTIMCNT	R	COUNT							
	W								
MTIMMOD	R	MOD							
	W								

### 14.3.1 MTIM Status and Control Register (MTIMSC)

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

	7	6	5	4	3	2	1	0
R	TOF	TOIE	0	TSTP	0	0	0	0
W			TRST					
Reset:	0	0	0	1	0	0	0	0

Figure 14-3. MTIM Status and Control Register (MTIMSC)

Table 14-3. MTIMSC Field Descriptions

Field	Description
7 TOF	<b>MTIM Overflow Flag</b> — 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 MTIMSC 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 MTIMMOD 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	<b>MTIM Overflow Interrupt Enable</b> — 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.

Table 14-3. MTIMSC Field Descriptions (continued)

Field	Description
5 TRST	<b>MTIM Counter Reset</b> — 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	<b>MTIM Counter Stop</b> — 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.

### 14.3.2 MTIM Clock Configuration Register (MTIMCLK)

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

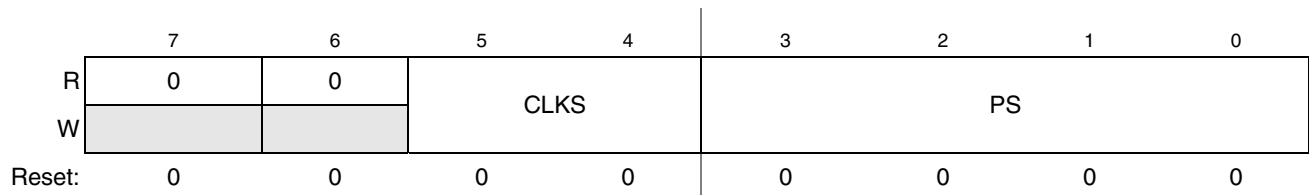


Figure 14-4. MTIM Clock Configuration Register (MTIMCLK)

Table 14-4. MTIMCLK Field Description

Field	Description
5:4 CLKS	<b>Clock Source Select</b> — 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 00. 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.
3:0 PS	<b>Clock Source Prescaler</b> — 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.

### 14.3.3 MTIM Counter Register (MTIMCNT)

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

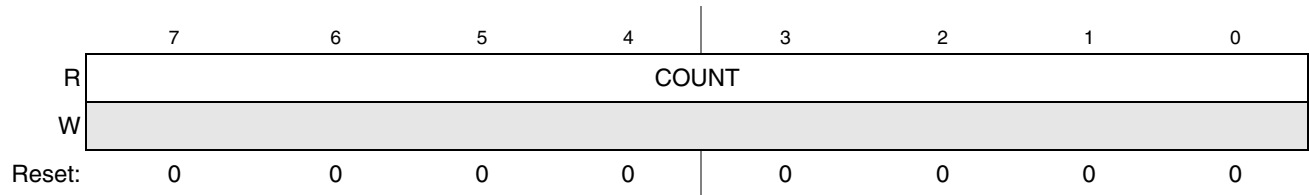


Figure 14-5. MTIM Counter Register (MTIMCNT)

Table 14-5. MTIMCNT Field Description

Field	Description
7:0 COUNT	<b>MTIM Count</b> — 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.

### 14.3.4 MTIM Modulo Register (MTIMMOD)

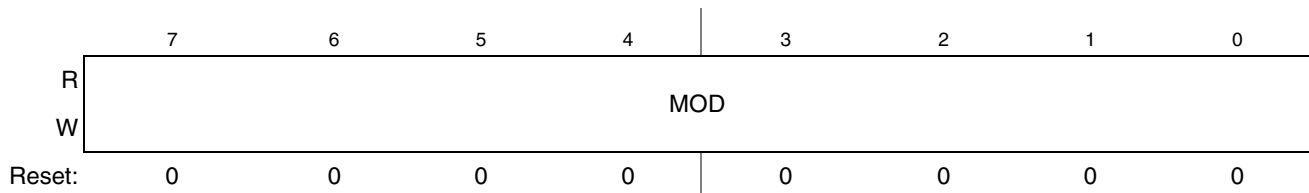


Figure 14-6. MTIM Modulo Register (MTIMMOD)

Table 14-6. MTIMMOD Descriptions

Field	Description
7:0 MOD	<b>MTIM Modulo</b> — 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 MTIMMOD resets the COUNT to \$00 and clears TOF. Reset sets the modulo to \$00.

## 14.4 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 (MTIMCNT) 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 (MTIMSC) 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 (CLKS) in MTIMCLK 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) in MTIMCLK 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 (MTIMMOD) 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 MTIMMOD while the counter is active resets the counter to \$00 and clears TOF.

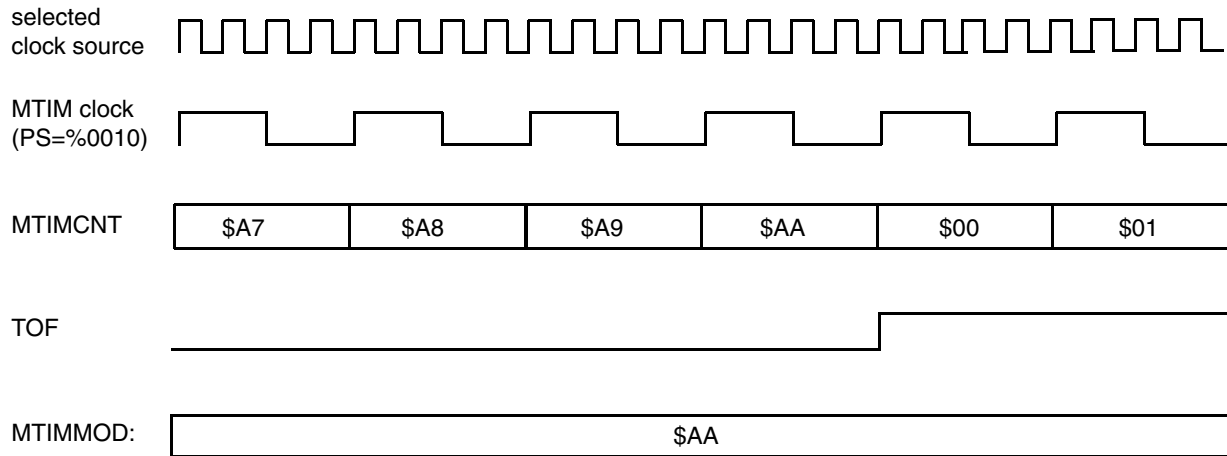
Clearing TOF is a two-step process. The first step is to read the MTIMSC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF will remain set after the second step is performed. This will 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 MTIMMOD 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 MTIMSC. TOIE must never be written to a 1 while TOF = 1. Instead, TOF must be cleared first, then the TOIE can be set to 1.



## 14.4.1 MTIM Operation Example

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



**Figure 14-7. MTIM Counter Overflow Example**

In the example of [Figure 14-7](#), the selected clock source could be any of the four possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMMOD register is set to \$AA. When the counter, MTIMCNT, 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 15

## Serial Communications Interface (S08SCIV4)

### 15.1 Introduction

MC9RS08LA8 has a 2-channel serial communications interface module (SCI). This SCI can work at two modes, fixed mode or mixed mode. Setting SCIMS bit in SOPT selects the working mode. When set, the SCI works at fixed mode. When clear, the SCI works at mixed mode.

#### 15.1.1 SCI Working In Fixed Mode

In fixed mode, only one channel can work while the other is switched off. The SCICS bit in SOPT registers indicates the channel being used by SCI. When set, the SCI uses PTC0 as receive channel and PTC1 as transmit channel. When clear, the SCI uses PTA2 as receive channel and PTA3 as transmit channel.

Figure 15-1 shows the diagram of the fixed mode working flow.

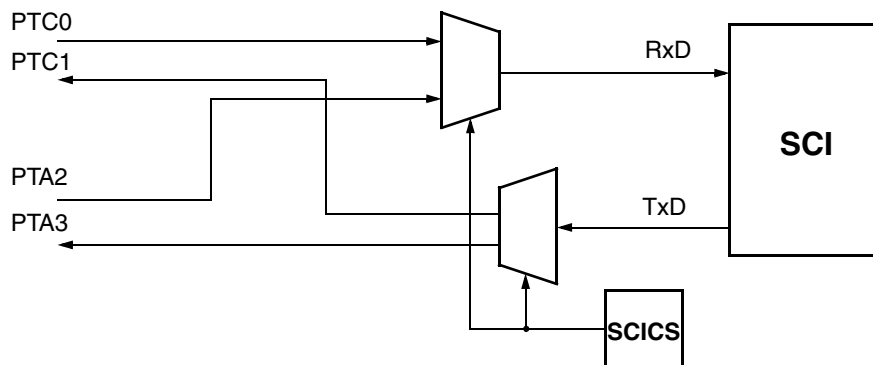


Figure 15-1. Fixed Mode Diagram

#### 15.1.2 SCI Working In Mixed Mode

In mixed mode, both channels share the same SCI. The user need select a certain channel to transmit data by setting SCICS bit in SOPT register and read data from the channel indicated by the SCICS bit in SOPT registers. Reading 0 from this bit indicates a data from PTC0 is received and reading 1 indicates a data from PTA2 is received. Writing 0 to this bit indicates PTC1 is used as transmit channel and writing 1

indicates PTA3 is used as transmit channel. Figure 15-2 shows the diagram of the mixed mode working flow.

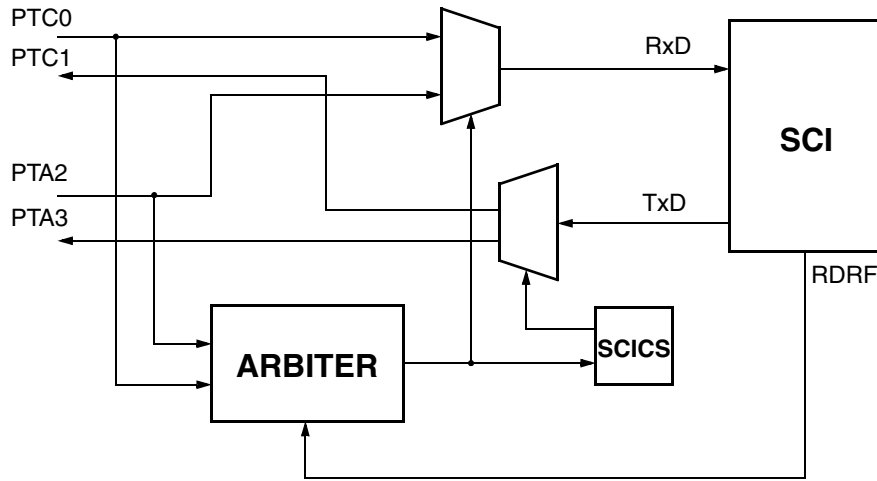


Figure 15-2. Mixed Mode Diagram

In the mixed mode, an arbiter with non-preemption mechanism is used to select the receiving channel. Once the arbiter detects a receiving operation from a certain channel, the other channel will be switched off and SCICS bit in SOPT register is set by arbiter to indicate from this channel the data comes. If data arrives at one channel when the arbiter is receiving data from the other channel, the arbiter will discard the new arrival data. This mechanism ensures the first data will be received correctly if a collision happens. A high level software retransmission is required in collision cases. Once the RDRF in SCIS1 is set, the arbiter will be reset, which means both channels will be switched on again.

**NOTE**

The PTC0 is the default receiving channel of SCI. If two receiving operations occur at the same time exactly, the arbiter will select the PTC0 as the receiving channel.

**15.1.3 SOPT Register Setting**

Table 15-1 summarizes the settings of SCIMS and SCICS bits in SOPT register.

Table 15-1. SOPT Register Setting

Mode	SCIMS	SCICS Action	Port Mapping
Fixed Mode	0	Read/Write: 0	RxD: PTC0 TxD: PTC1
	0	Read/Write: 1	RxD: PTA2 TxD: PTA3

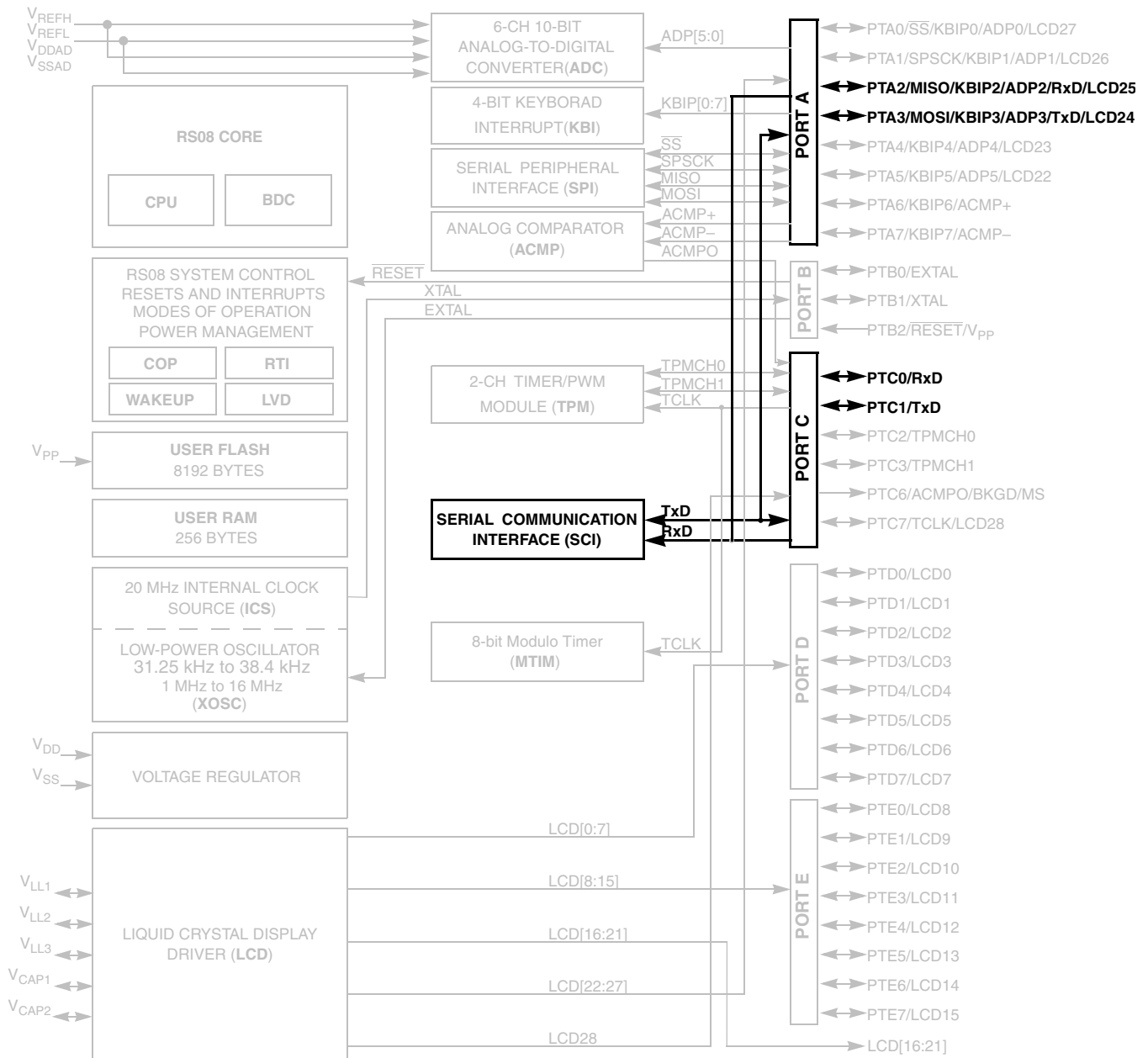
Table 15-1. SOPT Register Setting (continued)

Mode	SCIMS	SCICS Action	Port Mapping
Mixed Mode	1	Read: 0	RxD: PTC0
		Write: 0	TxD: PTC1
	1	Read: 1	RxD: PTA2
		Write: 1	TxD: PTA3

### 15.1.4 Open Drain Mode

In MC9RS08LA8 MCU, PTA3 and PTA2 are shared with both SCI and LCD functionalities. LCD features an open drain mode behavior. To operate SCI properly in open drain mode, external pullup resistors must be used to have a full driver capability when SCI is used. The other way to avoid open drain mode is to set FCDEN bit in LCDC1 register when  $V_{LL3}$  is connected to external power supply. For details about open drain mode, please see [Chapter 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#).

[Figure 15-3](#) shows the MC9RS08LA8 block diagram highlighting the SCI block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 15-3. MC9RS08LA8 Series Block Diagram Highlighting SCI Block and Pins**

## 15.1.5 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

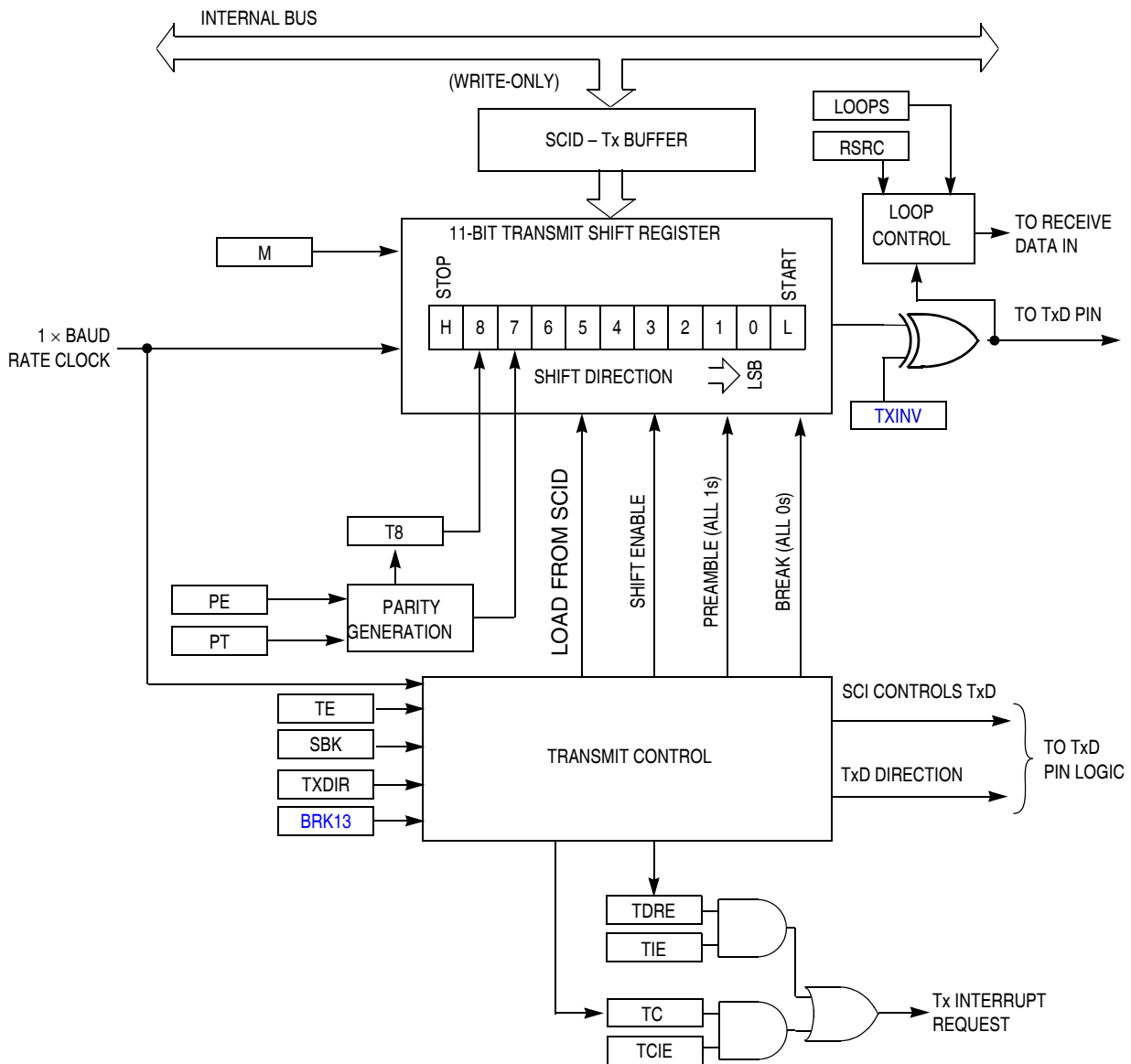
## 15.1.6 Modes of Operation

See [Section 15.3, “Functional Description,”](#) For details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

## 15.1.7 Block Diagram

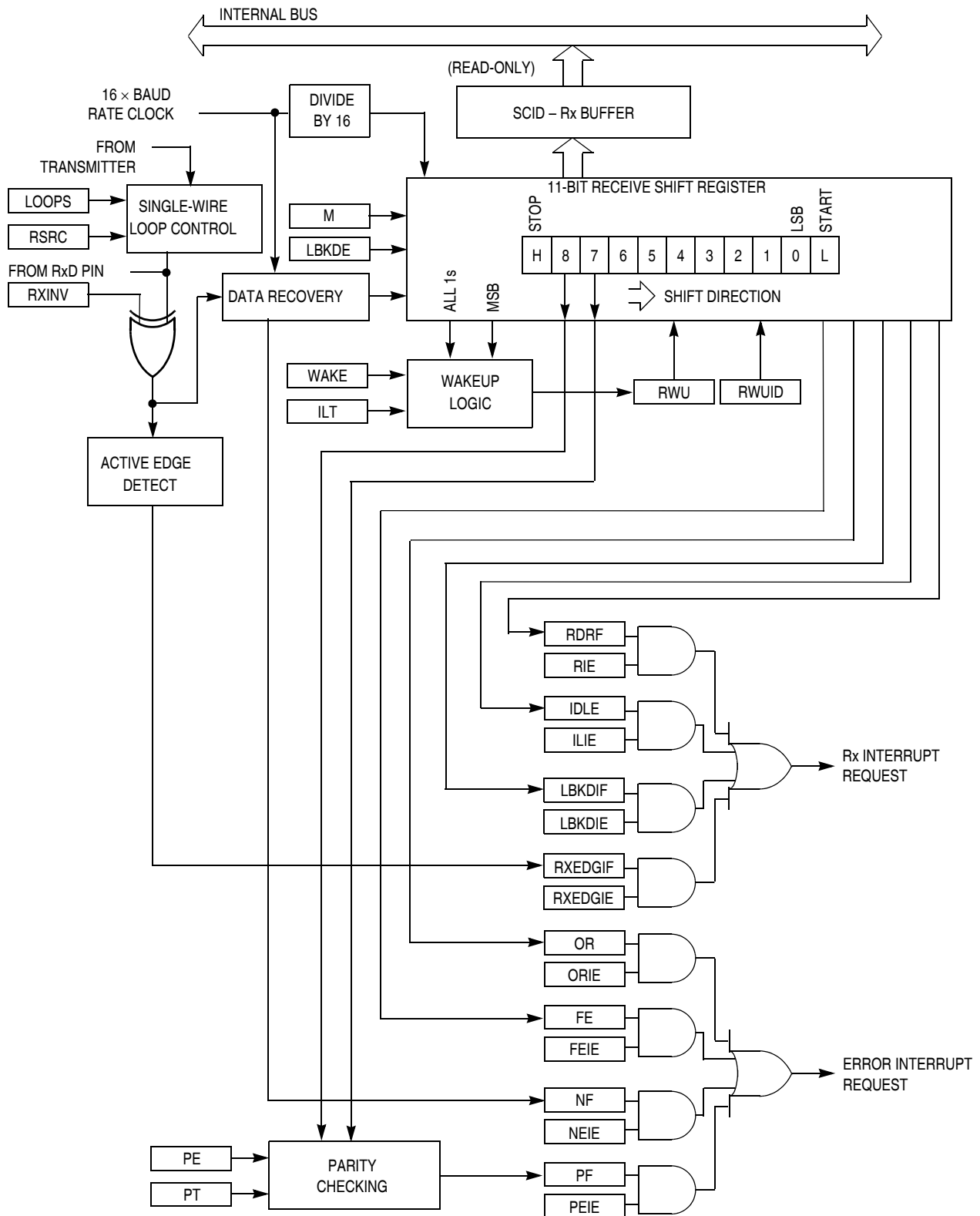
[Figure 15-4](#) shows the transmitter portion of the SCI.



**Figure 15-4. SCI Transmitter Block Diagram**

Figure 15-5 shows the receiver portion of the SCI.





**Figure 15-5. SCI Receiver Block Diagram**

## 15.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 15.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIBDH to buffer the high half of the new value and then write to SCIBDL. The working value in SCIBDH does not change until SCIBDL is written.

SCIBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIC2 are written to 1).

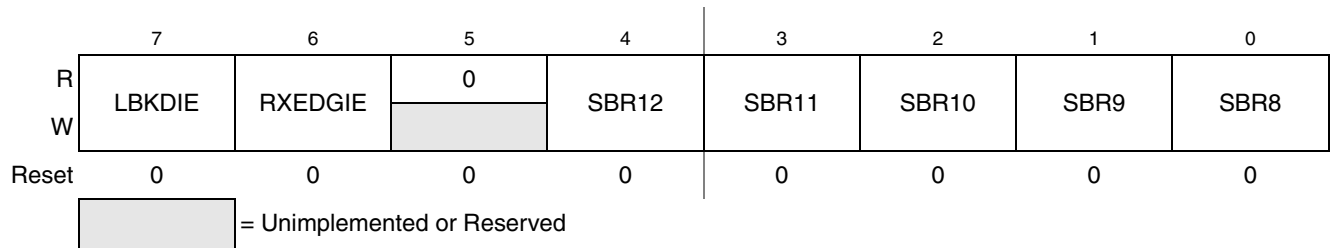


Figure 15-6. SCI Baud Rate Register (SCIBDH)

Table 15-2. SCIBDH Field Descriptions

Field	Description
7 LBKDIE	<b>LIN Break Detect Interrupt Enable (for LBKDIF)</b> 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	<b>RxD Input Active Edge Interrupt Enable (for RXEDGIF)</b> 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	<b>Baud Rate Modulo Divisor</b> — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = $BUSCLK/(16 \times BR)$ . See also BR bits in <a href="#">Table 15-3</a> .

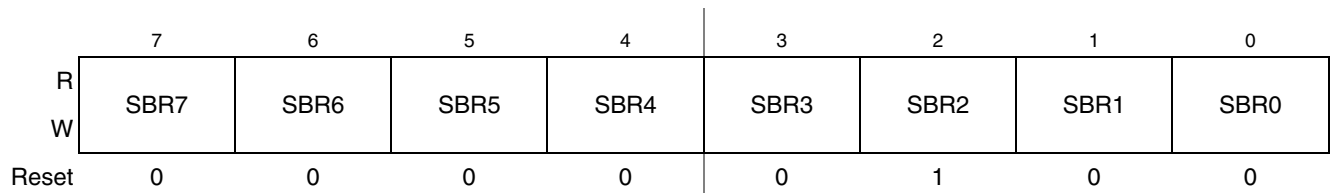


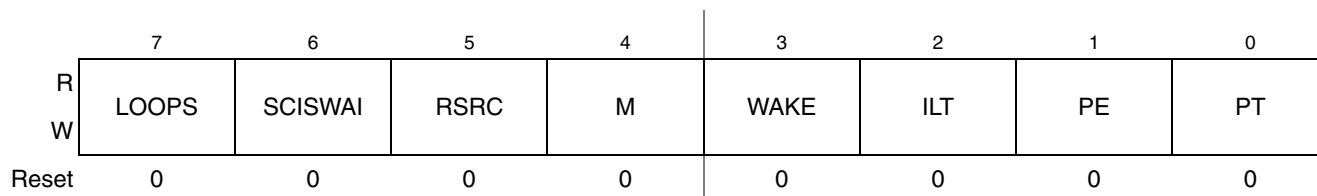
Figure 15-7. SCI Baud Rate Register (SCIBDL)

**Table 15-3. SCIBDL Field Descriptions**

Field	Description
7:0 SBR[7:0]	<b>Baud Rate Modulo Divisor</b> — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in <a href="#">Table 15-2</a> .

## 15.2.2 SCI Control Register 1 (SCIC1)

This read/write register is used to control various optional features of the SCI system.



**Figure 15-8. SCI Control Register 1 (SCIC1)**

**Table 15-4. SCIC1 Field Descriptions**

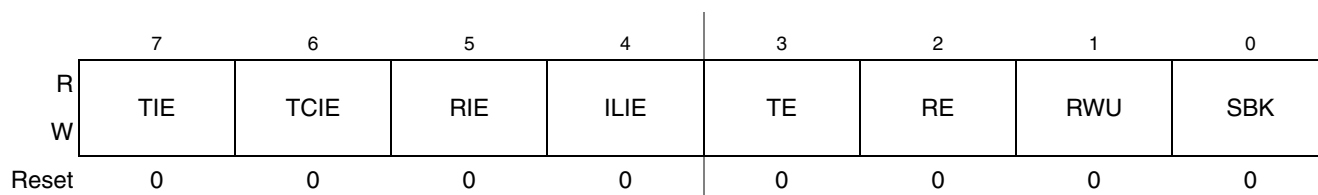
Field	Description
7 LOOPS	<b>Loop Mode Select</b> — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See <a href="#">RSRC</a> bit.) RxD pin is not used by SCI.
6 SCISWAI	<b>SCI Stops in Wait Mode</b> 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	<b>Receiver Source Select</b> — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	<b>9-Bit or 8-Bit Mode Select</b> 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.
3 WAKE	<b>Receiver Wakeup Method Select</b> — Refer to <a href="#">Section 15.3.3.2, “Receiver Wakeup Operation”</a> for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	<b>Idle Line Type Select</b> — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to <a href="#">Section 15.3.3.2.1, “Idle-Line Wakeup”</a> for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

**Table 15-4. SCIC1 Field Descriptions (continued)**

Field	Description
1 PE	<b>Parity Enable</b> — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	<b>Parity Type</b> — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

### 15.2.3 SCI Control Register 2 (SCIC2)

This register can be read or written at any time.



**Figure 15-9. SCI Control Register 2 (SCIC2)**

**Table 15-5. SCIC2 Field Descriptions**

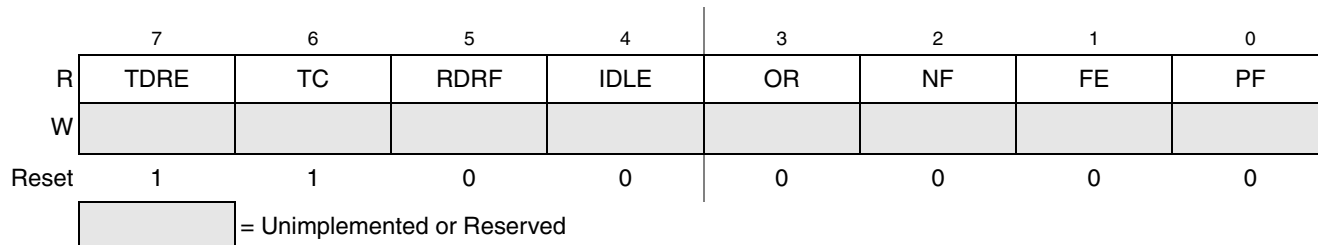
Field	Description
7 TIE	<b>Transmit Interrupt Enable (for TDRE)</b> 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	<b>Transmission Complete Interrupt Enable (for TC)</b> 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	<b>Receiver Interrupt Enable (for RDRF)</b> 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	<b>Idle Line Interrupt Enable (for IDLE)</b> 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.
3 TE	<b>Transmitter Enable</b> 0 Transmitter off. 1 Transmitter on. TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to <a href="#">Section 15.3.2.1, “Send Break and Queued Idle”</a> for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.

**Table 15-5. SCIC2 Field Descriptions (continued)**

Field	Description
2 RE	<b>Receiver Enable</b> — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	<b>Receiver Wakeup Control</b> — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to <a href="#">Section 15.3.3.2, “Receiver Wakeup Operation”</a> for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	<b>Send Break</b> — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to <a href="#">Section 15.3.2.1, “Send Break and Queued Idle”</a> for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

## 15.2.4 SCI Status Register 1 (SCIS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.



**Figure 15-10. SCI Status Register 1 (SCIS1)**

**Table 15-6. SCIS1 Field Descriptions**

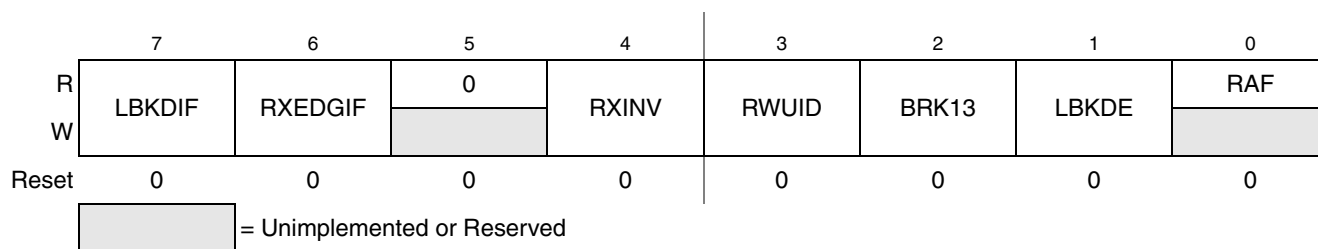
Field	Description
7 TDRE	<p><b>Transmit Data Register Empty Flag</b> — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIS1 with TDRE = 1 and then write to the SCI data register (SCID).</p> <p>0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.</p>
6 TC	<p><b>Transmission Complete Flag</b> — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.</p> <p>0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).</p> <p>TC is cleared automatically by reading SCIS1 with TC = 1 and then doing one of the following three things:</p> <ul style="list-style-type: none"> <li>• Write to the SCI data register (SCID) to transmit new data</li> <li>• Queue a preamble by changing TE from 0 to 1</li> <li>• Queue a break character by writing 1 to SBK in SCIC2</li> </ul>
5 RDRF	<p><b>Receive Data Register Full Flag</b> — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCID). To clear RDRF, read SCIS1 with RDRF = 1 and then read the SCI data register (SCID).</p> <p>0 Receive data register empty. 1 Receive data register full.</p>
4 IDLE	<p><b>Idle Line Flag</b> — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.</p> <p>To clear IDLE, read SCIS1 with IDLE = 1 and then read the SCI data register (SCID). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period.</p> <p>0 No idle line detected. 1 Idle line was detected.</p>
3 OR	<p><b>Receiver Overrun Flag</b> — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCID yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCID. To clear OR, read SCIS1 with OR = 1 and then read the SCI data register (SCID).</p> <p>0 No overrun. 1 Receive overrun (new SCI data lost).</p>
2 NF	<p><b>Noise Flag</b> — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIS1 and then read the SCI data register (SCID).</p> <p>0 No noise detected. 1 Noise detected in the received character in SCID.</p>

**Table 15-6. SCIS1 Field Descriptions (continued)**

Field	Description
1 FE	<b>Framing Error Flag</b> — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIS1 with FE = 1 and then read the SCI data register (SCID). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	<b>Parity Error Flag</b> — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIS1 and then read the SCI data register (SCID). 0 No parity error. 1 Parity error.

## 15.2.5 SCI Status Register 2 (SCIS2)

This register has one read-only status flag.



**Figure 15-11. SCI Status Register 2 (SCIS2)**

**Table 15-7. SCIS2 Field Descriptions**

Field	Description
7 LBKDIF	<b>LIN Break Detect Interrupt Flag</b> — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	<b>RxD Pin Active Edge Interrupt Flag</b> — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV <sup>1</sup>	<b>Receive Data Inversion</b> — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	<b>Receive Wake Up Idle Detect</b> — RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	<b>Break Character Generation Length</b> — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

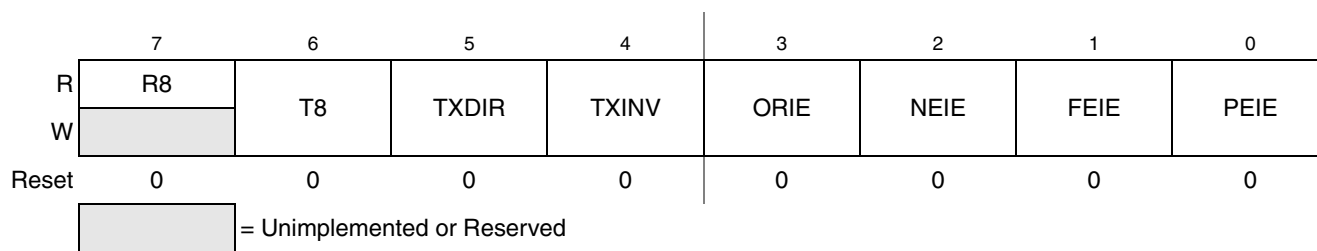
**Table 15-7. SCIS2 Field Descriptions (continued)**

Field	Description
1 LBKDE	<b>LIN Break Detection Enable</b> — LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	<b>Receiver Active Flag</b> — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

<sup>1</sup> Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

## 15.2.6 SCI Control Register 3 (SCIC3)



**Figure 15-12. SCI Control Register 3 (SCIC3)**

**Table 15-8. SCIC3 Field Descriptions**

Field	Description
7 R8	<b>Ninth Data Bit for Receiver</b> — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCID register. When reading 9-bit data, read R8 before reading SCID because reading SCID completes automatic flag clearing sequences which could allow R8 and SCID to be overwritten with new data.
6 T8	<b>Ninth Data Bit for Transmitter</b> — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCID register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCID is written so T8 must be written (if it needs to change from its previous value) before SCID is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCID is written.
5 TXDIR	<b>TxD Pin Direction in Single-Wire Mode</b> — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.



**Table 15-8. SCIC3 Field Descriptions (continued)**

Field	Description
4 TXINV <sup>1</sup>	<b>Transmit Data Inversion</b> — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	<b>Overrun Interrupt Enable</b> — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	<b>Noise Error Interrupt Enable</b> — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	<b>Framing Error Interrupt Enable</b> — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	<b>Parity Error Interrupt Enable</b> — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

<sup>1</sup> Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

## 15.2.7 SCI Data Register (SCID)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

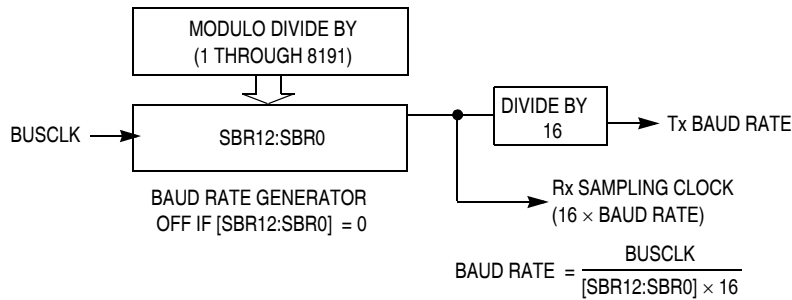
**Figure 15-13. SCI Data Register (SCID)**

## 15.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

### 15.3.1 Baud Rate Generation

As shown in [Figure 15-14](#), the clock source for the SCI baud rate generator is the bus-rate clock.



**Figure 15-14. SCI Baud Rate Generation**

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about  $\pm 4.5$  percent for 8-bit data format and about  $\pm 4$  percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

### 15.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in [Figure 15-4](#).

The transmitter output (TxD) idle state defaults to logic high ( $TXINV = 0$  following reset). The transmitter output is inverted by setting  $TXINV = 1$ . The transmitter is enabled by setting the TE bit in SCIC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCID).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume  $M = 0$ , selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCID.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

### 15.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

**Table 15-9. Break Character Length**

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

### 15.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 15-5) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 15.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF)

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to [Section 15.3.4, "Interrupts and Status Flags"](#) for more details about flag clearing.

### 15.3.3.1 Data Sampling Technique

The SCI receiver uses a  $16\times$  baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The  $16\times$  baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

### 15.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message

characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

#### **15.3.3.2.1 Idle-Line Wakeup**

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

#### **15.3.3.2.2 Address-Mark Wakeup**

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

### **15.3.4 Interrupts and Status Flags**

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCID. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared by reading SCIS1 while  $RDRF = 1$  and then reading SCID.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIS1 while  $IDLE = 1$  and then reading SCID. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled ( $RE = 1$ ).

### 15.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

#### 15.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIC3. For the receiver, the ninth bit is held in R8 in SCIC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCID.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCID to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

### 15.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2. An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software must ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

### 15.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

### 15.3.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.





---

## Chapter 16

# Serial Peripheral Interface (S08SPIV3)

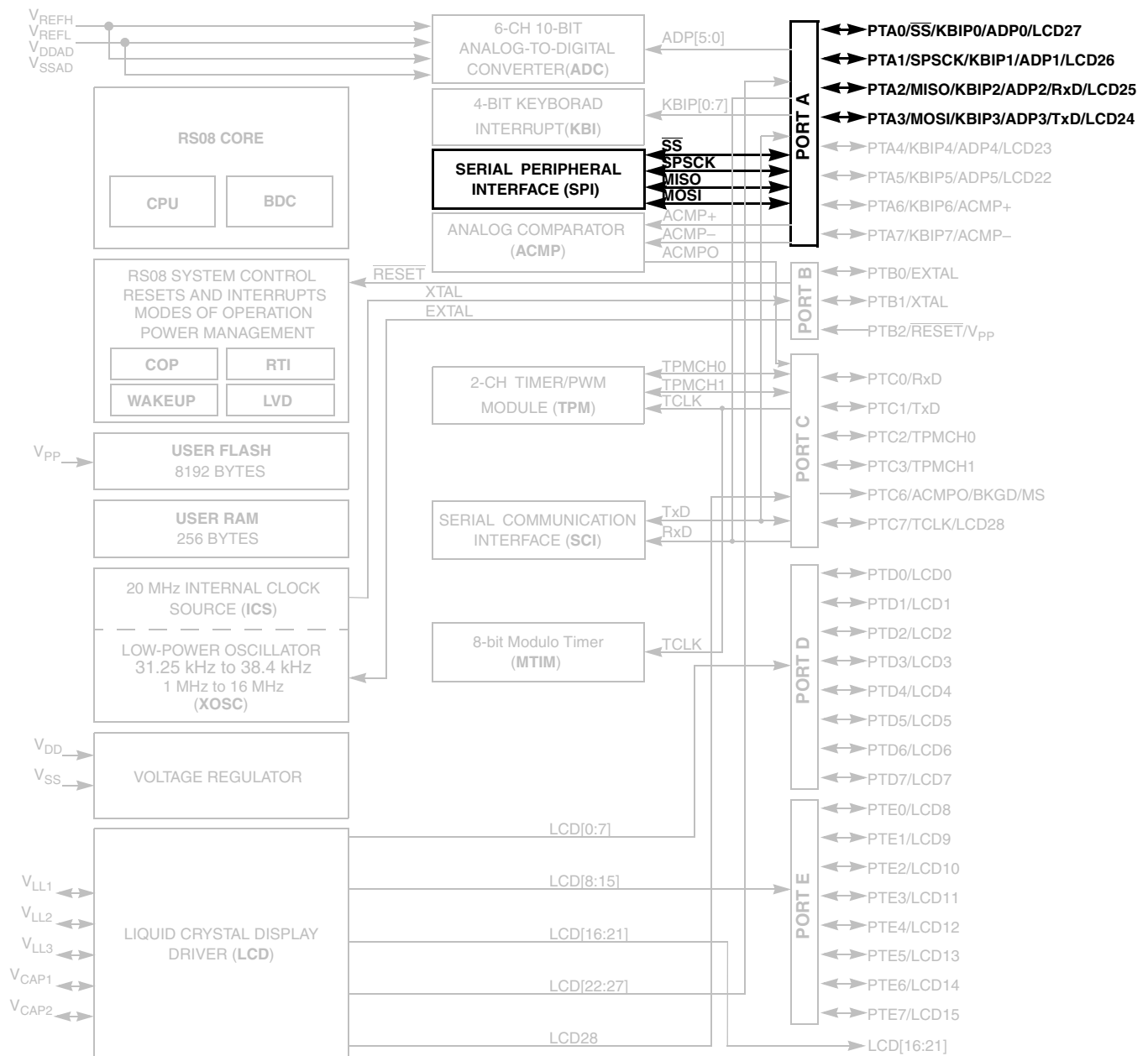
### 16.1 Introduction

MC9RS08LA8 MCU contains a serial peripheral interface module (SPI). The SPI module provides the connection between MC9RS08LA8 and external SPI peripherals.

#### 16.1.1 Open Drain Mode

In MC9RS08LA8 MCU, PTA0, PTA1, PTA2, and PTA3 are shared with both SPI and LCD functionalities. LCD features an open drain mode behavior. To operate SPI properly in open drain mode, external pullup resistors must be used to have a full driver capability when SCI is used. The other way to avoid open drain mode is to set FCDEN bit in LCDC1 register when  $V_{LL3}$  is connected to external power supply. For detailed information about open drain mode, please see [Chapter 10, “Liquid Crystal Display Module \(S08LCDV2\)”](#).

[Figure 16-1](#) shows the MC9RS08LA8 block diagram highlighting the SPI block and pins.



**NOTES:**

1. PTB2/RESET/V<sub>PP</sub> is an input only pin when used as port pin
2. PTC6/ACMPO/BKGD/MS is an output only pin

**Figure 16-1. MC9RS08LA8 Series Block Diagram Highlighting SPI Block and Pins**

## 16.1.2 Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- Programmable transmit bit rate
- Double-buffered transmit and receive
- Serial clock phase and polarity options
- Slave select output
- Selectable MSB-first or LSB-first shifting

## 16.1.3 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

### 16.1.3.1 SPI System Block Diagram

Figure 16-2 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ( $\overline{SS}$  pin). In this system, the master device has configured its  $\overline{SS}$  pin as an optional slave select output.

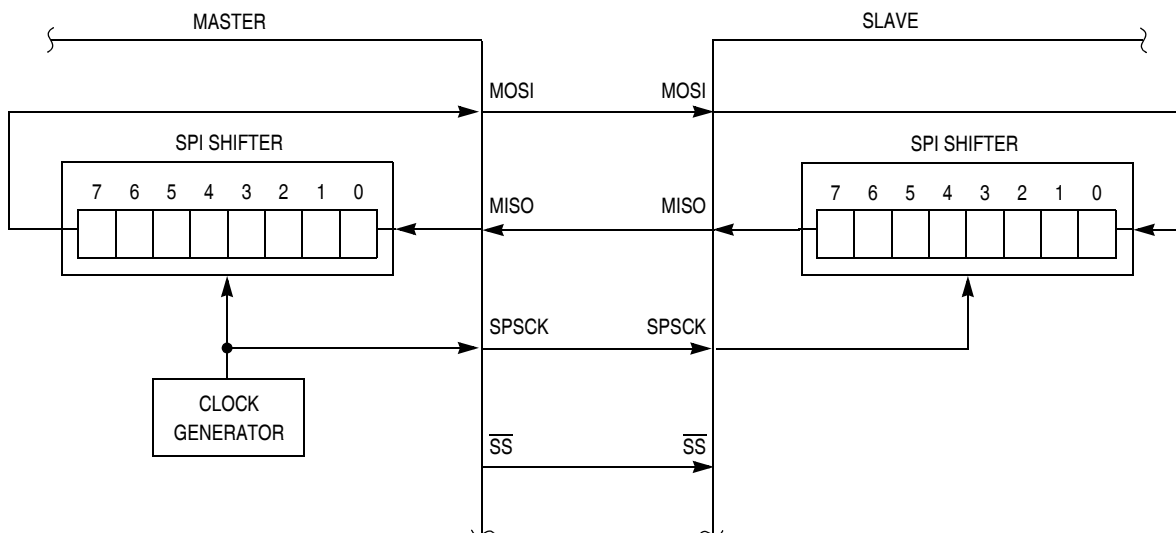


Figure 16-2. SPI System Connections

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although [Figure 16-2](#) shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

### 16.1.3.2 SPI Module Block Diagram

[Figure 16-3](#) is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPID) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

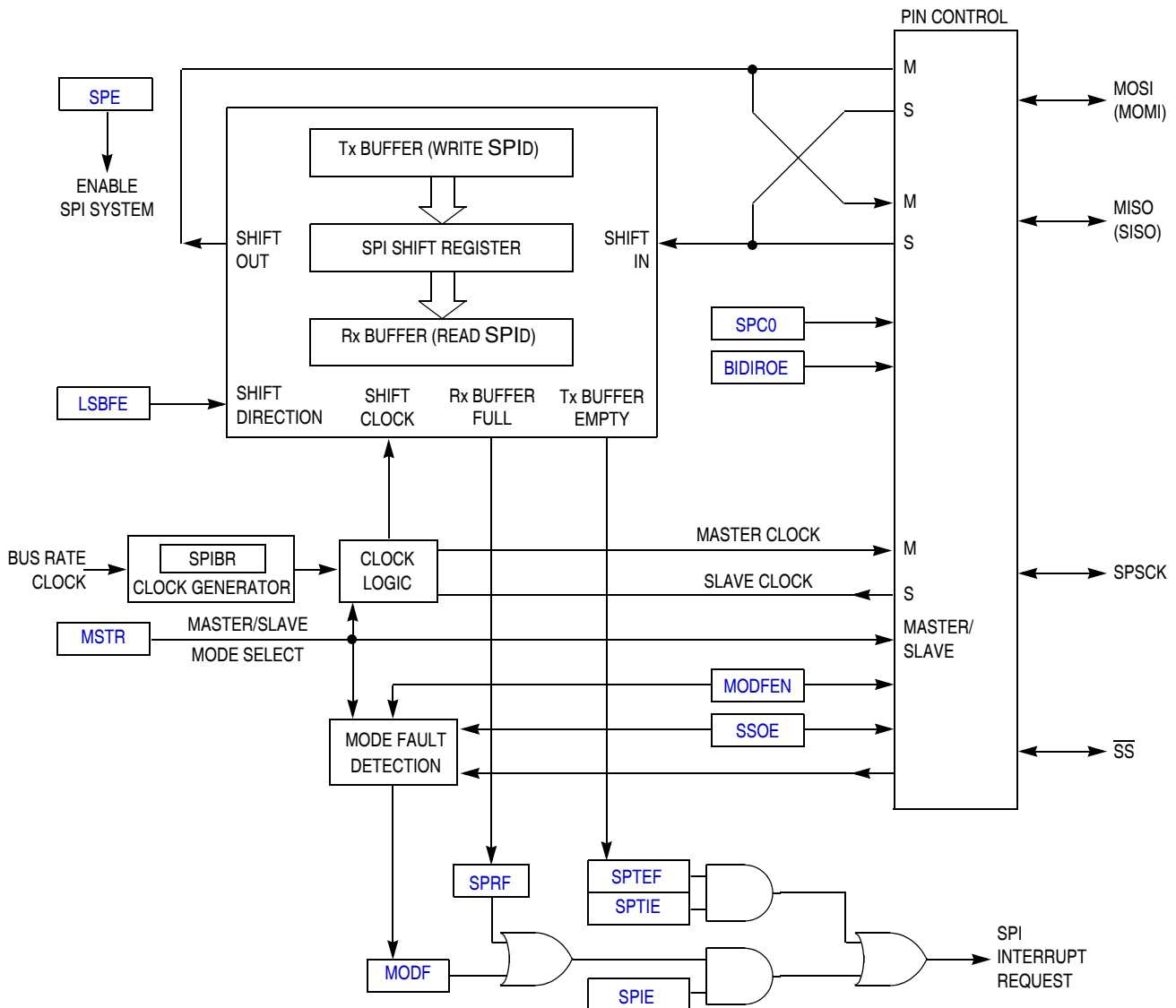


Figure 16-3. SPI Module Block Diagram

### 16.1.4 SPI Baud Rate Generation

As shown in Figure 16-4, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI master mode bit-rate clock.

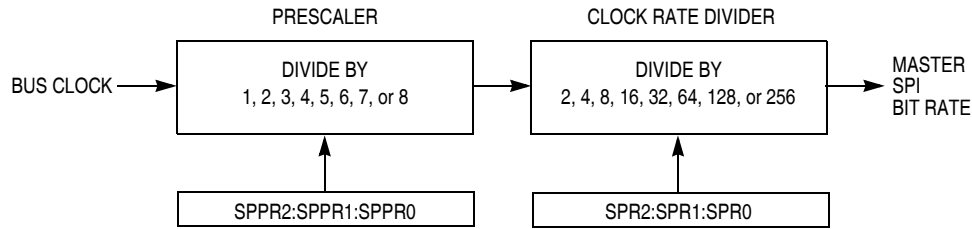


Figure 16-4. SPI Baud Rate Generation

## 16.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled ( $SPE = 0$ ), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 16.2.1 SPCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 16.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero ( $SPC0$ ) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and  $SPC0 = 0$ , this pin is the serial data input. If  $SPC0 = 1$  to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input ( $BIDIROE = 0$ ) or an output ( $BIDIROE = 1$ ). If  $SPC0 = 1$  and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 16.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero ( $SPC0$ ) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and  $SPC0 = 0$ , this pin is the serial data output. If  $SPC0 = 1$  to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input ( $BIDIROE = 0$ ) or an output ( $BIDIROE = 1$ ). If  $SPC0 = 1$  and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 16.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off ( $MODFEN = 0$ ), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and  $MODFEN = 1$ , the slave select output enable bit determines whether this pin acts as the mode fault input ( $SSOE = 0$ ) or as the slave select output ( $SSOE = 1$ ).

## 16.3 Modes of Operation

### 16.3.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

## 16.4 Register Definition

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data.

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 16.4.1 SPI Control Register 1 (SPIC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

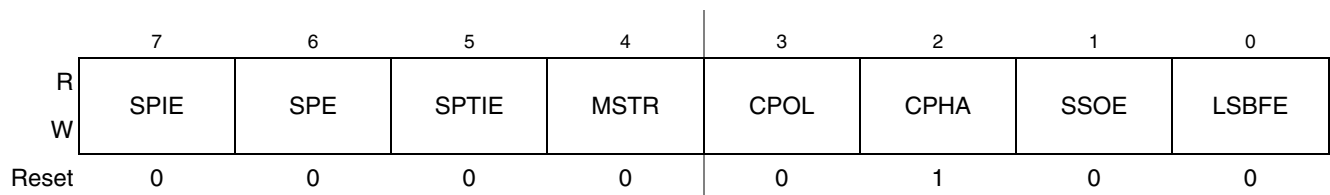


Figure 16-5. SPI Control Register 1 (SPIC1)

Table 16-1. SPIC1 Field Descriptions

Field	Description
7 SPIE	<b>SPI Interrupt Enable (for SPRF and MODF)</b> — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	<b>SPI System Enable</b> — Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	<b>SPI Transmit Interrupt Enable</b> — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested

**Table 16-1. SPIC1 Field Descriptions (continued)**

Field	Description
4 MSTR	<b>Master/Slave Mode Select</b> 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	<b>Clock Polarity</b> — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to <a href="#">Section 16.5.1, “SPI Clock Formats”</a> for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	<b>Clock Phase</b> — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to <a href="#">Section 16.5.1, “SPI Clock Formats”</a> for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer 1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer
1 SSOE	<b>Slave Select Output Enable</b> — This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in <a href="#">Table 16-2</a> .
0 LSBFE	<b>LSB First (Shifter Direction)</b> 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

**Table 16-2.  $\overline{SS}$  Pin Function**

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input


**NOTE**

Ensure that the SPI must not be disabled (SPE=0) at the same time as a bit change to the CPHA bit. These changes must be performed as separate operations or unexpected behavior may occur.

**16.4.2 SPI Control Register 2 (SPIC2)**

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

	7	6	5	4	3	2	1	0
R	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 16-6. SPI Control Register 2 (SPIC2)**



Table 16-3. SPIC2 Register Field Descriptions

Field	Description
4 MODFEN	<b>Master Mode-Fault Function Enable</b> — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 16-2 for more details). 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	<b>Bidirectional Mode Output Enable</b> — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	<b>SPI Stop in Wait Mode</b> 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	<b>SPI Pin Control 0</b> — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

### 16.4.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.



Figure 16-7. SPI Baud Rate Register (SPIBR)

Table 16-4. SPIBR Register Field Descriptions

Field	Description
6:4 SPPR[2:0]	<b>SPI Baud Rate Prescale Divisor</b> — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 16-5. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 16-4).
2:0 SPR[2:0]	<b>SPI Baud Rate Divisor</b> — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 16-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 16-4). The output of this divider is the SPI bit rate clock for master mode.

**Table 16-5. SPI Baud Rate Prescaler Divisor**

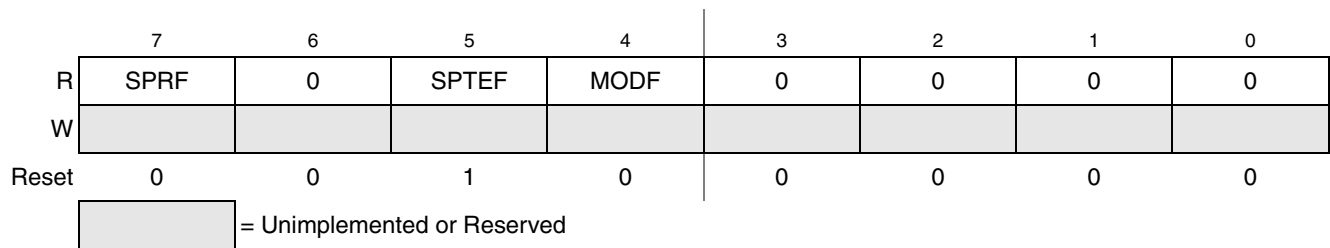
SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

**Table 16-6. SPI Baud Rate Divisor**

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

### 16.4.4 SPI Status Register (SPIS)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0. Writes have no meaning or effect.



**Figure 16-8. SPI Status Register (SPIS)**

Table 16-7. SPIS Register Field Descriptions

Field	Description
7 SPRF	<p><b>SPI Read Buffer Full Flag</b> — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPID). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register.</p> <p>0 No data available in the receive data buffer 1 Data available in the receive data buffer</p>
5 SPTEF	<p><b>SPI Transmit Buffer Empty Flag</b> — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIS with SPTEF set, followed by writing a data value to the transmit buffer at SPID. SPIS must be read with SPTEF = 1 before writing data to SPID or the SPID write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPIC1 is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPID is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter.</p> <p>0 SPI transmit buffer not empty 1 SPI transmit buffer empty</p>
4 MODF	<p><b>Master Mode Fault Flag</b> — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The <math>\overline{SS}</math> pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIC1).</p> <p>0 No mode fault error 1 Mode fault error detected</p>

### 16.4.5 SPI Data Register (SPID)

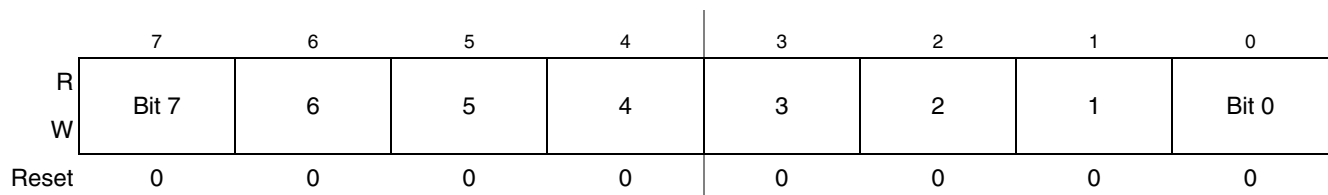


Figure 16-9. SPI Data Register (SPID)

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data must not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPID any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

## 16.5 Functional Description

An SPI transfer is initiated by checking for the SPI transmit buffer empty flag (SPTEF = 1) and then writing a byte of data to the SPI data register (SPID) in the master SPI device. When the SPI shift register is available, this byte of data is moved from the transmit data buffer to the shifter, SPTEF is set to indicate there is room in the buffer to queue another transmit character if desired, and the SPI serial transfer starts.

During the SPI transfer, data is sampled (read) on the MISO pin at one SPSCCK edge and shifted, changing the bit value on the MOSI pin, one-half SPSCCK cycle later. After eight SPSCCK cycles, the data that was in the shift register of the master has been shifted out the MOSI pin to the slave while eight bits of data were shifted in the MISO pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading SPID. If another byte of data is waiting in the transmit buffer at the end of a transfer, it is moved into the shifter, SPTEF is set, and a new transfer is started.

Normally, SPI data is transferred most significant bit (MSB) first. If the least significant bit first enable (LSBFE) bit is set, SPI data is shifted LSB first.

When the SPI is configured as a slave, its  $\overline{SS}$  pin must be driven low before a transfer starts and  $\overline{SS}$  must stay low throughout the transfer. If a clock format where CPHA = 0 is selected,  $\overline{SS}$  must be driven to a logic 1 between successive transfers. If CPHA = 1,  $\overline{SS}$  may remain low between successive transfers. See [Section 16.5.1, "SPI Clock Formats"](#) for more details.

Because the transmitter and receiver are double buffered, a second byte, in addition to the byte currently being shifted out, can be queued into the transmit data buffer, and a previously received character can be in the receive data buffer while a new character is being shifted in. The SPTEF flag indicates when the transmit buffer has room for a new character. The SPRF flag indicates when a received character is available in the receive data buffer. The received character must be read out of the receive buffer (read SPID) before the next transfer is finished or a receive overrun error results.

In the case of a receive overrun, the new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for such an overrun condition so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

### 16.5.1 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

[Figure 16-10](#) shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCCK edge and bit 8 ending one-half SPSCCK cycle after the sixteenth SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the

MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The  $\overline{SS}$  OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master  $\overline{SS}$  output goes to active low one-half SPSCCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The  $\overline{SS}$  IN waveform applies to the slave select input of a slave.

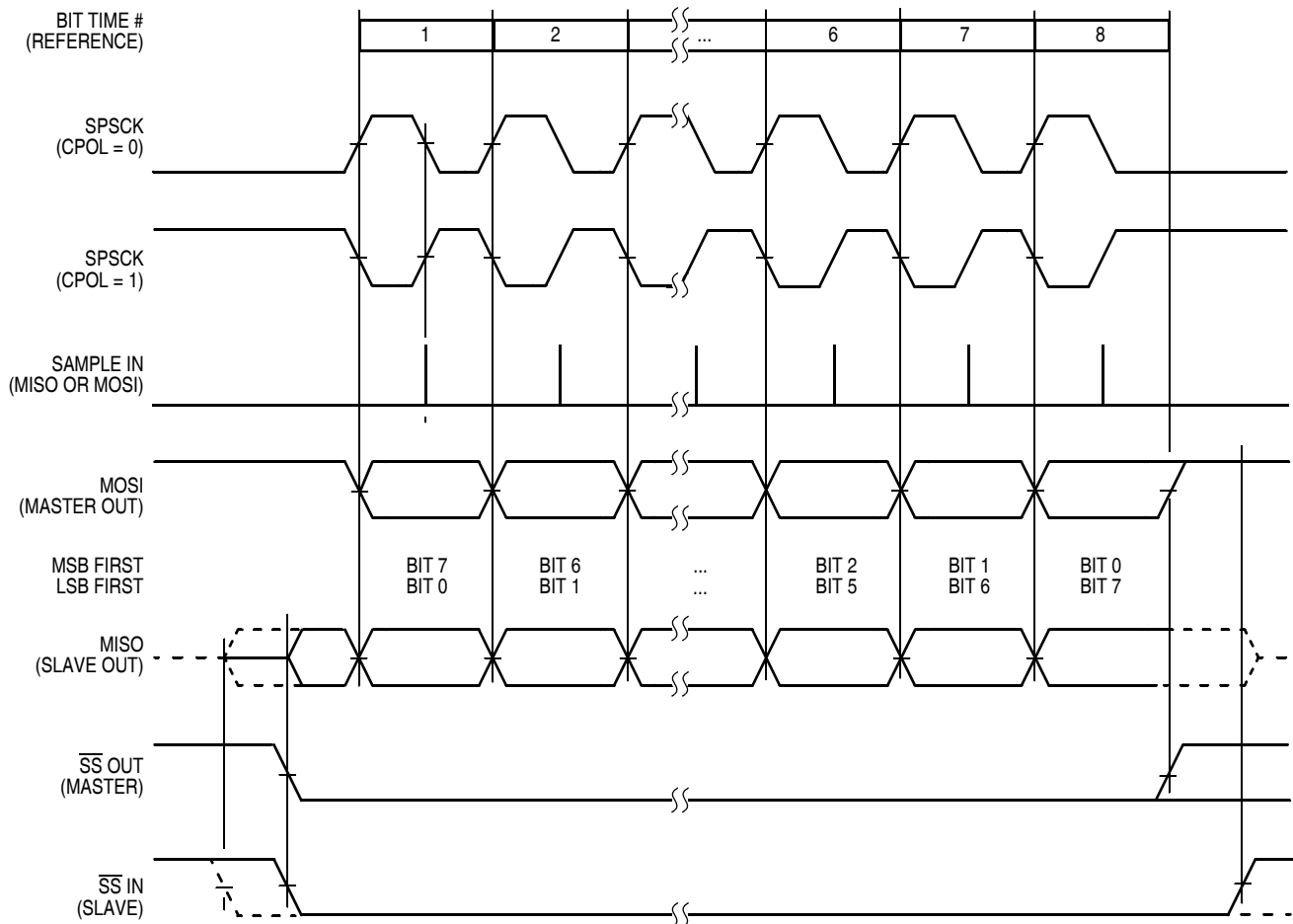


Figure 16-10. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when  $\overline{SS}$  goes to active low, but the data is not defined until the first SPSCCK edge. The first SPSCCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 1, the slave's  $\overline{SS}$  input is not required to go to its inactive high level between transfers.

Figure 16-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ( $\overline{SS}$  IN goes low), and bit 8 ends at the last SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting

in LSBFE. Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the  $\overline{\text{MOSI}}$  output pin from a master and the MISO waveform applies to the MISO output from a slave. The  $\overline{\text{SS}}$  OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master  $\overline{\text{SS}}$  output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCCK cycle after the end of the eighth bit time of the transfer. The  $\overline{\text{SS}}$  IN waveform applies to the slave select input of a slave.

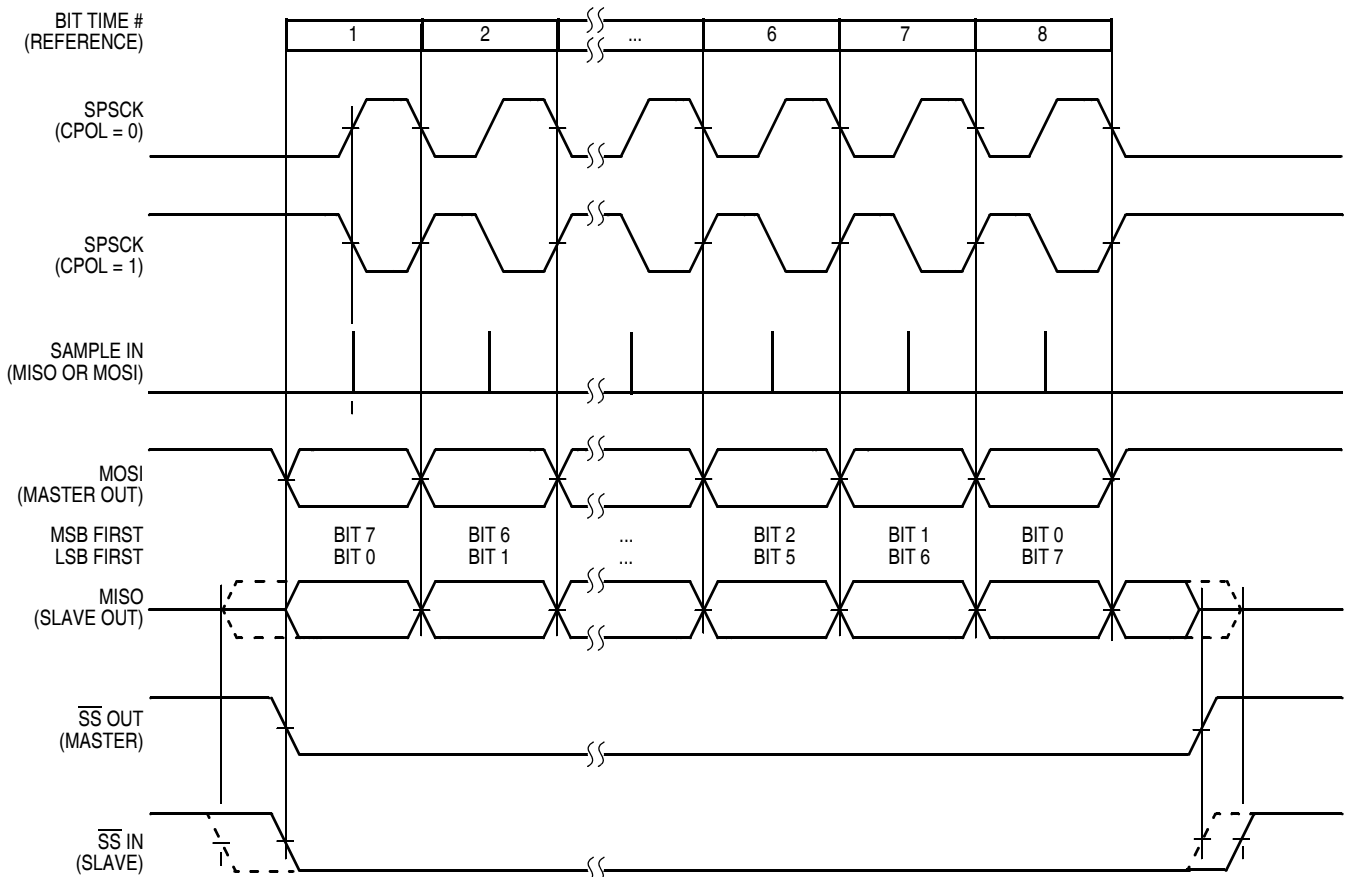


Figure 16-11. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when  $\overline{\text{SS}}$  goes to active low. The first SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's  $\overline{\text{SS}}$  input must go to its inactive high level between transfers.

## 16.5.2 SPI Interrupts

There are three flag bits, two interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) must check the flag bits to determine what event caused the interrupt. The service routine must also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

## 16.5.3 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the  $\overline{SS}$  pin (provided the  $\overline{SS}$  pin is configured as the mode fault input signal). The  $\overline{SS}$  pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's  $\overline{SS}$  pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPIC1). User software must verify the error condition has been corrected before changing the SPI back to master mode.





# Chapter 17

## Development Support

### 17.1 Introduction

Development support systems in the RS08 Family include the RS08 background debug controller (BDC).

The BDC provides a single-wire debug interface to the target MCU. This interface provides a convenient means for programming the on-chip FLASH and other nonvolatile memories. Also, the BDC is the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoint, and single-instruction trace commands.

In the RS08 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, including resetting the device without using a reset pin.

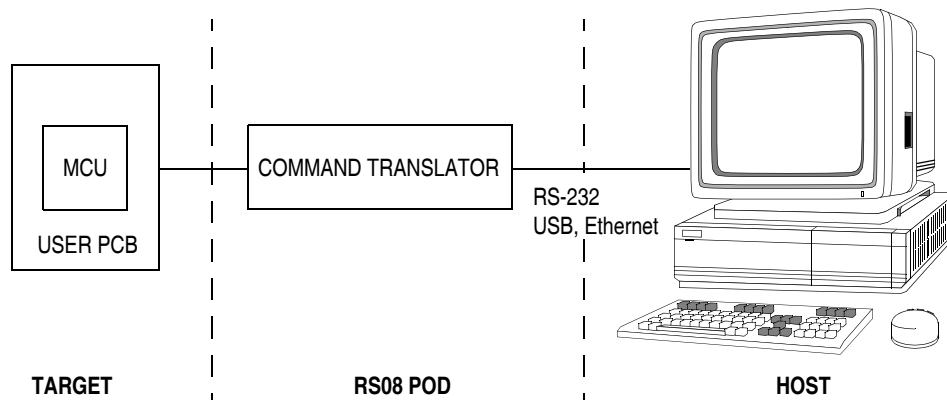


Figure 17-1. Connecting MCU to Host for Debugging

### 17.2 Features

Features of the RS08 background debug controller (BDC) include:

- Uses a single pin for background debug serial communications
- Non-intrusive of user memory resources; BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands allow access to memory resources while CPU is running user code without stopping applications
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from wait or stop modes

- BDC\_RESET command allows host to reset MCU without using a reset pin
- One hardware address breakpoint built into BDC
- RS08 clock source runs in stop mode if BDM enabled to allow debugging when CPU is in stop mode
- COP watchdog suspended while in active background mode

### 17.3 RS08 Background Debug Controller (BDC)

All MCUs in the RS08 Family contain a single-wire background debug interface which supports in-circuit programming of on-chip non-volatile memory and sophisticated debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this debug system provides for minimal interference with normal application resources. It does not use any user memory or locations in the memory map. It requires use of only the output-only BKGD pin. This pin will be shared with simple user output-only functions (typically port, comparator outputs, etc.), which can be easily debugged in normal user mode.

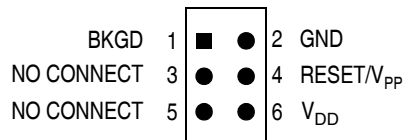
RS08 BDM commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). The BACKGROUND command causes the target MCU to enter active background mode. Active background mode commands allow the CPU registers to be read or written and allow the user to trace one (TRACE1) 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 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 (BDC).

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 communication such as Ethernet or a universal serial bus (USB) to communicate between the host PC and the pod.

Figure 17-2 shows the standard header for connection of a RS08 BDM pod. A pod is a small interface device that connects a host computer such as a personal computer to a target RS08 system. BKGD and GND are the minimum connections required to communicate with a target MCU. The pseudo-open-drain RESET signal is included in the connector to allow a direct hardware method for the host to force or monitor (if RESET is available as output) a target system reset.

The RS08 BDM pods supply the  $V_{PP}$  voltage to the RS08 MCU when in-circuit programming is required. The  $V_{PP}$  connection from the pod is shared with RESET as shown in Figure 17-2. For  $V_{PP}$  requirements see the flash specifications in the data sheet.



**Figure 17-2. Standard RS08 BDM Tool Connector**

Background debug controller (BDC) serial communications use a custom serial protocol that was first introduced on the M68HC12 Family of microcontrollers. This protocol requires that the host knows the communication clock rate, which is determined by the target BDC clock rate. 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.

For RS08 MCUs, the BDC clock is the same frequency as the MCU bus clock. For a detailed description of the communications protocol, refer to [Section 17.3.2, “Communication Details.”](#)

### 17.3.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. BKGD is a pseudo-open-drain pin that contains an on-chip pullup, therefore it requires no external pullup resistor. Unlike typical open-drain pins, the external resistor capacitor (RC) time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 17.3.2, “Communication Details,”](#) for more detail.

The primary function of this pin is bidirectional serial communication of background debug commands and data. During reset, this pin selects between starting in active background mode and normal user mode running an application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the target BDC clock frequency.

By controlling the BKGD pin and forcing an MCU reset (issuing a BDC\_RESET command, or through a power-on reset (POR)), the host can force the target system to reset into active background mode rather than start the user application program. This is useful to gain control of a target MCU whose FLASH program memory has not yet been programmed with a user application program.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD determines the normal operating mode.

On some RS08 devices, the BKGD pin may be shared with an alternative output-only function. To support BDM debugging, the user must disable this alternative function. Debugging of the alternative function must be done in normal user mode without using BDM.

### 17.3.2 Communication Details

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

The BDC serial communication protocol requires the host to know the target BDC clock speed. Commands and data are sent most significant bit first (MSB-first) at 16 BDC clock cycles per bit. 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.

Figure 17-3 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target 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 period, there is no need to treat the line as an open-drain signal during this period.

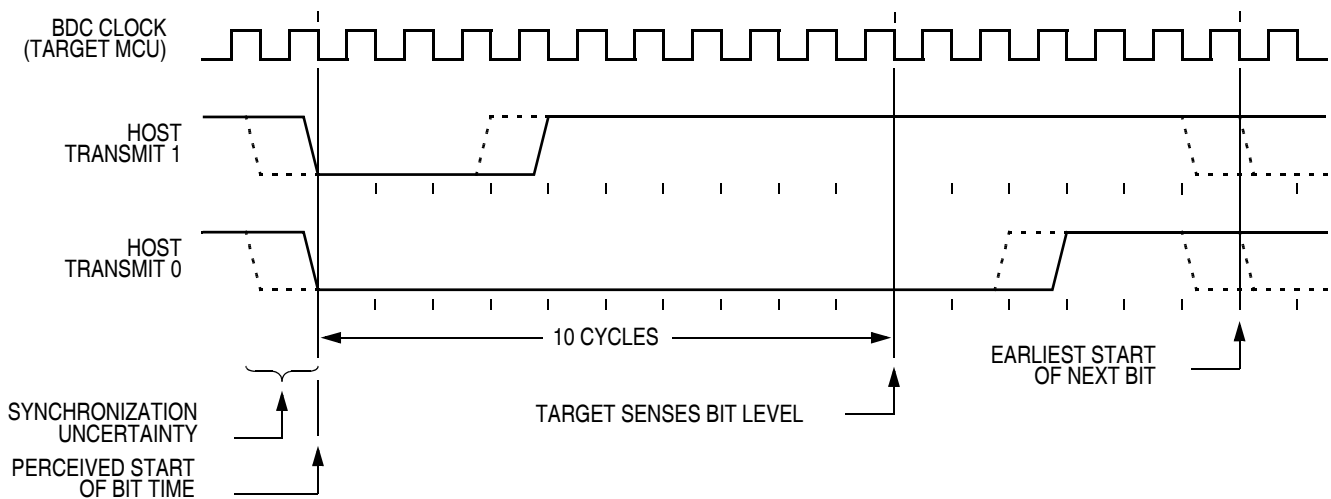
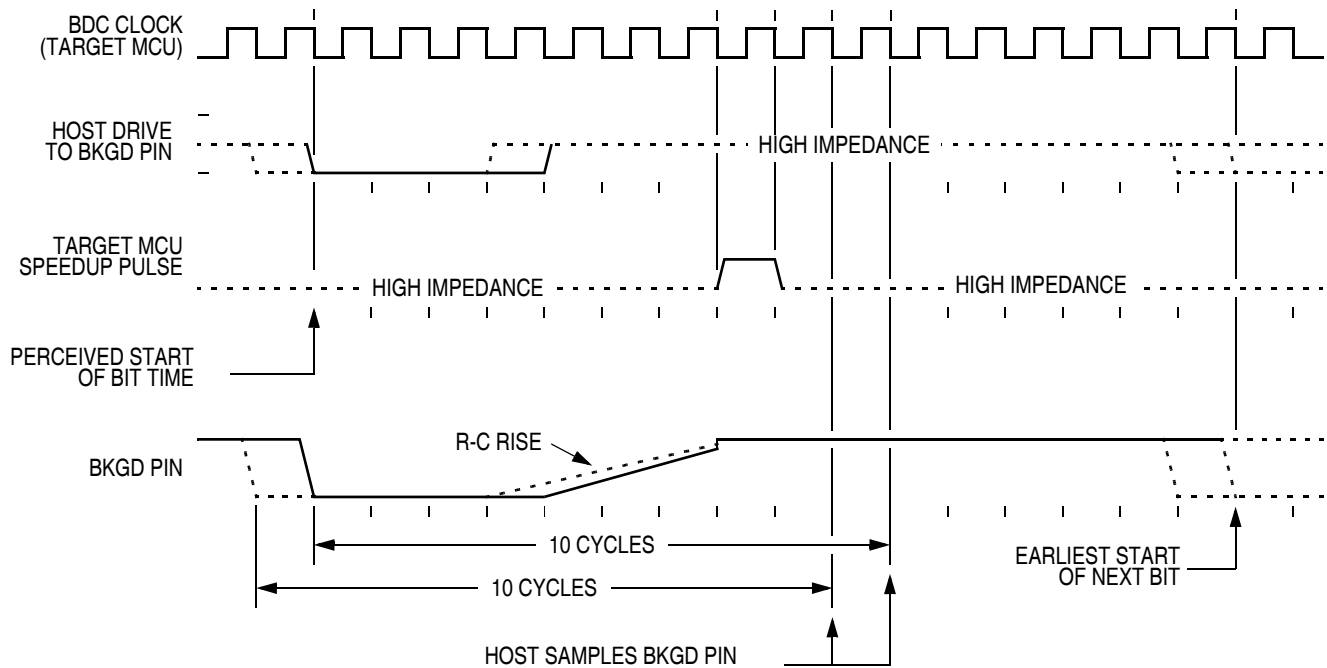


Figure 17-3. BDC Host-to-Target Serial Bit Timing

Figure 17-4 shows the host receiving a logic 1 from the target MCU. Because the host is asynchronous to the target, 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. 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 drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host must sample the bit level approximately 10 cycles after it started the bit time.



**Figure 17-4. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 17-5 shows the host receiving a logic 0 from the target MCU. Because the host is asynchronous to the target, 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. The host initiates the bit time but the target 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 approximately 10 cycles after starting the bit time.

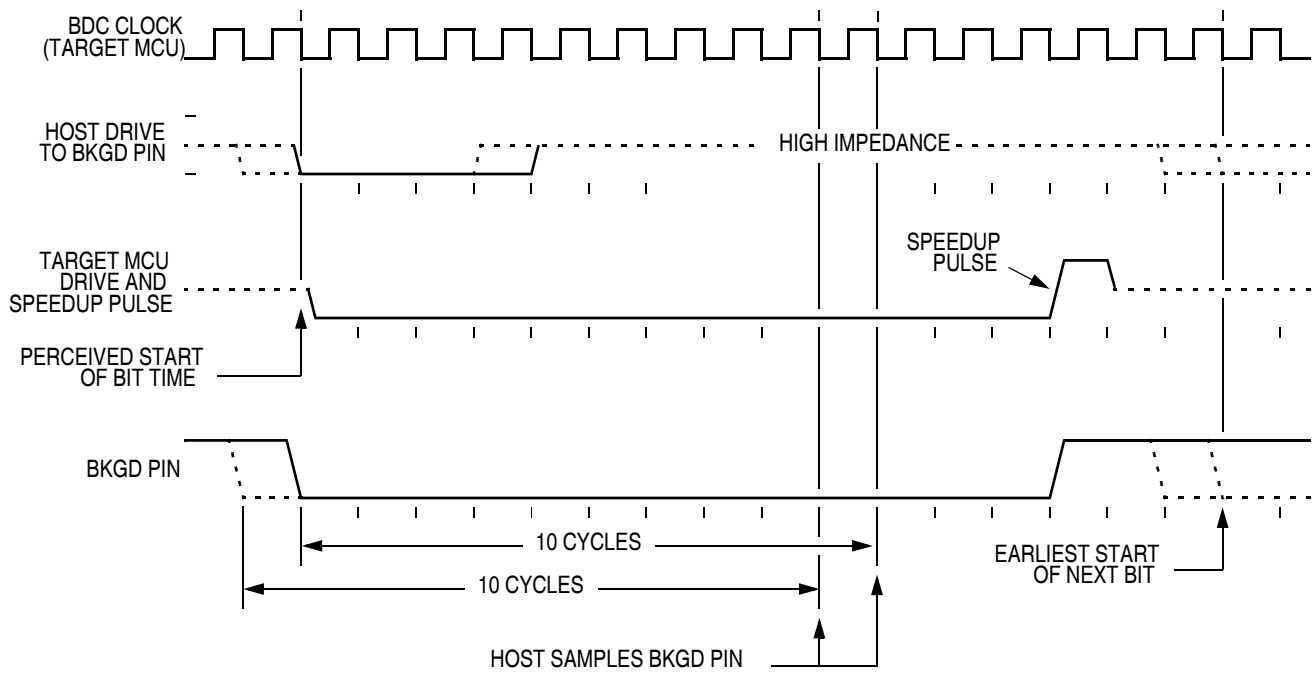


Figure 17-5. BDM Target-to-Host Serial Bit Timing (Logic 0)

### 17.3.3 SYNC and Serial Communication Timeout

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target will keep waiting for a rising edge on BKGD to answer the SYNC request pulse. If the rising edge is not detected, the target will keep waiting indefinitely, without any timeout limit. When a rising edge on BKGD occurs after a valid SYNC request, the BDC will drive the BKGD pin low for exactly 128 BDC cycles.

Consider now the case where the host returns BKGD to logic 1 before 128 cycles. This is interpreted as a valid bit transmission, and not as a SYNC request. The target will keep waiting for another falling edge marking the start of a new bit. If, however, a new falling edge is not detected by the target within 512 clock cycles since the last falling edge, a timeout occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset to the BDC.

If a read command is issued but the data is not retrieved within 512 serial clock cycles, a soft-reset will occur causing the command to be disregarded. The data is not available for retrieving after the timeout has occurred. A soft-reset is also used to end a READ\_BLOCK or WRITE\_BLOCK command.

The following describes the actual bit-time requirements for a host to guarantee logic 1 or 0 bit transmission without the target timing out or interpreting the bit as a SYNC command:

- To send a logic 0, BKGD must be kept low for a minimum of 12 BDC cycles and up to 511 BDC cycles except for the first bit of a command sequence, which will be detected as a SYNC request.
- To send a logic 1, BKGD must be held low for at least four BDC cycles, be released by the eighth cycle, and be held high until at least the sixteenth BDC cycle.

- Subsequent bits must occur within 512 BDC cycles of the last bit sent.

## 17.4 BDC Registers and Control Bits

The BDC contains two non-CPU accessible 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 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 status bits (BDMACT, WS, and WSF) are read-only status indicators and can never be written by the WRITE\_CONTROL serial BDC command.

### 17.4.1 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	0	WS	WSF	0
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	0	0	0	0

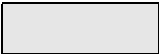
 = Unimplemented or Reserved

Figure 17-6. BDC Status and Control Register (BDCSCR)

Table 17-1. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	<b>Enable BDM (Permit Active Background Mode)</b> — 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. If the application can go into stop mode, this bit is required to be set if debugging capabilities are required. 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	<b>Background Mode Active Status</b> — This is a read-only status bit. 0 BDM not active (user application program running). 1 BDM active and waiting for serial commands.

Table 17-1. BDCSCR Register Field Descriptions (continued)

Field	Description
5 BKPTEN	<b>BDC Breakpoint Enable</b> — 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	<b>Force/Tag Select</b> — 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).
2 WS	<b>Wait or Stop Status</b> — 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 must 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	<b>Wait or Stop Failure Status</b> — 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.

## 17.4.2 BDC Breakpoint Match Register

This 16-bit register holds the 14-bit 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. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. However, because READ\_BKPT and WRITE\_BKPT are non-intrusive commands, they could be executed even while the user program is running. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to the RS08 Family Reference Manual [Section 17.6](#), “BDC Hardware Breakpoint.”



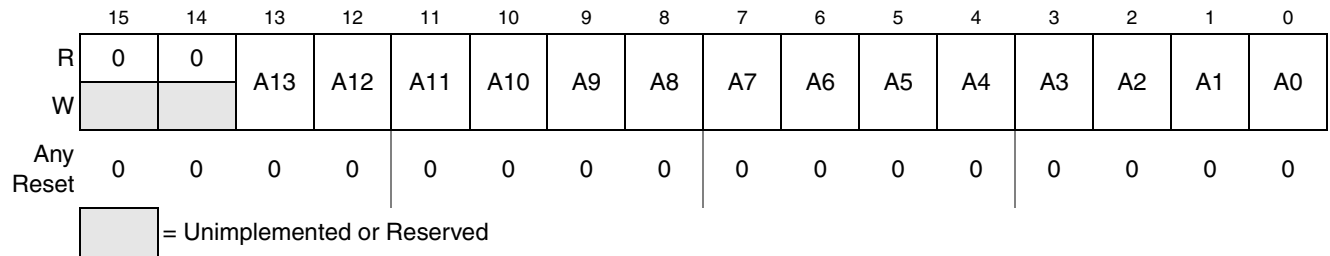


Figure 17-7. BDC Breakpoint Match Register (BDCBKPT)

## 17.5 RS08 BDC Commands

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

Table 17-2 shows all RS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

### Coding Structure Nomenclature

The following nomenclature is used in Table 17-2 to describe the coding structure of the BDC commands.

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

/ = Separates parts of the command

d = Delay 16 to 511 target BDC clock cycles

soft-reset = Delay of at least 512 BDC clock cycles from last host falling-edge

AAAA = 16-bit address in the host-to-target direction<sup>1</sup>

RD = Eight bits of read data in the target-to-host direction

WD = Eight 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 = Eight 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)

1. The RS08 CPU uses only 14 bits of address and occupies the lower 14 bits of the 16-bit AAAA address field. The values of address bits 15 and 14 in AAAA are truncated and thus do not matter.

# SYNC

**Table 17-2. RS08 BDC Command Summary**

Command Mnemonic	Active Background Mode/ Non-Intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a <sup>1</sup>	Request a timed reference pulse to determine target BDC communication speed
BDC_RESET	Any CPU mode	18 <sup>2</sup>	Request an MCU reset
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
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 background mode	08/d	Go to execute the user application program starting at the address currently in the PC
TRACE1	Active background mode	10/d	Trace one user instruction at the address in the PC, then return to active background mode
READ_BLOCK	Active background mode	80/AAAA/d/RD <sup>3</sup>	Read a block of data from target memory starting from AAAA continuing until a soft-reset is detected
WRITE_BLOCK	Active background mode	88/AAAA/WD/d <sup>4</sup>	Write a block of data to target memory starting at AAAA continuing until a soft-reset is detected
READ_A	Active background mode	68/d/RD	Read accumulator (A)

Table 17-2. RS08 BDC Command Summary (continued)

Command Mnemonic	Active Background Mode/ Non-Intrusive	Coding Structure	Description
WRITE_A	Active background mode	48/WD/d	Write accumulator (A)
READ_CCR_PC	Active background mode	6B/d/RD16 <sup>5</sup>	Read the CCR bits z, c concatenated with the 14-bit program counter (PC) RD16=zc:PC
WRITE_CCR_PC	Active background mode	4B/WD16/d <sup>6</sup>	Write the CCR bits z, c concatenated with the 14-bit program counter (PC) WD16=zc:PC
READ_SPC	Active background mode	6F/d/RD16 <sup>7</sup>	Read the 14-bit shadow program counter (SPC) RD16=0:0:SPC
WRITE_SPC	Active background mode	4F/WD16/d <sup>8</sup>	Write 14-bit shadow program counter (SPC) WD16 = x:x:SPC, the two most significant bits shown by "x" are ignored by target

<sup>1</sup> The SYNC command is a special operation which does not have a command code.

<sup>2</sup> 18 was HCS08 BDC command for TAGGO.

<sup>3</sup> Each RD requires a delay between host read data byte and next read, command ends when target detects a soft-reset.

<sup>4</sup> Each WD requires a delay between host write data byte and next byte, command ends when target detects a soft-reset.

<sup>5</sup> HCS08 BDC had separate READ\_CCR and READ\_PC commands, the RS08 BDC combined this commands.

<sup>6</sup> HCS08 BDC had separate WRITE\_CCR and WRITE\_PC commands, the RS08 BDC combined this commands.

<sup>7</sup> 6F is READ\_SP (read stack pointer) for HCS08 BDC.

<sup>8</sup> 4F is WRITE\_SP (write stack pointer) for HCS08 BDC.

Request a timed pulse (128 BDC clock cycles) from target

Non-Intrusive

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)
- 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 1
- Delays 16 cycles to allow the host to stop driving the high speedup pulse

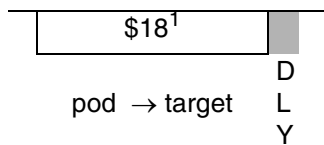
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

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

### 17.5.1 BDC\_RESET

Reset the MCU

Any CPU Mode



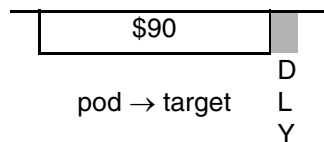
<sup>1</sup> \$18 is the HCS08 BDC command for TAGGO

Provided the BKGD pin is available, the target MCU can be reset to enter active background mode by the BDC\_RESET command followed immediately by asserting the BKGD pin low until the MCU reset sequence finishes. If BKGD is left high after a BDC\_RESET, the target MCU will reset into normal user mode. Systems that can place the CPU into wait or stop mode require ENBDM to be set to allow the BDC clocks to remain active while the CPU is in stop mode.

### 17.5.2 BACKGROUND

Enter Active Background Mode (if Enabled)

Non-intrusive



Provided ENBDM is set, the BACKGROUND command causes the target MCU to enter active background mode as soon as the current CPU instruction finishes.

If ENBDM is clear (its default value), the BACKGROUND command will be ignored by the BDC. The host should attempt to enable ENBDM using WRITE\_STATUS and verify that ENBDM is set using READ\_STATUS before issuing a BACKGROUND command.

If the target application uses wait or stop mode, it may not be possible to enter active background mode without causing a wakeup using an external interrupt.

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

Normally, the development host would set ENBDM once at the beginning of a debug session or after a target system reset, and then leave the ENBDM bit set during debugging operations. During debugging, the host would use GO and TRACE1 commands to move from active background mode to normal user program execution and would use BACKGROUND commands or breakpoints to return to active background mode. This method of debugging allow the host debugger to enter active background mode even if the CPU enters wait or stop mode.

### 17.5.3 READ\_STATUS

Read Status from BDCSCR

Non-intrusive

\$E4	Read BDCSCR (8)
pod → target	target → pod

This command allows a host to read the contents of the BDC status and control register (BDCSCR). This register is not in the memory map of the target MCU and is accessible only through READ\_STATUS and WRITE\_CONTROL serial BDC commands.

The most common use for this command is to allow the host to determine whether the target MCU is executing normal user program instructions or if it is in active background mode. For example, during a typical debug session, the host might set breakpoints in the user program and then use a GO command to begin normal user program execution. The host would then periodically execute READ\_STATUS commands to determine when a breakpoint has been encountered and the target processor has gone into active background mode. After the target has entered active background mode, the host reads the contents of target CPU registers.

READ\_STATUS can also be used to check whether the target MCU has gone into wait or stop mode. During a debug session, the host or user may decide that it has taken too long to reach a breakpoint in the user program. The host could then issue a READ\_STATUS command and check the WS status bit to determine whether the target MCU is still running user code or whether it has entered wait or stop mode. If WS = 0 and BDMACT = 0, meaning it is running user code and is not in wait or stop, the host might choose to issue a BACKGROUND command to stop the user program and enter active background mode where the host can check the CPU registers and find out what the target program is doing.

### 17.5.4 WRITE\_CONTROL

Write Control Bits in BDCSCR

Non-intrusive

\$C4	Write BDCSCR (8)
pod → target	pod → target

This command is used to enable active background mode and control the hardware breakpoint logic in the BDC by writing to control bits in the BDC status and control register (BDCSCR). This register is not in the memory map of the target MCU and is only accessible through READ\_STATUS and WRITE\_CONTROL serial BDC commands. Some bits in BDCSCR have write restrictions (such as status

bits BDMACT, WS, and WSF, which are read-only status indicators, and ENBDM, which cannot be cleared while BDM is active.

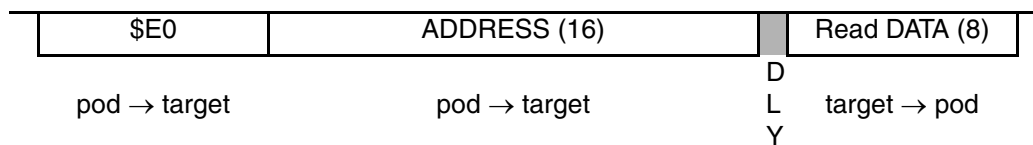
The ENBDM control bit defaults to 0 (active background mode not allowed) when the target MCU is reset in normal operating mode. WRITE\_CONTROL is used to enable the active background mode. This is normally done once and ENBDM is left enabled throughout the debug session. However, the debug system may want to change ENBDM to 0 and measure true stop current in the target system (because ENBDM = 1 prevents the clock generation circuitry from disabling the internal clock oscillator or crystal oscillator to allow the BDC clock to continue when the CPU executes a STOP instruction).

The breakpoint enable (BKPTEN) and force/tag select (FTS) control bits are used to control the hardware breakpoint logic in the BDC. This is a single breakpoint that compares the current CPU address against the value in the BDCBKPT register. A WRITE\_CONTROL command is used to change BKPTEN and FTS, and a WRITE\_BKPT command is used to write the 16-bit BDCBKPT address match register.

### 17.5.5 READ\_BYTE

Read Data from Target Memory Location

Non-intrusive



This command is used to read the contents of a memory location in the target MCU without checking the BDC status to ensure the data is valid. In systems where the target is currently in active background mode or is known to be executing a program which has no STOP or WAIT instructions, READ\_BYTE is faster than the more general READ\_BYTE\_WS, which reports status in addition to returning the requested read data. The most significant use of the READ\_BYTE command is during in-circuit FLASH programming where the host downloads data to be programmed at the same time the target CPU is executing the code that actually programs the FLASH memory. Because the host provides the FLASH programming code, it can guarantee that there are no STOP or WAIT instructions.

The READ\_BYTE command should not be used in general-purpose user programs that use STOP or WAIT instructions. To avoid invalid data due to CPU operation in stop or wait mode, the READ\_BYTE\_WS should be used determine whether the data is valid.

## 17.5.6 READ\_BYTE\_WS

Read Data from Target and Report Status

Non-intrusive

\$E1	ADDRESS (16)	Read BDCSCR (8)	Read DATA (8)
pod → target	pod → target	target → pod	target → pod
		D	
		L	
		Y	

This is the command normally used by a host debug system to perform general-purpose memory read operations. In addition to returning the data from the requested target memory location, this command returns the contents of the BDC status and control register. The status information can be used to determine whether the data that was returned is valid or not. If the target MCU was just entering wait or stop mode at the time of the read, the wait/stop failure (WSF) status bit will be 1. If WSF is 0, the data that was returned is valid.

If the WSF bit indicates a WAIT or STOP instruction caused the read operation to fail, do a BACKGROUND command to force the target system out of wait or stop mode and into active background mode. From there, repeat the failed read operation, and if desired, adjust the PC to point to the WAIT or STOP instruction and issue a GO to return the target to wait or stop mode.

## 17.5.7 WRITE\_BYTE

Write Data to Target Memory Location

Non-intrusive

\$C0	ADDRESS(16)	Write DATA(8)	
pod → target	pod → target	pod → target	
			D
			L
			Y

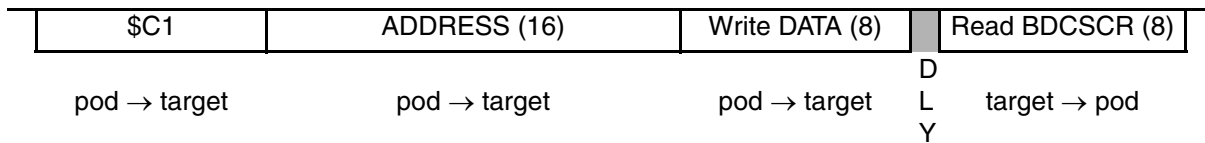
This command is used to write the contents of a memory location in the target MCU without checking the BDC status to ensure the write was completed successfully. In systems where the target is currently in active background mode or is known to be executing a program that has no STOP or WAIT instructions, WRITE\_BYTE is faster than the more general WRITE\_BYTE\_WS, which reports status in addition to performing the requested write operation. The most significant use of the WRITE\_BYTE command is during in-circuit FLASH programming where the host downloads data to be programmed at the same time the target CPU is executing the code that actually programs the FLASH memory. Because the host provides the FLASH programming code, it can guarantee that there are no STOP or WAIT instructions.

The WRITE\_BYTE command should not be used in general-purpose user programs which use STOP or WAIT instructions that can occur at any time. To avoid invalid data due to CPU in stop or wait mode, the WRITE\_BYTE\_WS should be used to determine whether the data that was returned is valid or not.

## 17.5.8 WRITE\_BYTE\_WS

Write Data to Target and Report Status

Non-intrusive



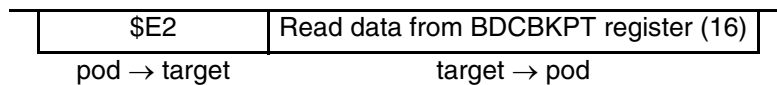
This is the command normally used by a host debug system to perform general-purpose memory write operations. In addition to performing the requested write to a target memory location, this command returns the contents of the BDC status and control register. The status information can be used to determine whether the write operation was completed successfully. If the target MCU was just entering wait or stop mode at the time of the read, the wait/stop failure (WSF) status bit will be 1 and the write command is cancelled. If WSF is 0, the write operation was completed successfully.

If the WSF bit indicates a WAIT or STOP instruction caused the write operation to fail, do a BACKGROUND command to force the target system out of wait or stop mode and into active background mode. From there, repeat the failed write operation, and if desired, adjust the PC to point to the WAIT or STOP instruction and issue a GO to return the target to wait or stop mode.

## 17.5.9 READ\_BKPT

Read 16-Bit BDC Breakpoint Register (BDCBKPT)

Non-intrusive

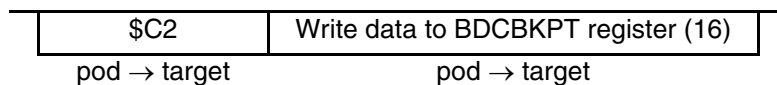


This command is used to read the 16-bit BDCBKPT address match register in the hardware breakpoint logic in the BDC.

## 17.5.10 WRITE\_BKPT

Write 16-Bit BDC Breakpoint Register (BDCBKPT)

Non-intrusive



This command is used to write a 16-bit address value into the BDCBKPT register in the BDC. This establishes the address of a breakpoint. The BKPTEN bit in the BDCSCR determines whether the breakpoint is enabled. If BKPTEN = 1 and the FTS control bit in the BDCSCR is set (force), a successful match between the CPU address and the value in the BDCBKPT register will force a transition to active background mode at the next instruction boundary. If BKPTEN = 1 and FTS = 0, the opcode at the address specified in the BDCBKPT register will be tagged as it is fetched into the instruction queue. If and when a tagged opcode reaches the top of the instruction queue and is about to be executed, the MCU will enter active background mode rather than execute the tagged instruction.

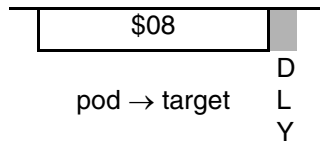


In normal debugging environments, breakpoints are established while the target MCU is in active background mode before going to the user program. However, because this is a non-intrusive command, it could be executed even when the MCU is running a user application program.

### 17.5.11 GO

Start Execution of User Program Starting at Current PC

Active Background Mode

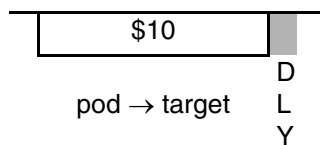


This command is used to exit the active background mode and begin execution of user program instructions starting at the address in the PC. Typically, the host debug monitor program modifies the PC value (using a `WRITE_PC` command) before issuing a `GO` command to go to an arbitrary point in the user program. This `WRITE_PC` command is not needed if the host simply wants to continue the user program where it left off when it entered active background mode.

### 17.5.12 TRACE1

Run One User Instruction Starting at the Current PC

Active Background Mode

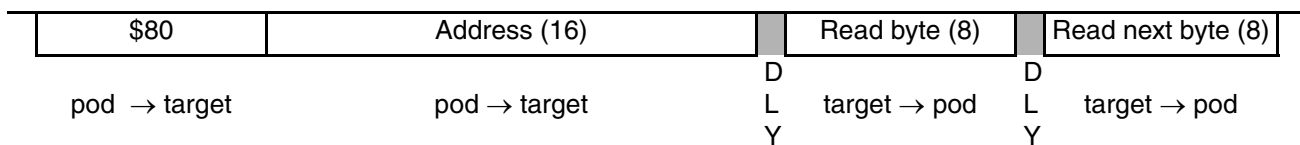


This command is used to run one user instruction and return to active background mode. The address in the PC determines what user instruction will be executed, and the PC value after `TRACE1` is completed will reflect the results of the executed instruction.

### 17.5.13 READ\_BLOCK

Read a block of data from target memory

Active Background Mode



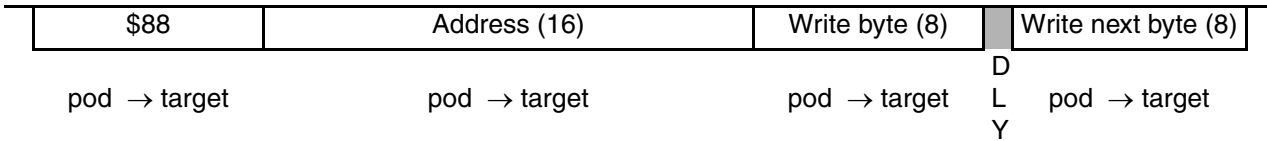
This command is used to read the contents of a block of memory starting at the location provided in the command. Because this command is only available in active background mode, the CPU cannot enter wait or stop; therefore the command does not return the contents of the `BDMSCR`. This command will continue to read data from the next memory location until the `BDC` detects a soft-reset, which is a timeout of 512

BDC cycles from the last falling edge of the host. This command is useful when dumping large blocks of data for memory displays or in programmers to verify the device data after programming.

### 17.5.14 WRITE\_BLOCK

Write a block of data to target memory

Active Background Mode

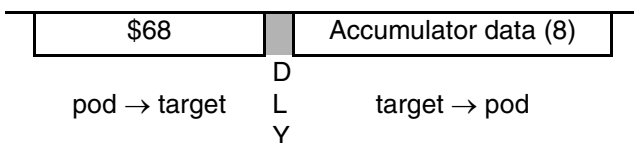


This command is used to write data to a block of memory starting at the location provided in the command. Because this command is only available in active background mode, the CPU cannot enter wait or stop, therefore the command does not return the contents of the BDMSCR. This command will continue to write data to the next memory location until the BDC detects a soft-reset, which is a timeout of 512 BDC cycles from the last falling edge of the host. This command is useful when writing large blocks of data such as full blocks of RAM.

### 17.5.15 READ\_A

Read Accumulator A of the Target CPU

Active Background Mode

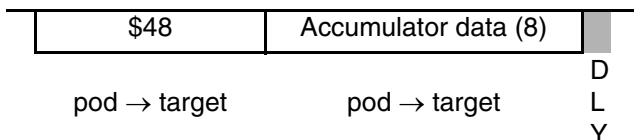


Read the contents of the accumulator (A) of the target CPU. Because the CPU in the target MCU is effectively halted while the target is in active background mode, there is no need to save the target CPU registers on entry into active background mode and no need to restore them on exit from active background mode to a user program.

### 17.5.16 WRITE\_A

Write Accumulator A of the Target CPU

Active Background Mode

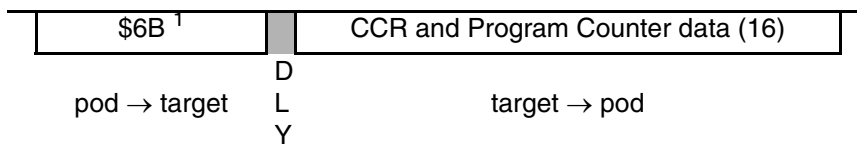


Write new data to the accumulator (A) of the target CPU. This command can be used to change the value in the accumulator before returning to the user application program via a GO or TRACE1 command.

### 17.5.17 READ\_CCR\_PC

Reads the CCR and the Program Counter of the Target CPU

Active Background Mode



<sup>1</sup> \$6B is the HCS08 BDC command for READ\_PC, the HCS08 CCR was read using READ\_CCR

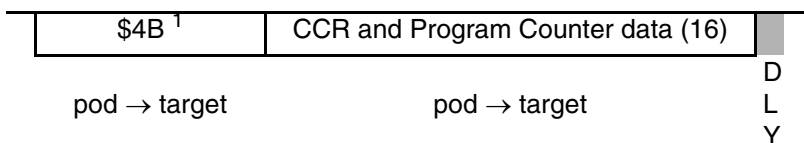
Read the Z and C bits of the condition code register (CCR) and contents of the 14-bit program counter (PC) of the target CPU.

The value in the PC when the target MCU enters active background mode is the address of the instruction that would have executed next if the MCU had not entered active background mode. If the target CPU was in wait or stop mode when a BACKGROUND command caused it to go to active background mode, the PC will hold the address of the instruction after the WAIT or STOP instruction that was responsible for the target CPU being in wait or stop, and the WS bit will be set. In the boundary case (where an interrupt and a BACKGROUND command arrived at approximately the same time and the interrupt was responsible for the target CPU leaving wait or stop—and then the BACKGROUND command took effect), the WS bit will be clear and the PC will be pointing at the next instruction after the WAIT or STOP. In the case of a software breakpoint (where the host placed a BGND opcode at the desired breakpoint address), the PC will be pointing at the address immediately following the inserted BGND opcode, and the host monitor will adjust the PC backward by one after removing the software breakpoint.

### 17.5.18 WRITE\_CCR\_PC

Write the CCR and Program Counter of the Target CPU

Active Background Mode



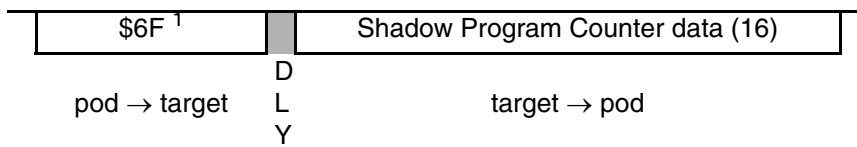
<sup>1</sup> \$4B is the HCS08 BDC command for WRITE\_PC, the HCS08 CCR was written using WRITE\_CCR

This command is used to change the contents of Z and C bits the condition code register (CCR) and the 14-bit program counter (PC) of the target CPU before returning to the user application program via a GO or TRACE1 command.

## 17.5.19 READ\_SPC

Reads the Shadow Program Counter of the Target CPU

Active Background Mode



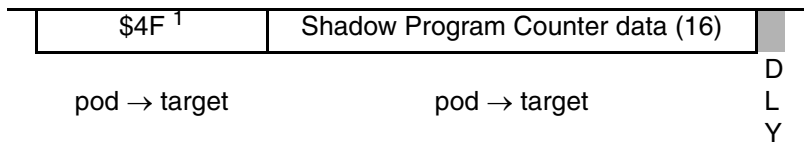
<sup>1</sup> \$6F is the HCS08 BDC command for READ\_SP (stack pointer)

Read the contents of the 14-bit shadow program counter (PC) of the target CPU.

## 17.5.20 WRITE\_SPC

Write the Shadow Program Counter of the Target CPU

Active Background Mode



<sup>1</sup> \$4F is the HCS08 BDC command for WRITE\_SP (stack pointer)

Writes the contents of the 14-bit shadow program counter (PC) of the target CPU. The two most significant bits of the 16-bit WD16 are ignored by the target.

## 17.6 BDC Hardware Breakpoint

The BDC includes one hardware breakpoint, which compares the CPU address bus to a 14-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. Tagged breakpoints must be placed only 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 8-bit BDCSCR and the 16-bit BDCBKPT address match register are built directly into the BDC and are not accessible in the normal MCU memory map. This means that the user application program cannot access these registers. Dedicated BDC serial commands are the only way to access these registers.

READ\_STATUS and WRITE\_CONTROL are used to read or write BDCSCR, respectively.

READ\_BKPT and WRITE\_BKPT are used to read or write the 16-bit BDCBKPT address match register, respectively.

If the background mode has not been enabled, ENBDM = 0, the CPU will cause an illegal opcode reset instead of going into active background mode.

## 17.7 BDM in Stop and Wait Modes

The clock architecture of the RS08 permits the BDC to prevent the BDC clock from stopping during wait or stop mode if ENBDM is set. In such a system, the debug host can use READ\_STATUS commands to determine whether the target is in a low-power mode (wait or stop). If the target is in wait or stop (WS = 1), the BACKGROUND command may be used to wake the target and place it in active background mode. When the CPU returns to active background mode, the PC will be pointing at the address of the instruction after the WAIT or STOP. From active background mode, the debug host can read or write memory or registers. The debug host can then choose to adjust the PC such that a GO command will return the target MCU to wait or stop mode.

If the CPU is in active background mode and the user issues a GO command and the next instruction is WAIT or STOP, the CPU goes into wait or stop mode.

If the user issues a TRACE1 command and the next instruction is WAIT or STOP, the CPU executes and completes the instruction and re-enters active background mode. When the CPU returns to active background mode, the PC will be pointing at the address of the instruction after the WAIT or STOP.

## 17.8 BDC Command Execution

The RS08 BDC requires no system resources except for the BKGD pin. A running application that is not in wait or stop mode can have its memory or register contents read or written without stopping the application. The RS08 BDC steals CPU cycles whenever a BDC command requires reading or writing memory or registers. This has little impact on real-time operation of user code because a memory access command takes eight bits for the command, 16 bits for the address, at least eight bits for the data, and a 16-cycle delay within the command. Each bit time is at least 16 BDC clock cycles so  $(32 \times 16) + 16 = 528$  cycles, thus the worst case impact is no more than 1/528 cycles, even if there are continuous back-to-back memory access commands through the BDM, which would be very ugrep timenlikely.

Because the RS08 BDC doesn't wait for free cycles, the delays between address and data in read commands and the delay after the data portion of a write command can be very short. In the RS08, the delay within a memory access command is 16 target bus cycles. For read or write accesses to registers within the BDC (STATUS and BDCBKPT), no delay is required.

Because the memory access commands in the RS08 BDC are actually performed by the CPU circuitry, it is possible for a memory access to fail when the memory access command coincides with the CPU entering wait or stop.

The WSF status bit was added to indicate an access failed because the CPU was just entering wait or stop mode. The READ\_BYTE\_WS command returns byte of status information and read data byte. The WRITE\_BYTE\_WS command includes the byte of status information in the target-to-host direction after the write data byte (which is in the host-to-target direction).





## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **Web Support:**

<http://www.freescale.com/support>

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
1-800-521-6274 or +1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

### **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)  
[www.freescale.com/support](http://www.freescale.com/support)

### **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](mailto:support.japan@freescale.com)

### **Asia/Pacific:**

Freescale Semiconductor China Ltd.  
Exchange Building 23F  
No. 118 Jianguo Road  
Chaoyang District  
Beijing 100022  
China  
+86 10 5879 8000  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### **For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or +1-303-675-2140  
Fax: +1-303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

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