THE DATASHEET OF

MC9RS08KB4CFK

www.win-source.net 0086-755-83957316

MC9RS08KB12
MC9RS08KB8
MC9RS08KB4
MC9RS08KB2

Reference Manual

RS08 Microcontrollers

Related Documentation:

- **MC9RS08KB12(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

MC9RS08KB12RM
Rev. 4
1/2012

freescale.com

freescale semiconductor

MC9RS08KB12 Series Features

8-Bit RS08 Central Processor Unit (CPU)

- Up to 20 MHz CPU at 1.8 V to 5.5 V across temperature range of –40 °C to 85 °C
- Subset of HC08 instruction set with added BGND instruction
- Single global interrupt vector

On-Chip Memory

- Up to 12 KB flash read/program/erase over full operating voltage and temperature, 12 KB/8 KB/4 KB/2 KB flash are optional
- Up to 254-byte random-access memory (RAM), 254-byte/126-byte RAM are optional
- Security circuitry to prevent unauthorized access to flash contents

Power-Saving Modes

- Wait mode — CPU shuts down; system clocks continue to run; full voltage regulation
- Stop mode — CPU shuts down; system clocks are stopped; voltage regulator in standby
- Wakeup from power-saving modes using RTI, KBI, ADC, ACMP, SCI and LVD

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; precision trimming of internal reference allows 0.2% resolution and 2% deviation over temperature and voltage; supports bus frequencies up to 10 MHz

System Protection

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1 kHz internal low power oscillator
- Low-voltage detection with reset or interrupt
- Illegal opcode detection with reset
- Illegal address detection with reset
- Flash-block protection

Development Support

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

Peripherals

- **ADC** — 12-channel, 10-bit resolution; 2.5 µs conversion time; automatic compare function; 1.7 mV/°C temperature sensor; internal bandgap reference channel; operation in stop; hardware trigger
- **ACMP** — Analog comparator; full rail-to-rail supply operation; option to compare to fixed internal bandgap reference voltage; can operate in stop mode
- **TPM** — One 2-channel timer/pulse-width modulator module; selectable input capture, output compare, or buffered edge- or center-aligned PWM on each channel
- **IIC** — Inter-integrated circuit bus module capable of operation up to 100 kbps with maximum bus loading; capable of higher baudrates with reduced loading
- **SCI** — One serial communications interface module with optional 13-bit break; LIN extensions
- **MTIM** — Two 8-bit modulo timers; optional clock sources
- **RTI** — One real-time clock with optional clock sources
- **KBI** — Keyboard interrupts; up to 8 ports

Input/Output

- 18 GPIOs in 24- and 20-pin packages; 14 GPIOs in 16-pin package; 6 GPIOs in 8-pin package; 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

- MC9RS08KB12/MC9RS08KB8/MC9RS08KB4
  - 24-pin QFN, 20-pin SOIC, 16-pin SOIC NB or TSSOP
- MC9RS08KB2
  - 8-pin SOIC or DFN

MC9RS08KB12 Series Reference Manual

Covers: MC9RS08KB12
MC9RS08KB8
MC9RS08KB4
MC9RS08KB2

MC9RS08KB12
Rev. 4
1/2012

freescale™
semiconductor

Revision History

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

http://freescale.com

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

Revision Number	Revision Date	Description of Changes
1	4/7/2009	Initial release for alpha customers.
2	8/18/2009	Added Section 1.1, “Devices in the MC9RS08KB12 Series.”
3	2/1/2010	Updated the ADC feature in the feature list.
4	1/30/2012	Added 24-pin QFN package information.

This product incorporates SuperFlash® technology licensed from SST.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
© Freescale Semiconductor, Inc., 2008-2012. All rights reserved.

List of Chapters

Chapter Number	Title	Page
Chapter 1	Device Overview	19
Chapter 2	Pins and Connections	23
Chapter 3	Modes of Operation	29
Chapter 4	Memory	35
Chapter 5	Resets, Interrupts, and General System Control	49
Chapter 6	Parallel Input/Output Control	65
Chapter 7	Keyboard Interrupt (RS08KBIV1)	77
Chapter 8	Central Processor Unit (RS08CPUV1)	83
Chapter 9	16-Bit Timer/PWM (S08TPMV3)	101
Chapter 10	Serial Communications Interface (S08SCIV4)	129
Chapter 11	Modulo Timer (S08MTIMV1)	149
Chapter 12	10-Bit Analog-to-Digital Converter (S08ADC10V1)	159
Chapter 13	Analog Comparator (RS08ACMPV1)	187
Chapter 14	Inter-Integrated Circuit (S08IICV2)	195
Chapter 15	Internal Clock Source (S08ICSV1)	213
Chapter 16	Development Support	225

Contents

Section Number	Title	Page

Chapter 1
Device Overview

1.1 Devices in the MC9RS08KB12 Series 19
1.2 MCU Block Diagram 19
1.3 System Clock Distribution 21

Chapter 2
Pins and Connections

2.1 Introduction 23
2.2 Device Pin Assignment 23
2.3 Recommended System Connections 24
2.4 Pin Detail 25
2.4.1 Power Pins (VDD, VSS) 25
2.4.2 PTA5/TCLK/RESET/V_{PP} Pin 26
2.4.3 PTA4/ACMPO/BKGD/MS Pin 26
2.4.4 Oscillator (XTAL, EXTAL) 27
2.4.5 General-Purpose I/O and Peripheral Ports 27

Chapter 3
Modes of Operation

3.1 Introduction 29
3.2 Features 29
3.3 Run Mode 29
3.4 Active Background Mode 29
3.5 Wait Mode 30
3.6 Stop Mode 31
3.6.1 Active BDM Enabled in Stop Mode 32
3.6.2 LVD Enabled in Stop Mode 32

Chapter 4
Memory

4.1 Memory Map 35
4.2 Unimplemented Memory 38
4.3 Indexed/Indirect Addressing 38
4.4 RAM and Register Addresses and Bit Assignments 38
4.5 RAM 42
4.6 Flash 42
4.6.1 Features 42
4.6.2 Flash Programming Procedure 43

4.6.3 Flash Mass Erase Operation 43
4.6.4 Security 44
4.7 Flash Registers and Control Bits 45
4.7.1 Flash Options Register (FOPT and NVOPT) 45
4.7.2 Flash Control Register (FLCR) 45
4.8 Page Select Register (PAGESEL) 46

Chapter 5
Resets, Interrupts, and General System Control

5.1 Introduction 49
5.2 Features 49
5.3 MCU Reset 49
5.4 Computer Operating Properly (COP) Watchdog 50
5.5 Interrupts 51
5.5.1 Interrupt Operation 51
5.5.2 Interrupt Operation in Wait Mode 52
5.5.3 Interrupt Operation Stop Mode 52
5.6 Low-Voltage Detect (LVD) System 52
5.6.1 Power-On Reset Operation 53
5.6.2 LVD Reset Operation 53
5.6.3 LVD Interrupt Operation 53
5.7 Real-Time Interrupt (RTI) 53
5.8 Reset, Interrupt, and System Control Registers and Control Bits 53
5.8.1 System Reset Status Register (SRS) 53
5.8.2 System Options Register (SOPT1) 54
5.8.3 System Options Register 2 (SOPT2) 56
5.8.4 System Device Identification Register (SDIDH, SDIDL) 56
5.8.5 System Real-Time Interrupt Status and Control Register (SRTISC) 57
5.8.6 System Power Management Status and Control 1 Register (SPMSC1) 59
5.8.7 System Interrupt Pending Register 1 (SIP1) 60
5.8.8 System Interrupt Pending Register 2 (SIP2) 61
5.8.9 Interrupt Exit Address Register (IEA) 62
5.8.10 Interrupt Return Address Registers (IRA = IRAH:IRAL) 62

Chapter 6
Parallel Input/Output Control

6.1 Pin Behavior in Low-Power Modes 66
6.2 Parallel I/O Registers 66
6.2.1 Port A Registers 66
6.2.2 Port B Registers 67
6.2.3 Port C Registers 68
6.3 Pin Control Registers 69
6.3.1 Port A Pin Control Registers 69
6.3.1.1 Internal Pulling Device Enable 69
6.3.1.2 Pullup/Pulldown Control 70

6.3.1.3 Output Slew Rate Control Enable 70
6.3.1.4 Port A Drive Strength Selection Register (PTADS) 71
6.3.2 Port B Pin Control Registers 71
6.3.2.1 Internal Pulling Device Enable 71
6.3.2.2 Pullup/Pulldown Control 72
6.3.2.3 Output Slew Rate Control Enable 72
6.3.2.4 Port B Drive Strength Selection Register (PTBDS) 73
6.3.3 Port C Pin Control Registers 73
6.3.3.1 Internal Pulling Device Enable 73
6.3.3.2 Pullup/Pulldown Control 74
6.3.3.3 Output Slew Rate Control Enable 74
6.3.3.4 Port C Drive Strength Selection Register (PTCDS) 75

Chapter 7
Keyboard Interrupt (RS08KBIV1)

7.1 Introduction 77
7.1.1 Features 78
7.1.2 Modes of Operation 78
7.1.2.1 Operation in Wait Mode 78
7.1.2.2 Operation in Stop Mode 78
7.1.2.3 Operation in Active Background Mode 78
7.1.3 Block Diagram 78
7.2 External Signal Description 79
7.3 Register Definition 79
7.3.1 KBI Status and Control Register (KBISC) 80
7.3.2 KBI Pin Enable Register (KBIPE) 80
7.3.3 KBI Edge Select Register (KBIES) 81
7.4 Functional Description 81
7.4.1 Edge Only Sensitivity 81
7.4.2 Edge and Level Sensitivity 82
7.4.3 KBI Pullup/Pulldown Resistors 82
7.4.4 KBI Initialization 82

Chapter 8
Central Processor Unit (RS08CPUV1)

8.1 Introduction 83
8.2 Programmer’s Model and CPU Registers 83
8.2.1 Accumulator (A) 84
8.2.2 Program Counter (PC) 85
8.2.3 Shadow Program Counter (SPC) 85
8.2.4 Condition Code Register (CCR) 85
8.2.5 Indexed Data Register (D[X]) 86
8.2.6 Index Register (X) 86
8.2.7 Page Select Register (PAGESEL) 87
8.3 Addressing Modes 87

8.3.1 Inherent Addressing Mode (INH) 87
8.3.2 Relative Addressing Mode (REL) 87
8.3.3 Immediate Addressing Mode (IMM) 88
8.3.4 Tiny Addressing Mode (TNY) 88
8.3.5 Short Addressing Mode (SRT) 89
8.3.6 Direct Addressing Mode (DIR) 89
8.3.7 Extended Addressing Mode (EXT) 89
8.3.8 Indexed Addressing Mode (IX, Implemented by Pseudo Instructions) 89
8.4 Special Operations 89
8.4.1 Reset Sequence 90
8.4.2 Interrupts 90
8.4.3 Wait and Stop Mode 90
8.4.4 Active Background Mode 90
8.5 Summary Instruction Table 91

Chapter 9
16-Bit Timer/PWM (S08TPMV3)

9.1 Introduction 101
9.1.1 Features 103
9.1.2 Modes of Operation 103
9.1.3 Block Diagram 104
9.2 Signal Description 106
9.2.1 Detailed Signal Descriptions 106
9.2.1.1 EXTCLK — External Clock Source 107
9.2.1.2 TPMCHn — TPM Channel n I/O Pin(s) 107
9.3 Register Definition 110
9.3.1 TPM Status and Control Register (TPMSC) 110
9.3.2 TPM-Counter Registers (TPMCNTH:TPMCNTL) 111
9.3.3 TPM Counter Modulo Registers (TPMMODH:TPMMODL) 112
9.3.4 TPM Channel n Status and Control Register (TPMCnSC) 113
9.3.5 TPM Channel Value Registers (TPMCnVH:TPMCnVL) 114
9.4 Functional Description 116
9.4.1 Counter 116
9.4.1.1 Counter Clock Source 116
9.4.1.2 Counter Overflow and Modulo Reset 117
9.4.1.3 Counting Modes 118
9.4.1.4 Manual Counter Reset 118
9.4.2 Channel Mode Selection 118
9.4.2.1 Input Capture Mode 118
9.4.2.2 Output Compare Mode 118
9.4.2.3 Edge-Aligned PWM Mode 119
9.4.2.4 Center-Aligned PWM Mode 120
9.5 Reset Overview 121
9.5.1 General 121
9.5.2 Description of Reset Operation 121

9.6 Interrupts 121
9.6.1 General 121
9.6.2 Description of Interrupt Operation 122
9.6.2.1 Timer Overflow Interrupt (TOF) Description 122
9.6.2.2 Channel Event Interrupt Description 123

Chapter 10
Serial Communications Interface (S08SCIV4)

10.1 Introduction 129
10.1.1 Features 131
10.1.2 Modes of Operation 131
10.1.3 Block Diagram 132
10.2 Register Definition 134
10.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL) 134
10.2.2 SCI Control Register 1 (SCIC1) 135
10.2.3 SCI Control Register 2 (SCIC2) 136
10.2.4 SCI Status Register 1 (SCIS1) 137
10.2.5 SCI Status Register 2 (SCIS2) 139
10.2.6 SCI Control Register 3 (SCIC3) 140
10.2.7 SCI Data Register (SCID) 141
10.3 Functional Description 141
10.3.1 Baud Rate Generation 141
10.3.2 Transmitter Functional Description 142
10.3.2.1 Send Break and Queued Idle 143
10.3.3 Receiver Functional Description 143
10.3.3.1 Data Sampling Technique 144
10.3.3.2 Receiver Wakeup Operation 144
10.3.4 Interrupts and Status Flags 145
10.3.5 Additional SCI Functions 146
10.3.5.1 8- and 9-Bit Data Modes 146
10.3.5.2 Stop Mode Operation 147
10.3.5.3 Loop Mode 147
10.3.5.4 Single-Wire Operation 147

Chapter 11
Modulo Timer (S08MTIMV1)

11.1 Introduction 149
11.1.1 Features 151
11.1.2 Modes of Operation 151
11.1.2.1 MTIM in Wait Mode 151
11.1.2.2 MTIM in Stop Modes 151
11.1.2.3 MTIM in Active Background Mode 151
11.1.3 Block Diagram 152
11.2 External Signal Description 152
11.3 Register Definition 153

11.3.1 MTIMx Status and Control Register (MTIMxSC) 154
11.3.2 MTIMx Clock Configuration Register (MTIMxCLK) 155
11.3.3 MTIMx Counter Register (MTIMxCNT) 156
11.3.4 MTIMx Modulo Register (MTIMxMOD) 156
11.4 Functional Description 157
11.4.1 MTIM Operation Example 158

Chapter 12
10-Bit Analog-to-Digital Converter (S08ADC10V1)

12.1 Introduction 159
12.1.1 Module Configurations 160
12.1.1.1 Analog Supply and Voltage Reference Connections 160
12.1.1.2 Alternate Clock 161
12.1.1.3 Hardware Trigger 161
12.1.1.4 Temperature Sensor 161
12.1.1.5 Channel Assignment 162
12.1.1.6 Low-Power Mode Operation 162
12.1.2 Features 163
12.1.3 Block Diagram 163
12.2 External Signal Description 164
12.2.1 Analog Power (V_{DDAD}) 165
12.2.2 Analog Ground (V_{SSAD}) 165
12.2.3 Voltage Reference High (V_{REFH}) 165
12.2.4 Voltage Reference Low (V_{REFL}) 165
12.2.5 Analog Channel Inputs (ADx) 165
12.3 Register Definition 165
12.3.1 Status and Control Register 1 (ADCSC1) 165
12.3.2 Status and Control Register 2 (ADCSC2) 167
12.3.3 Data Result High Register (ADCRH) 168
12.3.4 Data Result Low Register (ADCRL) 168
12.3.5 Compare Value High Register (ADCCVH) 169
12.3.6 Compare Value Low Register (ADCCVL) 169
12.3.7 Configuration Register (ADCCFG) 169
12.3.8 Pin Control 1 Register (APCTL1) 171
12.3.9 Pin Control 2 Register (APCTL2) 172
12.3.10Pin Control 3 Register (APCTL3) 173
12.4 Functional Description 174
12.4.1 Clock Select and Divide Control 174
12.4.2 Input Select and Pin Control 175
12.4.3 Hardware Trigger 175
12.4.4 Conversion Control 175
12.4.4.1 Initiating Conversions 175
12.4.4.2 Completing Conversions 176
12.4.4.3 Aborting Conversions 176
12.4.4.4 Power Control 176

12.4.4.5 Total Conversion Time 176
12.4.5 Automatic Compare Function 178
12.4.6 MCU Wait Mode Operation 178
12.4.7 MCU Stop3 Mode Operation 178
12.4.7.1 Stop3 Mode With ADACK Disabled 178
12.4.7.2 Stop3 Mode With ADACK Enabled 179
12.4.8 MCU Stop1 and Stop2 Mode Operation 179
12.5 Initialization Information 179
12.5.1 ADC Module Initialization Example 179
12.5.1.1 Initialization Sequence 179
12.5.1.2 Pseudo — Code Example 180
12.6 Application Information 181
12.6.1 External Pins and Routing 181
12.6.1.1 Analog Supply Pins 181
12.6.1.2 Analog Reference Pins 182
12.6.1.3 Analog Input Pins 182
12.6.2 Sources of Error 183
12.6.2.1 Sampling Error 183
12.6.2.2 Pin Leakage Error 183
12.6.2.3 Noise-Induced Errors 183
12.6.2.4 Code Width and Quantization Error 184
12.6.2.5 Linearity Errors 184
12.6.2.6 Code Jitter, Non-Monotonicity and Missing Codes 184

Chapter 13
Analog Comparator (RS08ACMPV1)

13.1 Introduction 187
13.1.1 Features 189
13.1.2 Modes of Operation 189
13.1.2.1 Operation in Wait Mode 189
13.1.2.2 Operation in Stop Mode 189
13.1.2.3 Operation in Active Background Mode 189
13.1.3 Block Diagram 189
13.2 External Signal Description 191
13.3 Register Definition 191
13.3.1 ACMP Status and Control Register (ACMPSC) 191
13.4 Functional Description 192

Chapter 14
Inter-Integrated Circuit (S08IICV2)

14.1 Introduction 195
14.1.1 Module Configuration 195
14.1.2 Features 197
14.1.3 Modes of Operation 197
14.1.4 Block Diagram 197

14.2 External Signal Description 198
14.2.1 SCL — Serial Clock Line 198
14.2.2 SDA — Serial Data Line 198
14.3 Register Definition 198
14.3.1 IIC Address Register (IICA) 199
14.3.2 IIC Frequency Divider Register (IICF) 199
14.3.3 IIC Control Register (IICC1) 202
14.3.4 IIC Status Register (IICS) 202
14.3.5 IIC Data I/O Register (IICD) 203
14.3.6 IIC Control Register 2 (IICC2) 204
14.4 Functional Description 205
14.4.1 IIC Protocol 205
14.4.1.1 Start Signal 205
14.4.1.2 Slave Address Transmission 206
14.4.1.3 Data Transfer 206
14.4.1.4 Stop Signal 206
14.4.1.5 Repeated Start Signal 207
14.4.1.6 Arbitration Procedure 207
14.4.1.7 Clock Synchronization 207
14.4.1.8 Handshaking 208
14.4.1.9 Clock Stretching 208
14.4.2 10-bit Address 208
14.4.2.1 Master-Transmitter Addresses a Slave-Receiver 208
14.4.2.2 Master-Receiver Addresses a Slave-Transmitter 208
14.4.3 General Call Address 209
14.5 Resets 209
14.6 Interrupts 209
14.6.1 Byte Transfer Interrupt 209
14.6.2 Address Detect Interrupt 210
14.6.3 Arbitration Lost Interrupt 210
14.7 Initialization/Application Information 211

Chapter 15
Internal Clock Source (S08ICSV1)

15.1 Introduction 213
15.1.1 Features 215
15.1.2 Modes of Operation 215
15.1.2.1 FLL Engaged Internal (FEI) 215
15.1.2.2 FLL Engaged External (FEE) 215
15.1.2.3 FLL Bypassed Internal (FBI) 215
15.1.2.4 FLL Bypassed Internal Low Power (FBILP) 215
15.1.2.5 FLL Bypassed External (FBE) 216
15.1.2.6 FLL Bypassed External Low Power (FBELP) 216
15.1.2.7 Stop (STOP) 216
15.1.3 Block Diagram 216

15.2 External Signal Description 217
15.3 Register Definition 217
15.3.1 ICS Control Register 1 (ICSC1) 217
15.3.2 ICS Control Register 2 (ICSC2) 218
15.3.3 ICS Trim Register (ICSTRM) 219
15.3.4 ICS Status and Control (ICSSC) 219
15.4 Functional Description 220
15.4.1 Operational Modes 220
15.4.1.1 FLL Engaged Internal (FEI) 220
15.4.1.2 FLL Engaged External (FEE) 221
15.4.1.3 FLL Bypassed Internal (FBI) 221
15.4.1.4 FLL Bypassed Internal Low Power (FBILP) 221
15.4.1.5 FLL Bypassed External (FBE) 221
15.4.1.6 FLL Bypassed External Low Power (FBELP) 222
15.4.1.7 Stop 222
15.4.2 Mode Switching 222
15.4.3 Bus Frequency Divider 222
15.4.4 Low Power Bit Usage 223
15.4.5 Internal Reference Clock 223
15.4.6 Optional External Reference Clock 223
15.4.7 Fixed Frequency Clock 223

Chapter 16
Development Support

16.1 Introduction 225
16.1.1 Features 225
16.2 RS08 Background Debug Controller (BDC) 226
16.2.1 BKGD Pin Description 227
16.2.2 Communication Details 227
16.2.3 SYNC and Serial Communication Timeout 230
16.3 BDC Registers and Control Bits 231
16.3.1 BDC Status and Control Register (BDCSCR) 231
16.3.2 BDC Breakpoint Match Register 232
16.4 RS08 BDC Commands 233
16.4.1 BDC_RESET 236
16.4.2 BACKGROUND 236
16.4.3 READ_STATUS 237
16.4.4 WRITE_CONTROL 237
16.4.5 READ_BYTE 238
16.4.6 READ_BYTE_WS 239
16.4.7 WRITE_BYTE 239
16.4.8 WRITE_BYTE_WS 240
16.4.9 READ_BKPT 240
16.4.10 WRITE_BKPT 240
16.4.11 GO 241

16.4.12TRACE1 241
16.4.13READ_BLOCK 241
16.4.14WRITE_BLOCK 242
16.4.15READ_A 242
16.4.16WRITE_A 242
16.4.17READ_CCR_PC 243
16.4.18WRITE_CCR_PC 243
16.4.19READ_SPC 244
16.4.20WRITE_SPC 244
16.5 BDC Hardware Breakpoint 244
16.6 BDM in Stop and Wait Modes 245
16.7 BDC Command Execution 245

Chapter 1
Device Overview

The MC9RS08KB12 series microcontroller units (MCU) are extremely low-cost, small pin count, high performance devices for home appliances and medical equipment. As general purpose microcontrollers, these devices are composed of standard on-chip modules including a very small and highly efficient RS08 CPU core of version 2 which supports single global interrupt vector, 254/126 bytes RAM, 12 /8 /4 /2 KB flash, two 8-bit modulo timers (MTIM), one 12-channel 10-bit analog-to-digital converter (ADC), one 2-channel 16-bit timer/PWM (TPM), one inter-integrated circuit bus module (IIC), one keyboard interrupt (KBI), one analog comparator (ACMP), and one serial communication interface (SCI). These devices are available in 24-pin, 20-pin, 16-pin or 8-pin packages.

1.1 Devices in the MC9RS08KB12 Series

Table 1-1 summarizes the feature set available in the MC9RS08KB12 series of MCUs.

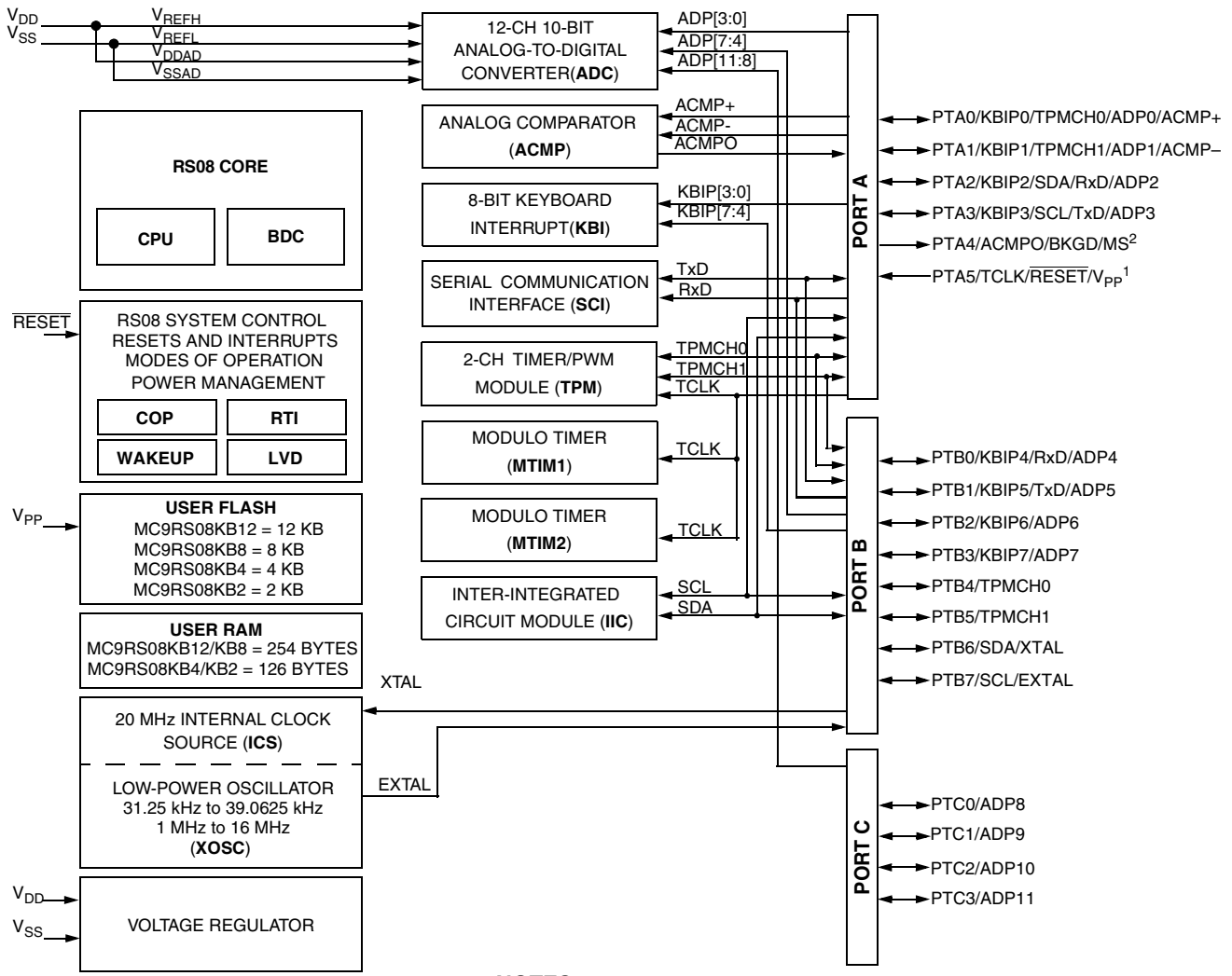
Table 1-1. . MC9RS08KB12 Series Features by MCU and Package

Feature	MC9RS08KB12	MC9RS08KB8		MC9RS08KB4		MC9RS08KB2
FLASH size (KB)	12	8		4		2
RAM size (bytes)	254	254		126		126
Pin Quantity	20	20	16	20	16	8
ACMP	yes	yes		yes		yes
ADC	12-ch	12-ch	8-ch	12-ch	8-ch	4-ch
ICS	yes	yes		yes		yes
IIC	yes	yes		yes		yes[1]
KBI	8-pin	8-pin		8-pin		4-pin
MTIM	2	2		2		2
RTI	yes	yes		yes		yes
SCI	yes	yes		yes		yes[1]
TPM	2-ch	2-ch		2-ch		2-ch
XOCS	yes	yes		yes		NA
I/O pins	18	18	14	18	14	6

[1] In MC9RS08KB2, IIC and SCI cannot function simultaneously.

1.2 MCU Block Diagram

The block diagram, Figure 1-1, shows the MC9RS08KB12 MCU structure.



NOTES:

1. PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} is an input-only pin when used as port pin
2. PTA4/ACMPO/BKGD/MS is an output-only pin when used as port pin

Figure 1-1. MC9RS08KB12 Series Block Diagram

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

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

Module		Version
Central Processing Unit	(CPU)	2
Internal Clock Source	(ICS)	1
Oscillator	(XOSC)	1
Keyboard Interrupt	(KBI)	1
Analog-to-Digital Converter	(ADC10)	1
Analog Comparator	(ACMP)	1

Table 1-2. Versions of the On-Chip Modules (continued)

Module		Version
Serial Communications Interface	(SCI)	4
Inter-Integrated Circuit	(IIC)	2
Timer Pulse-Width Modulator	(TPM)	3
Modulo Timer	(MTIM)	1

1.3 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram for the MCU. The bus clock (BUSCLK) frequency is half the ICS output (ICSOUT) frequency and used by all internal modules.

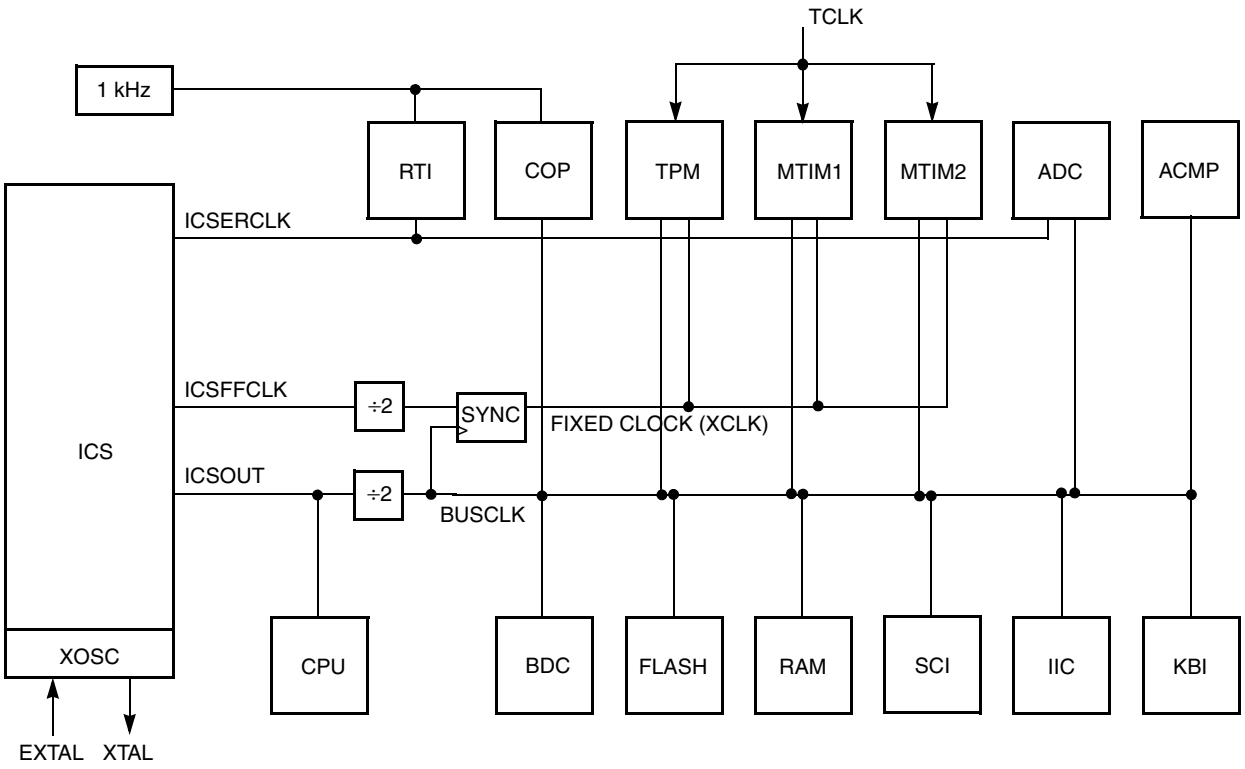


Figure 1-2. System Clock Distribution Diagram

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) output

See Chapter 15, “Internal Clock Source (S08ICSV1),” for details on configuring the ICSOUT clock.

- ICSERCLK — This external reference clock can be selected as the alternate clock for the ADC and RTI modules. Section 15.4.6, “Optional External Reference Clock,” explains the ICSERCLK in more detail. See Chapter 12, “10-Bit Analog-to-Digital Converter (S08ADC10V1),” for more information regarding the use of ICSERCLK with this module.
- ICSFFCLK — This clock can be selected as the alternate clock for TPM and MTIM modules. 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 15.4.7, “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 9, “16-Bit Timer/PWM (S08TPMV3),” and Chapter 11, “Modulo Timer (S08MTIMV1),” 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 and RTI modules. See Section 5.4, “Computer Operating Properly (COP) Watchdog,” for details on using the 1 kHz clock with these modules.

Chapter 2
Pins and Connections

2.1 Introduction

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

2.2 Device Pin Assignment

Figure 2-1, Figure 2-2, Figure 2-3 and Figure 2-4 show the pin assignments in the packages of the MC9RS08KB12 series.

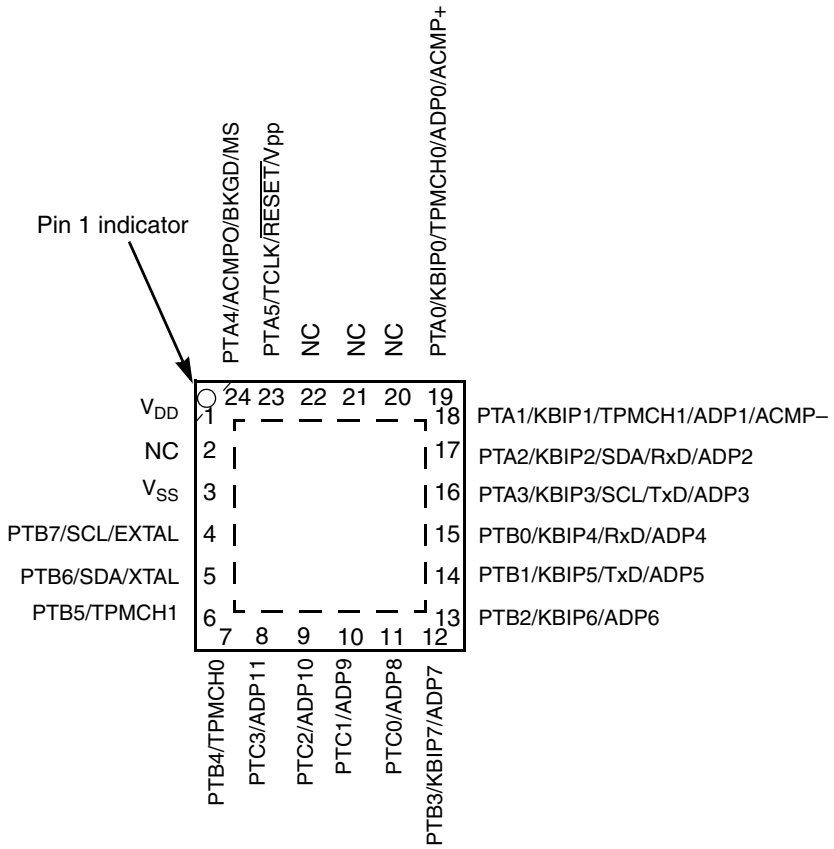


Figure 2-1. MC9RS08KB12 Series 24-Pin QFN Package

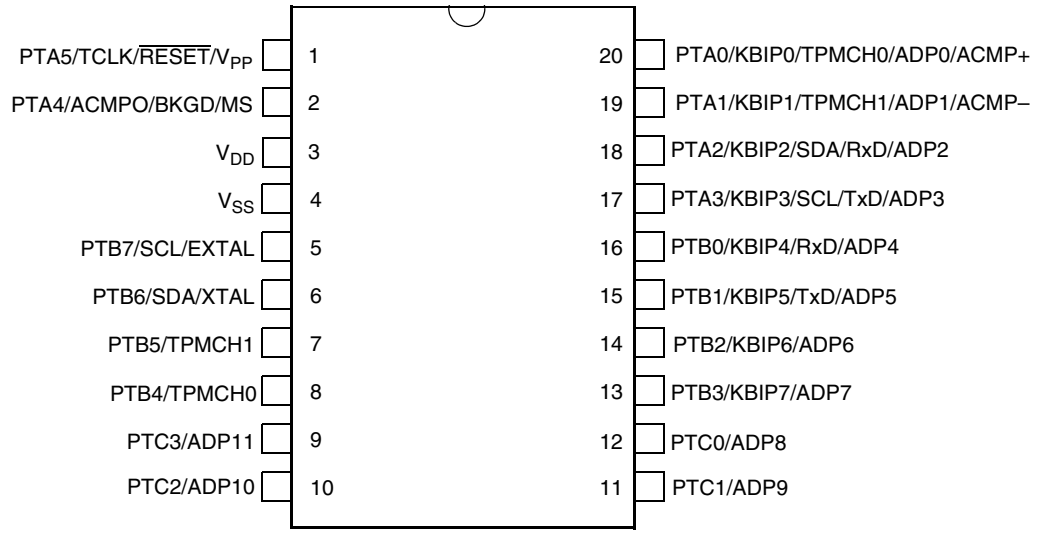


Figure 2-2. MC9RS08KB12 Series 20-Pin SOIC Package

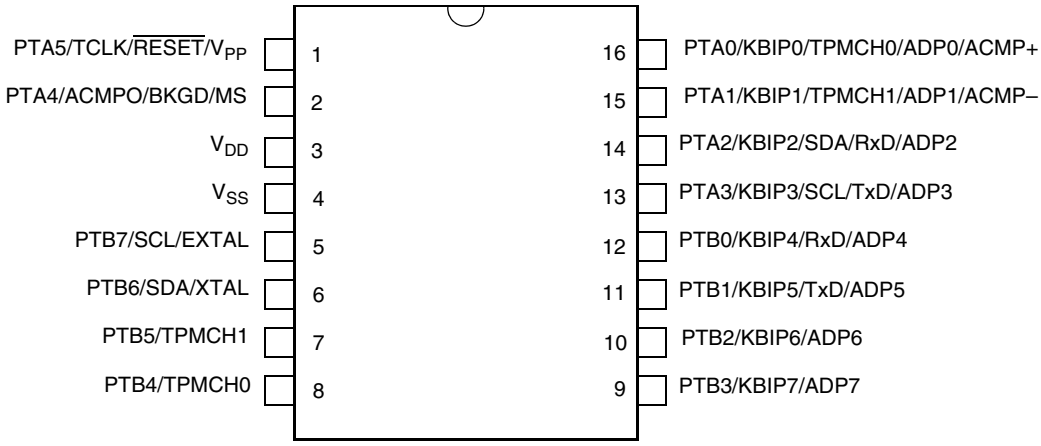


Figure 2-3. MC9RS08KB12 Series 16-Pin SOIC NB/TSSOP Package

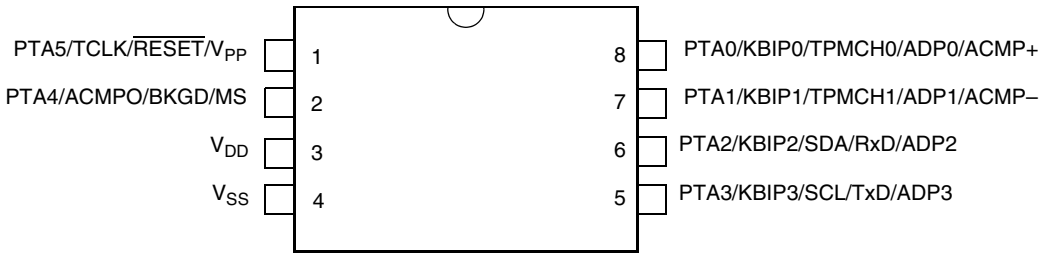
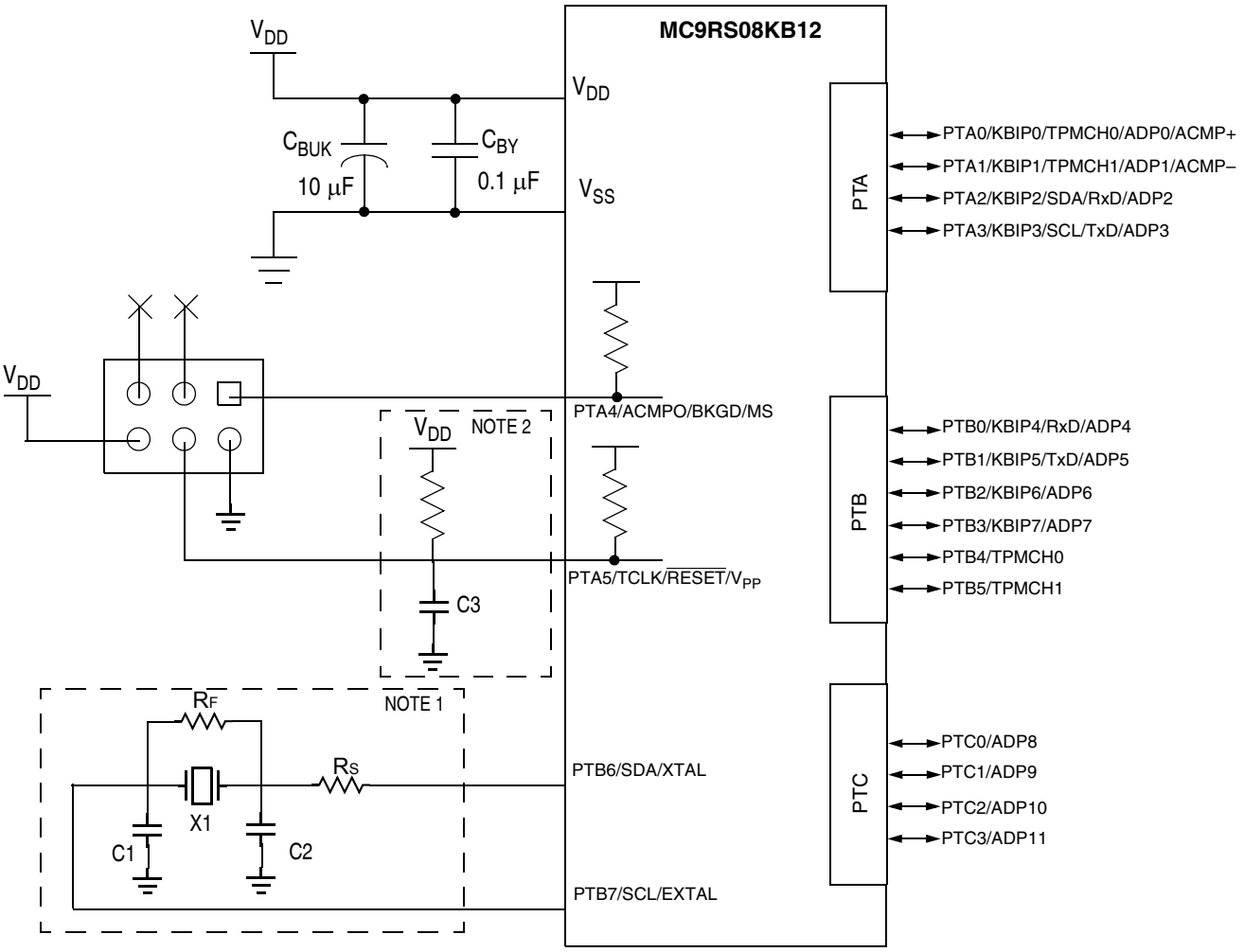


Figure 2-4. MC9RS08KB12 Series 8-Pin SOIC/DFN Package

2.3 Recommended System Connections

Figure 2-5 shows the reference connection for background debug and flash programming.



NOTES:
1 External crystal circuit not required if using the internal clock option.
2 RC filter on $\overline{\text{RESET}}$ pin recommended for noisy environment.

Figure 2-5. Reference System Connection Diagram

2.4 Pin Detail

This section provides a detailed description of system connections.

2.4.1 Power Pins (V_{DD}, V_{SS})

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 a regulated lower-voltage source to the CPU and other MCU internal circuitry.

Typically, application systems have two separate capacitors across the power pins: a bulk electrolytic capacitor such as a 10 µF tantalum capacitor, to provide bulk charge storage for the overall system, and a

bypass capacitor such as a 0.1 µF ceramic capacitor, located as near to the MCU power pins as practical to suppress high-frequency noise.

2.4.2 PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} Pin

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

External V_{PP} voltage (typically 12 V, see MC9RS08KB12 Series *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 does not occur.

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 does not occur.

2.4.3 PTA4/ACMPO/BKGD/MS Pin

The background/mode select function is shared with an output-only pin on PTA4 pin and the optional analog comparator output. During a power-on-reset (POR) or background debug force reset, the pin functions as a mode selection pin. Immediately after any reset rises, the pin functions as the background pin and can be used for background debug communication. While functioning as a background / mode selection pin, this pin has an internal pullup device enabled.

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

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of the internal reset after a POR or force BDC reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during the power-on-reset (POR) or immediately after issuing a background debug force reset, which will force the MCU to active background mode.

The BKGD pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock equals the bus clock rate; therefore, significant capacitance must not be 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 the internal pullup device do not affect rise and fall times on the BKGD pin.

2.4.4 Oscillator (XTAL, EXTAL)

Immediately after reset, the MCU uses an internally generated clock provided by the clock source generator (ICS) module. The default ICS settings will provide a 4 MHz bus clock out of reset.

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

Refer to Figure 2-5 for the following discussion. RS (when used) and RF 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.

RF 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 to 10 M. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) may 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. Printed circuit board (PCB) capacitance and MCU pin capacitance must be taken into account 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).

2.4.5 General-Purpose I/O and Peripheral Ports

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

NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program must enable on-chip pullup/pulldown devices or change the direction of unused pins to outputs.

Table 1. Pin Availability by Package Pin-Count

Pin Number				<-- Lowest **Priority** --> Highest				
24	**20**	**16**	**8**	**Port Pin**	**Alt 1**	**Alt 2**	**Alt 3**	**Alt 4**
1	3	3	3					V_{DD}
2	—	—	—	NC				
3	4	4	4					V_{SS}
4	5	5	—	PTB7	SCL[1]			EXTAL
5	6	6	—	PTB6	SDA[1]			XTAL

Table 1. Pin Availability by Package Pin-Count

Pin Number				<-- Lowest **Priority** --> Highest				
24	**20**	**16**	**8**	**Port Pin**	**Alt 1**	**Alt 2**	**Alt 3**	**Alt 4**
6	7	7	—	PTB5	TPMCH1[2]			
7	8	8	—	PTB4	TPMCH0[2]			
8	9	—	—	PTC3			ADP11	
9	10	—	—	PTC2			ADP10	
10	11	—	—	PTC1			ADP9	
11	12	—	—	PTC0			ADP8	
12	13	9	—	PTB3	KBIP7		ADP7	
13	14	10	—	PTB2	KBIP6		ADP6	
14	15	11	—	PTB1	KBIP5	TxD[3]	ADP5	
15	16	12	—	PTB0	KBIP4	RxD[3]	ADP4	
16	17	13	5	PTA3	KBIP3	SCL[1]	TxD[3]	ADP3
17	18	14	6	PTA2	KBIP2	SDA[1]	RxD[3]	ADP2
18	19	15	7	PTA1	KBIP1	TPMCH1[2]	ADP1	ACMP–
19	20	16	8	PTA0	KBIP0	TPMCH0[2]	ADP0	ACMP+
20	—	—	—	NC				
21	—	—	—	NC				
22	—	—	—	NC				
23	1	1	1	PTA5		TCLK	$\overline{\text{RESET}}$	V_{PP}
24	2	2	2	PTA4	ACMPO	BKGD	MS	

1 IIC pins can be remapped to PTB6 and PTB7, default reset location is PTA2 and PTA3. It can be configured only once.

2 TPM pins can be remapped to PTB4 and PTB5, default reset location is PTA0 and PTA1.

3 SCI pins can be remapped to PTA2 and PTA3, default reset location is PTB0 and PTB1. It can be configured only once.

Chapter 3
Modes of Operation

3.1 Introduction

This chapter describes the MC9RS08KB12 series operating modes. It also details entry into each mode, exit from each mode, and functionality while in each of the modes.

3.2 Features

- Active background mode for code development
- Wait mode:
  - CPU shuts down to conserve power
  - System clocks continue running
  - Full voltage regulation is maintained
- Stop mode:
  - System clocks are stopped
  - All internal circuits remain powered for fast recovery

3.3 Run Mode

Run mode is the normal operating mode for the MC9RS08KB12 series. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory beginning at the address $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 where the user program starts. Instead of using 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 power-on-reset (POR) or immediately after issuing a background debug force reset (BDC_RESET) command
- When a BACKGROUND command is received through the BKGD pin
- When a BGND instruction is executed

- When a BDC breakpoint is encountered

After active background mode is entered, the CPU stays 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 — Commands that can be issued while the user program is running, 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
  - BACKGROUND command
- Active background commands — Can be executed only while the MCU is in active background mode, 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)

Active background mode is used to program user application code into the flash program memory before the MCU is operated in run mode for the first time. When the MC9RS08KB12 series are shipped, the flash program memory is usually erased so no program can be executed in run mode until the flash memory is initially programmed. The active background mode can also be used to erase and reprogram the flash memory after it is programmed.

For additional information about active background mode, refer to the Chapter 16, “Development Support.”

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The program counter (PC) is halted at the position where the WAIT instruction executes.The RS08 supports wakeup from WAIT differently depending on the state of IMASK in SIP2.

If IMASK = 1, interrupt vectoring is disabled. When an interrupt pending flag becomes set, the MCU exits wait mode and resumes processing at the next instruction.

If IMASK = 0, interrupt vectoring is enabled. When an interrupt pending flag becomes set, the MCU exits wait mode, the address of the instruction following the WAIT is stored in IRA, PC is loaded with 0x3FF7 and code execution begins.

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. 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 summarizes the behavior of the MCU in wait mode.

Table 3-1. Wait Mode Behavior

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

3.6 Stop Mode

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 modules are halted. If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU does 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[1]	ICS[2]	RTI	ADC[3]	ACMP[4]	Digital Peripherals	I/O Pins
Stop	Standby	Optionally on	Optionally on	Optionally on	Optionally on	Optionally on	Standby	States held

1 When BDM is enabled, the regulator is on. Or else, only when both LVDE and LVDSE bits in the SPMSC1 to be set, regulator is in on mode.

2 ICS requires IREFSTEN = 1 and LVDE and LVDSE must be set to allow operation in stop.

3 Requires the asynchronous ADC clock, both LVDE and LVDSE bits in the SPMSC1 to be set, otherwise ADC is in standby mode.

4 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 clocks in the MCU are halted. The ICS is turned off by default when the IREFSTEN bit is cleared and the voltage regulator enters standby. The states of all of the internal registers and logic, as well as the RAM content, are maintained.

Exit from stop is done by asserting reset, any enabled asynchronous interrupt, or the real-time interrupt. The asynchronous interrupts are the KBI pins, LVD interrupt, ADC interrupt, SCI edge detect interrupt, or the ACMP interrupt.

Stop is exited by asserting RESET or any asynchronous interrupt that is enabled. If stop is exited by asserting the RESET pin, the MCU will be reset and program execution starts at location $3FFD. If exited by means of an asynchronous interrupt, the sequence varies depending on the state of IMASK in SIP2.

If IMASK = 1, interrupt vectoring is disabled. When an asynchronous interrupt request occurs. the MCU exits stop mode and resumes processing at the next instruction.

If IMASK = 0, interrupt vectoring is enabled. When an interrupt pending flag becomes set, the MCU exits stop mode, the address of the instruction following the STOP is stored in IRA, PC is loaded with 0x3FF7 and code execution begins.

A separate self-clocked source (≈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 RTICLKS = 1, the internal 1 kHz oscillator is disabled and power consumption is lower.

The external clock source can also be enabled for the real-time interrupt to allow a wakeup from stop mode with no external components. Setting the ERCLKEN = 1 and EREFSTEN=1 enables the external clock source when in STOP mode.

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.

To enable XOSC to operate with an external reference clock source in STOP mode, LVD must be enabled by setting LVDE and LVDSE.

3.6.1 Active BDM Enabled in Stop Mode

Entry into active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. This register is described in the Chapter 16, "Development Support." If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode so background debug communication is still possible. The voltage regulator does not enter its low-power standby state. It maintains full internal regulation.

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access. 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 stop and enter active background mode if the ENBDM bit is set. After active background mode is entered, all background commands are available.

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

Table 3-3. BDM Enabled Stop Mode Behavior

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

3.6.2 LVD Enabled in Stop Mode

The LVD system can generate an interrupt or a reset when the supply voltage drops below the LVD voltage. The voltage regulator remains active if the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) when the CPU executes a STOP instruction.

Table 3-4 summarizes the behavior of the MCU in stop when LVD is enabled.

Table 3-4. LVD Enabled Stop Mode Behavior

Mode	CPU	Regulator	ICS	RTI	ADC[1]	ACMP	Digital Peripherals	I/O Pins
Stop	Standby	On	Optionally on	Optionally on	Optionally on	Optionally on	Standby	States held

1 Requires the asynchronous ADC clock to be enabled.

Chapter 4
Memory

4.1 Memory Map

The memory map of the MCU is 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 MC9RS08KB12 and MC9RS08KB8 ($0050–$00BF, $0100–$017F)
- RAM for MC9RS08KB4 and MC9RS08KB2 ($0050–$00BF)
- Paging window ($00C0–$00FF)
- Other peripheral registers ($0200–$023F)
- Nonvolatile memory for MC9RS08KB12 ($1000–$3FFF)
- Nonvolatile memory for MC9RS08KB8 ($2000–$3FFF)
- Nonvolatile memory for MC9RS08KB4 ($3000–$3FFF)
- Nonvolatile memory for MC9RS08KB2 ($3800–$3FFF)

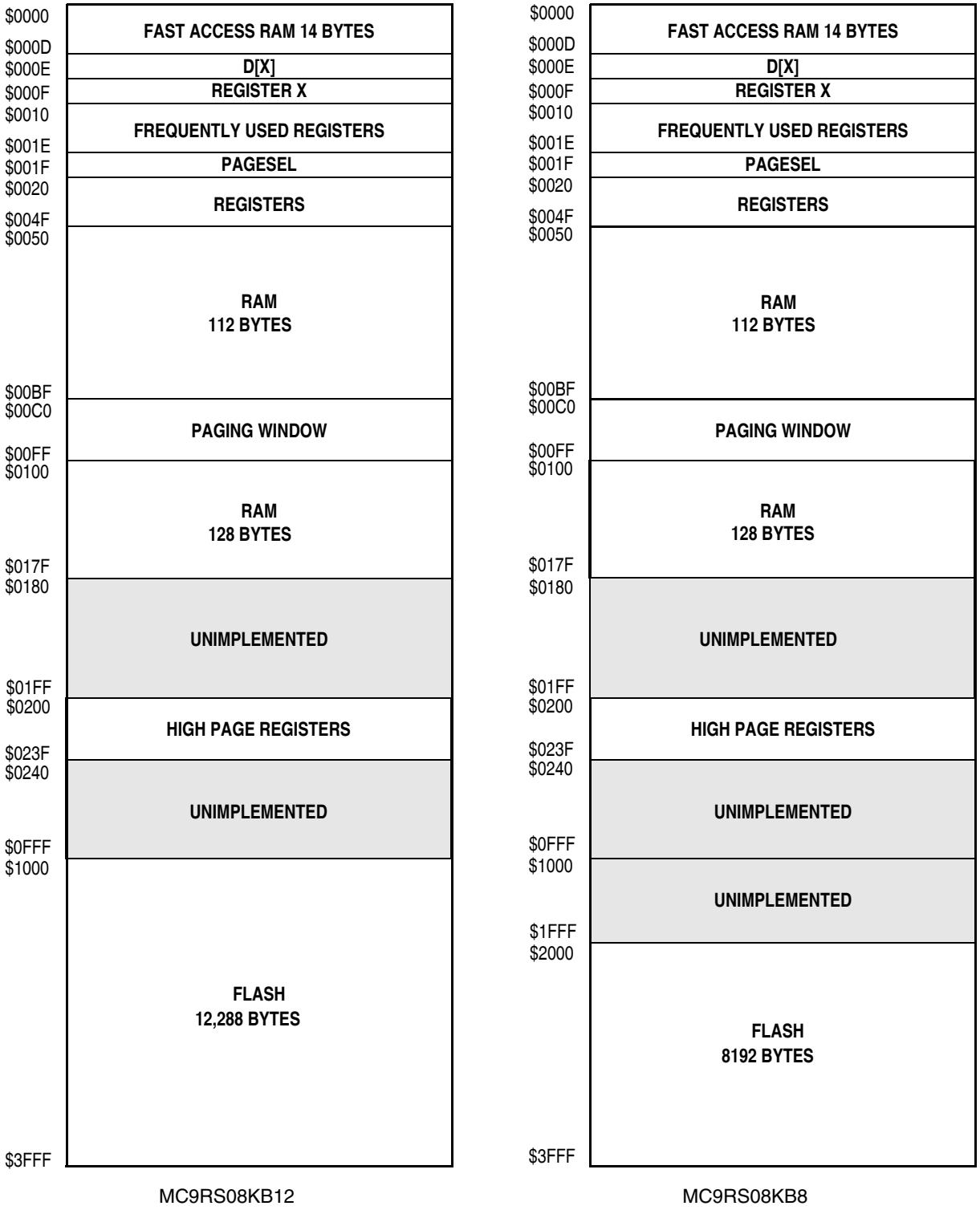


Figure 4-1. MC9RS08KB12 and MC9RS08KB8 Memory Maps

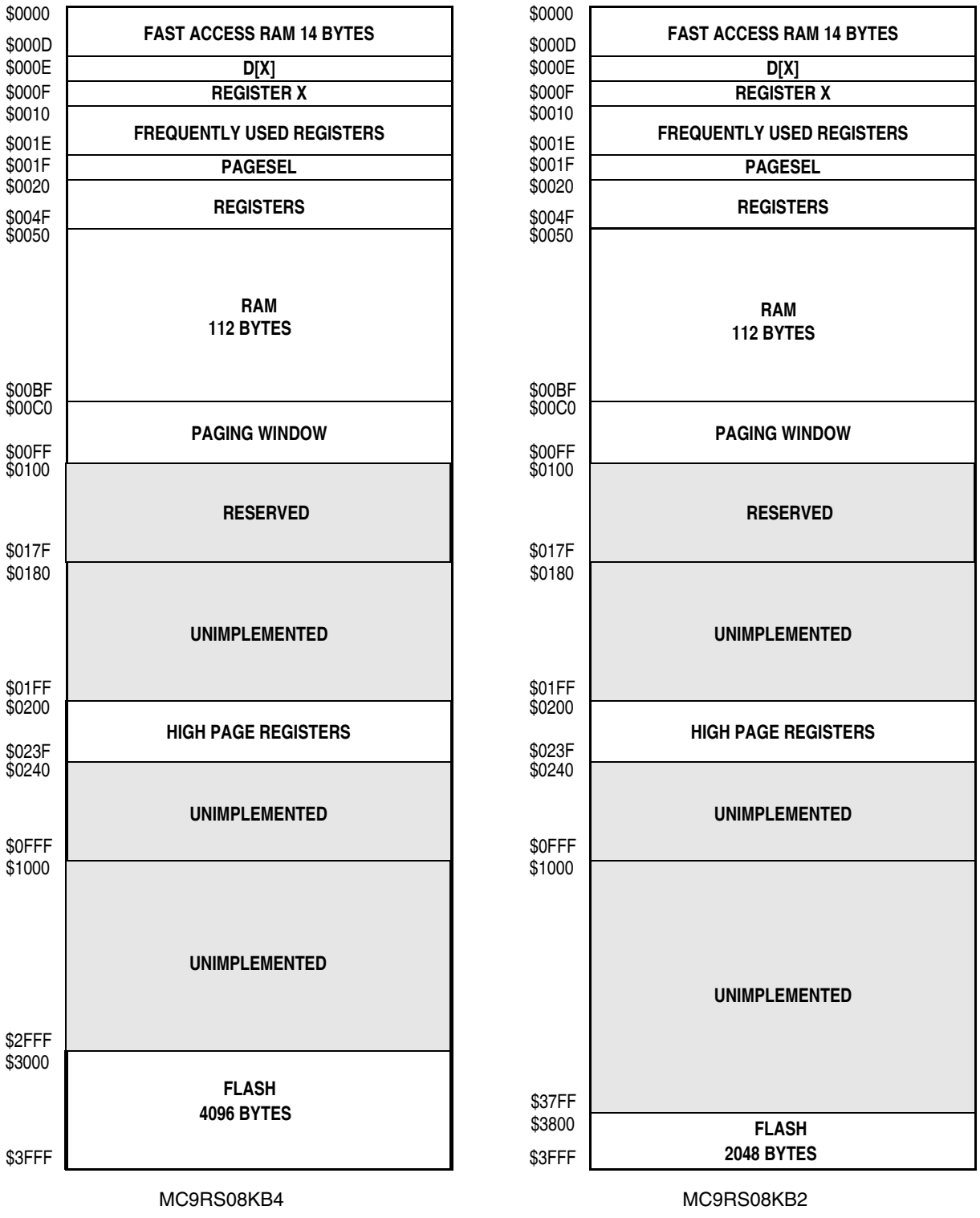


Figure 4-2. MC9RS08KB4 and MC9RS08KB2 Memory Maps

4.2 Unimplemented Memory

Attempting to access data or an instruction at an unimplemented memory address causes reset.

4.3 Indexed/Indirect Addressing

Register D[X] and register X combined performs 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 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 *latex* 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.

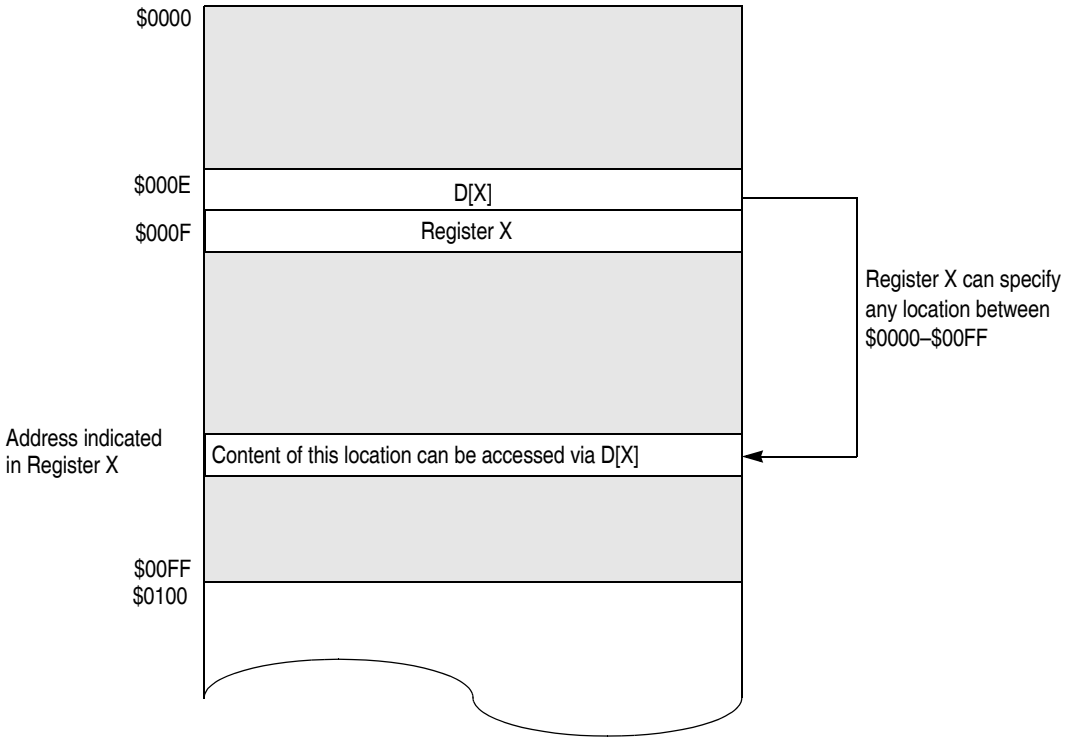


Figure 4-3. Indirect Addressing Registers

4.4 RAM and Register Addresses and Bit Assignments

Use short and direct addressing mode instructions to read and write to the fast access RAM area. For tiny addressing mode instructions, the operand is encoded 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.

Table 4-1. Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0000–$000D		Fast Access RAM							
$000E	D[X][1]	Bit 7	6	5	4	3	2	1	Bit 0
$000F	X	Bit 7	6	5	4	3	2	1	Bit 0
$0010	ADCSC1	COCO	AIEN	ADCO	ADCH				
$0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	0	0	R	R
$0012	ADCRH	0	0	0	0	0	0	ADR9	ADR8
$0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
$0014	ADCCVH	0	0	0	0	—	—	ADCV9	ADCV8
$0015	ADCCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
$0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
$0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
$0018	APCTL2	0	0	0	0	ADPC11	ADPC10	ADPC9	ADPC8
$0019	Unimplemented	—	—	—	—	—	—	—	—
$001A	ACMPSC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD	
$001B	Unimplemented	—	—	—	—	—	—	—	—
$001C	SIP1	LVD	KBI	ACMP	ADC	IIC	MTIM2	MTIM1	RTI
$001D	SIP2	IMASK	TPMCH0	TPMCH1	TPMOV	0	SCIE	SCIR	SCIT
$001E	Unimplemented	—	—	—	—	—	—	—	—
$001F	PAGESEL	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6
$0020	PTAD	0	0	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
$0021	PTADD	0	0	0	0	PTADD3	PTADD2	PTADD1	PTADD0
$0022	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
$0023	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
$0024	PTCD	0	0	0	0	PTCD3	PTCD2	PTCD1	PTCD0
$0025	PTCDD	0	0	0	0	PTCDD3	PTCDD2	PTCDD1	PTCDD0
$0026	Unimplemented	0	0	0	0	0	0	0	0
$0027	Unimplemented	0	0	0	0	0	0	0	0
$0028	MTIM1SC	TOF	TOIE	TRST	TSTP	0	0	0	0
$0029	MTIM1CLK	0	0	CLKS		PS			
$002A	MTIM1CNT	COUNT							
$002B	MTIM1MOD	MOD							
$002C	MTIM2SC	TOF	TOIE	TRST	TSTP	0	0	0	0
$002D	MTIM2CLK	0	0	CLKS		PS			
$002E	MTIM2CNT	COUNT							
$002F	MTIM2MOD	MOD							
$0030	SCIBDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
$0031	SCIBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
$0032	SCIC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
$0033	SCIC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Table 4-1. Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0034	SCIS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
$0035	SCIS2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
$0036	SCIC3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
$0037	SCID	R7	R6	R5	R4	R3	R2	R1	R0
		T7	T6	T5	T4	T3	T2	T1	T0
$0038	IICA	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
$0039	IICF	MULT		ICR					
$003A	IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
$003B	IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
$003C	IICD	DATA							
$003D	IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
$003E–$003F	Unimplemented	—	—	—	—	—	—	—	—
$0040	TPMSC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
$0041	TPMCNTH	Bit15	14	13	12	11	10	9	Bit8
$0042	TPMCNTL	Bit7	6	5	4	3	2	1	Bit0
$0043	TPMMODH	Bit15	14	13	12	11	10	9	Bit8
$0044	TPMMODL	Bit7	6	5	4	3	2	1	Bit0
$0045	TPMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
$0046	TPMC0VH	Bit15	14	13	12	11	10	9	Bit8
$0047	TPMC0VL	Bit7	6	5	4	3	2	1	Bit0
$0048	TPMC1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
$0049	TPMC1VH	Bit15	14	13	12	11	10	9	Bit8
$004A	TPMC1VL	Bit7	6	5	4	3	2	1	Bit0
$004B	Unimplemented	—	—	—	—	—	—	—	—
$004C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBMOD
$004D	KBIPE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
$004E	KBIES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
$004F	Unimplemented	—	—	—	—	—	—	—	—
$0050–$00BF		RAM							
$00C0–$00FF		Paging Window							
$0100–$017F		RAM							
$0180–$01FF	Unimplemented	—	—	—	—	—	—	—	—
$0200	IEA	1	0	1	1	1	1	0	0
$0201	IRAH	ZF	CF	A13	A12	A11	A10	A9	A8
$0202	IRAL	A7	A6	A5	A4	A3	A2	A1	A0
$0203–$0207	Unimplemented	—	—	—	—	—	—	—	—
$0208	FOPT	0	0	0	0	0	0	0	SECD
$0209	FLCR	0	0	0	0	HVEN	MASS	0	PGM

Table 4-1. Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$020A–$020B	Reserved	—	—	—	—	—	—	—	—
$020C–$020F	Unimplemented	—	—	—	—	—	—	—	—
$0210	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
$0211	SOPT1	COPE	COPT	STOPE	0	IICPS	SCIS	BKGDPE	RSTPE
$0212	SOPT2	0	0	0	0	MTIM1EC	ADCHTS	TPMCH1PS	TPMCH0PS
$0213	Reserved	—	—	—	—	—	—	—	—
$0214	Reserved	—	—	—	—	—	—	—	—
$0215	Unimplemented	—	—	—	—	—	—	—	—
$0216	SDIDH	REV3	REV2	REV1	REV0	ID[11:8]			
$0217	SDIDL	ID[7:0]							
$0218	SRTISC	RTIF	RTIACK	RTICLKS	RTIE	0	RTIS		
$0219	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
$021A	Reserved	—	—	—	—	—	—	—	—
$021B	Reserved	—	—	—	—	—	—	—	—
$021C–$021F	Unimplemented	—	—	—	—	—	—	—	—
$0220	PTAPE	0	0	PTAPE5	0	PTAPE3	PTAPE2	PTAPE1	PTAPE0
$0221	PTAPUD	0	0	PTAPUD5	0	PTAPUD3	PTAPUD2	PTAPUD1	PTAPUD0
$0222	PTASE	0	0	0	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
$0223	PTADS	0	0	0	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
$0224	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
$0225	PTBPUD	PTBPUD7	PTBPUD6	PTBPUD5	PTBPUD4	PTBPUD3	PTBPUD2	PTBPUD1	PTBPUD0
$0226	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
$0227	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
$0228	PTCPE	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
$0229	PTCPUD	0	0	0	0	PTCPUD3	PTCPUD2	PTCPUD1	PTCPUD0
$022A	PTCSE	0	0	0	0	PTCSE3	PTCSE2	PTCSE1	PTCSE0
$022B	PTCDS	0	0	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
$022C–$0237	Unimplemented	—	—	—	—	—	—	—	—
$0238	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
$0239	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
$023A	ICSTRM	TRIM							
$023B	ICSSC	0	0	0	0	CLKST		OSCINIT	FTRIM
$023C–$023F	Unimplemented	—	—	—	—	—	—	—	—
$3FF7–$3FF9[2]	Reserved	—	—	—	—	—	—	—	—

Table 4-1. Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$3FFA[3]	Reserved	ICS Trim Value							
$3FFB[3]	Reserved	Reserved							FTRIM
$3FFC	NVOPT	0	0	0	0	0	0	0	SECD
$3FFD–$3FFF[4]	Reserved	—	—	—	—	—	—	—	—

= Unimplemented or Reserved

1 Physical RAM in $000E can be accessed through D[X] register when the content of the index register X is $0E.

2 Once a pending interrupt is allowed to interrupt the CPU, the CPU executes code from internal memory with execution beginning at the address $3FF7. A jump instruction (opcode $BC) at $3FF7 with operand located at $3FF8-$3FF9 must be programmed into the user application for correct interrupt operation. The operand defines where the user interrupt service routing (ISR).

3 If using the MCU untrimmed, $3FFA and $3FFB may be used by applications.

4 During reset, the program counter is started from location $3FFD. A JMP instruction (opcode $BC) with operand at $3FFE-$3FFF must be programmed into the user application for correct reset operation. The operand defines where the user program starts.

The ICS factory-trimmed value will be stored in 0x3FFA and 0x3FFB (bit 0). The factory-trimmed bus frequency is 8 MHz.

4.5 RAM

The device includes three static RAM sections. 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 RAM can 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 can be accessed using direct-addressing mode instructions. The third section of RAM (not available for MC9RS08KB4 and MC9RS08KB2) starts from $0100 to $017F.

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

4.6 Flash

The flash memory is for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. You can 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

Flash memory features include:

- Up to 1000 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 $2X00, $2X40, $2X80, or $2XC0. 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 any data to any 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 PAGESEL register must be configured correctly to map the high-page-accessing window to the corresponding flash row.)
4. Wait for a time, t_{nvs}.
5. Set the HVEN bit.
6. Wait for a time, t_{pgs}.
7. Write data to the flash location to be programmed.
8. Wait for a time, t_{prog}.
9. Repeat steps seven and eight until all bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for a time, t_{nvh}.
12. Clear the HVEN bit.
13. After time, t_{rcv}, 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

Software code executed from flash locations cannot program or erase flash memory. To program or erase flash, commands must be executed from RAM or BDC commands. User code must not enter wait or stop during an erase or program sequence.

These operations must be performed in the order shown. Unrelated operations may occur between the steps.

4.6.3 Flash Mass Erase Operation

Mass erase the entire flash memory by the following operations:

1. Apply external V_{PP}.
2. Set the MASS bit in the flash control register.
3. Write any data to any flash location via the high-page-accessing window $00C0–$00FF. (Prior to the data writing operation, the PAGESEL register must be configured correctly to map the high-page-accessing window to any flash locations.)
4. Wait for a time, t_{nvs}.

5. Set the HVEN bit.
6. Wait for a time t_{me}.
7. Clear the MASS bit.
8. Wait for a time, t_{nvhl}.
9. Clear the HVEN bit.
10. After t_{rcv} time, the memory can be accessed in read mode again.
11. Remove external V_{PP}.

NOTE

Software code executed from flash locations cannot program or erase flash memory. To program or erase flash, commands must be executed from RAM or BDC commands. User code must not enter wait or stop during an erase or program sequence.

These operations must be performed in the order shown. Unrelated operations may occur between the steps.

4.6.4 Security

The MC9RS08KB12 series include circuitry to help prevent unauthorized access to flash memory contents. When security is engaged, flash is a secure resource. The RAM, direct-page registers, and background debug controller are unsecured resources. Attempts to access a secure memory location are blocked (reads return all 0s) if they are through the background debug interface, or when BKGDPE is set.

Security is engaged or disengaged based on the state of a nonvolatile register bit (SECD) in the FOPT register. During reset, the nonvolatile location NVOPT contents are copied from flash into the working FOPT register in high-page register space. Engage security by programming the NVOPT location. You can do this while the flash memory is programmed. The erased state (SECD = 1) makes the MCU unsecure. When SECD in NVOPT is programmed (SECD = 0), the next time the device is reset via POR, internal reset, or external reset, security is engaged. To disengage security, mass erase must be performed via BDM commands and followed by any reset.

The separate background debug controller can also be used for registers and RAM access. Via BDM commands, 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.”

Security can always be disengaged through the background debug interface when you:

1. mass erase flash via background BDM commands or RAM loaded program.
2. perform reset. The device boots up with security disengaged.

NOTE

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

4.7 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 at reset.

4.7.1 Flash Options Register (FOPT and NVOPT)

During reset, the nonvolatile location NVOPT contents 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. To change the value in this register, erase and reprogram the NVOPT location in flash memory then issue a new MCU reset.

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

= Unimplemented or Reserved

Figure 4-4. Flash Options Register (FOPT)

Table 4-2. FOPT Field Descriptions

Field	Description
0 SECD	**Security State Code** — This bit field determines the MCU security state. When the MCU is secured, the flash memory contents cannot be accessed by instructions from any unsecured source including the background debug interface; see Section 4.6.4, “Security.” 0 Security engaged. 1 Security disengaged.

4.7.2 Flash Control Register (FLCR)

	7	6	5	4	3	2	1	0
R	0	0	0	0	HVEN	MASS	0	PGM[1]
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 4-5. Flash Control Register (FLCR)

Table 4-3. FLCR Field Descriptions

Field	Description
3 HVEN	**High Voltage Enable** — This read/write bit enables high voltages to the flash array for program and erase operations. HVEN can be set only if 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	**Mass Erase Control Bit** — This read/write bit configures the memory for mass erase operation. 0 Mass-erase operation not selected. 1 Mass-erase operation selected.
0 PGM[1]	**Program Control Bit** — This read/write bit configures the memory for program operation. PGM is interlocked with the MASS bit, so both bits cannot be equal to 1 or set to 1 at the same time. 0 Program operation not selected. 1 Program operation selected.

[1] When flash security is engaged, writing to PGM bit has no effect. As a result, flash programming is not allowed.

4.8 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 registers ($0200 – $023F) can be accessed through the paging window ($00C0 – $00FF) via direct-addressing mode instructions.

	7	6	5	4	3	2	1	0
R W	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6
Reset	0	0	0	0	1	0	0	0

Figure 4-6. Page Select Register (PAGESEL)

Table 4-4. PAGESEL Field Descriptions

Field	Description
7:0 AD[13:6]	**Page Selector**— 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-page window. See Figure 4-7 and Table 4-5.

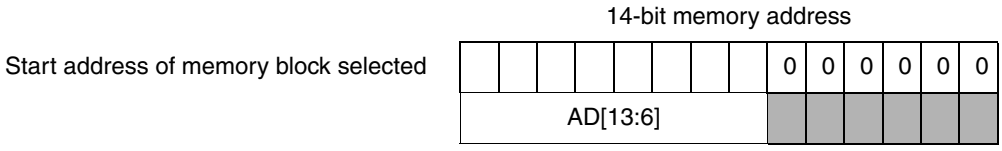


Figure 4-7. Memory Block Boundary Selector

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

Table 4-5. Paging Window for $00C0–$00FF

Page	Memory Address	Contents
$00	$0000–$003F	RAM, D[X], X, PAGESEL, and register
$01	$0040–$007F	register and RAM
$02	$0080–$00BF	RAM
$03	$00C0–$00FF	Paging window itself
$04–$05	$0100–$017F	RAM
$06–$07	$0180–$01FF	Unimplemented
$08	$0200–$023F	High-page pegister
$09–$3F	$0240–$0FFF	Unimplemented
$40–$FF	$1000–$3FFF	Flash

NOTE

Physical location $0000–$000E is RAM. Physical location $000F is register X. The D[X] register is mapped to address $000E only. The physical RAM in $000E can be accessed through the D[X] register when the X register is $0E or $CE with PAGESEL as $00.

When PAGESEL register is $00, the paging window is mapped to the first page ($00–$3F). Paged location $00C0–$00CE is mapped to physical location $0000–$000E, that is, RAM. Paged location $00CF is mapped to register X. Therefore, accessing address $CE returns the physical RAM content in $000E. Accessing address $000E returns D[X] register content.

Chapter 5
Resets, Interrupts, and General System Control

5.1 Introduction

This chapter discusses basic reset, interrupt mechanisms, and the various reset and interrupt sources in the MC9RS08KB12 series. Some interrupt sources from peripheral modules are discussed in detail in other chapters of this reference manual. This chapter gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and wakeup sources, including the computer operating properly (COP) watchdog and real-time interrupt (RTI), are not part of on-chip peripheral systems with their own chapters but are part of the system control logic.

5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- System reset status register (SRS) to indicate the source of the most recent reset
- A single interrupt vector
- A global interrupt mask bit (IMASK) in SIP2 register to enable/disable the interrupt vectoring mechanism
- System interrupt pending register (SIP1 and SIP2) to indicate the status of pending system interrupts
  - Analog comparator interrupt with enable
  - Keyboard interrupt with enable
  - Modulo timer interrupt with enable
  - Real-time interrupt with enable
  - ADC interrupt with enable
  - IIC interrupt with enable
  - TPM interrupts with enable
  - SCI interrupts with enable
  - Low voltage detect interrupt with enable
  - Real-time interrupts with enable

5.3 MCU Reset

Resetting the MCU provides a way to start processing from an known set of 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 starts. 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 MC9RS08KB12 series have seven reset sources:

- External pin reset (PIN) — Enabled using RSTPE in SOPT1
- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Background debug forced reset via BDC command BDC_RESET

Each source except the background debug forced reset has an associated bit in the system reset status register (SRS).

5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is used to force a system reset if the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must periodically reset the COP counter. 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 an known starting point.

After any reset, the COPE becomes set in SOPT1, which enables the COP watchdog (see Section 5.8.2, “System Options Register (SOPT1).”). If the COP watchdog is not used in an application, it is 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 SOPT1. Table 5-1 summarizes the COPT bit control functions. The COP watchdog operates from the 1 kHz clock source and defaults to the associated long timeout (2^8 cycles).

Table 5-1. COP Configuration Options

COPT	COP Overflow Count[1]
0	2^5 cycles (32 ms)
1	2^8 cycles (256 ms)

[1] Values in this column are based on $t_{RTI} \approx 1$ ms. See t_{RTI} in the “MC9RS08KB12 Series *Data Sheet*,” for the tolerance value.

Even if the application uses the reset default settings of COPE and COPT, write to the write-once SOPT1 registers during reset initialization to lock in the settings so they cannot be changed accidentally if the application program gets lost. The initial write to SOPT1 resets the COP counter.

In background debug mode, the COP counter does 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 original RS08 architecture does not include an interrupt controller with vector table lookup mechanism as used on the HC08 and HCS08 devices. To support interrupts it is the responsibility of the user application to poll the system interrupt pending registers (SIPx) to determine if an interrupt was pending. Based on the need for better interrupt support for the RS08 family, a single interrupt vector is added.

5.5.1 Interrupt Operation

The RS08 supports a single interrupt vector in much the same way as the RS08 reset vector is supported. The CPU executes code from internal memory with execution beginning at the address $3FF7. A jump instruction (opcode $BC) at $3FF7 with operand located at $3FF8–$3FF9 must be programmed into the user application for correct interrupt operation. The operand defines the location of the user interrupt service routine (ISR) once a pending interrupt is allowed to interrupt the CPU.

A global interrupt mask (IMASK) bit has been added to the SIP2 register to enable/disable the interrupt vectoring mechanism. When IMASK is cleared, a pending interrupt (any flag set in SIP1 or SIP2) will force an interrupt request to the CPU and the interrupt will be serviced immediately after completion of the current instruction. When IMASK is set, interrupts must be polled as in original RS08 architecture.

The program counter (PC) of the instruction that would have been executed if not for the interrupt request is saved in the interrupt return address (IRA) register. In this document, IRA refers to the concatenation of IRAH and IRAL (IRAH:IRAL). Once the ISR code has completed, the address saved in IRA will be used as the return address to the program executing prior to servicing the interrupt. A jump instruction is hardware encoded in the user memory map at the address immediately preceding the IRA register and is referred to as the interrupt exit address (IEA) register. A jump to IEA is used at the end of the ISR as the method for returning to the program executing prior to servicing the interrupt.

The condition code register (CCR) contents at the time of the interrupt are also saved in the IRAH most-significant two bits and will be restored upon a return from the ISR.

No other registers are saved automatically during interrupt vectoring. If the user determines that his ISR will corrupt the accumulator or index registers, then it is up to the user to save this contents in RAM and must be restored before a return from the ISR.

The IMASK bit is set when the CPU vectors begin to interrupt servicing.Within the ISR, the user can poll SIP1 and SIP2 and service pending interrupts based on application priority.

Once the ISR has been executed, the user must follow an ISR exit sequence:

- To re-enable interrupts on return from the ISR, the last two instructions of the ISR must be a clear of the IMASK bit followed by a jump to IEA. There is a 3-cycle delay between the write to clear

IMASK and the enabling of interrupts to allow the double jump (JMP IEA, JMP [IRA]) to normal program execution to occur before handling any pending interrupts.

- To continue to mask interrupts on return from the ISR, the user must execute the double jump (JMP IEA, JMP [IRA]) to normal program execution without write to IMASK.

5.5.2 Interrupt Operation in 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. The RS08 supports wakeup from WAIT differently depending on the state of IMASK in SIP2.

If IMASK = 1, interrupt vectoring is disabled. When an interrupt pending flag becomes set, the MCU exits wait mode and resumes processing at the next instruction.

If IMASK = 0, interrupt vectoring is enabled. When an interrupt pending flag becomes set, the MCU exits wait mode, the address of the instruction following the WAIT is stored in IRA, PC is loaded with 0x3FF7 and code execution begins.

5.5.3 Interrupt Operation Stop Mode

Stop mode is entered upon execution of a STOP instruction when the STOPE bit in the SOPT1 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.

Stop is exited by asserting $\overline{\text{RESET}}$ or any asynchronous interrupt that is enabled. 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, the sequence varies depending on the state of IMASK in SIP2.

If IMASK = 1, interrupt vectoring is disabled. When an asynchronous interrupt request occurs. the MCU exits stop mode and resumes processing at the next instruction.

If IMASK = 0, interrupt vectoring is enabled. When an interrupt pending flag becomes set, the MCU exits stop mode, the address of the instruction following the STOP is stored in IRA, PC is loaded with 0x3FF7 and code execution begins.

5.6 Low-Voltage Detect (LVD) System

The MC9RS08KB12 series include a system to protect memory contents and control MCU system states against low voltage conditions during supply voltage variations. The system is composed 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 is 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 causes a reset condition. As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply rises above the V_{LVD} level. The POR bit and LVD bit in SRS are set after a POR.

5.6.2 LVD Reset Operation

The LVD can be configured to generate a reset upon detecting a low voltage condition by setting LVDRE to 1. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above the level V_{LVD}. The LVD bit in the SRS register is set after 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 is set and an LVD interrupt request occurs.

5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts. The RTI is driven from the 1 kHz internal clock oscillator or the external clock souce. The RTICLKS bit in SRTISC is used to select the RTI clock source. The 1 kHz internal or external clock source 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 0s, and no interrupts are generated. See Section 5.8.5, “System Real-Time Interrupt Status and Control Register (SRTISC),” for more information.

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 operation modes. Although brief descriptions of these bits are provided here, the related functions are further discussed in Chapter 3, “Modes of Operation.”

5.8.1 System Reset Status Register (SRS)

This high-page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by the BDC_RESET command, all of the status bits in SRS are cleared. Writing any value to this register address clears the COP watchdog timer without affecting this register’s contents. 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	Note 1	Note 1	Note 1	Note 1	0	0	0

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

Figure 5-1. System Reset Status (SRS)

Table 5-2. SRS Field Descriptions

Field	Description
7 POR	**Power-On Reset** — Reset caused by 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	**External Reset Pin** — Reset caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 External reset pin caused reset.
5 COP	**Computer Operating Properly (COP) Watchdog** — Reset caused by the COP watchdog timer timing out. COPE = 0 in the SOPT1 register can block the reset source. 0 Reset not caused by COP timeout. 1 COP timeout caused reset.
4 ILOP	**Illegal Opcode** — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if STOPE = 0 in the SOPT register disables stop mode. The BGND instruction is considered illegal if ENBDM = 0 in the BDCSC register disables background debug mode. 0 Reset not caused by an illegal opcode. 1 Illegal opcode caused reset.
3 ILAD	**Illegal Address** — Reset 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 Illegal address caused reset.
1 LVD	**Low Voltage Detect** — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset occurs. POR sets this bit. 0 Reset not caused by LVD trip or POR. 1 Either LVD trip or POR caused reset.

5.8.2 System Options Register (SOPT1)

This high-page register is a write-once register so only the first write after reset is honored. It can be read at any time. Any subsequent attempt to write to SOPT1 (intentionally or unintentionally) is ignored to

avoid accidental changes to these sensitive settings. SOPT1 must be written during the user's reset initialization program to set the desired controls even if the desired settings and reset settings are the same.

	7	6	5	4	3	2	1	0
R	COPE	COPT	STOPE	0	IICPS	SCIS	BKGDPE	RSTPE
W								
Reset:	1	1	0	0	0	0	1 (Note 1)	u
POR:	1	1	0	0	0	0	1 (Note1)	0

= Unimplemented or Reserved u = Unaffected

1 When the device is in power-on reset, BKGEPE is reset to 1. 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 (SOPT1)

Table 5-3. SOPT1 Register Field Descriptions

Field	Description
7 COPE	**COP Watchdog Enable** — This write-once bit selects whether the COP watchdog is enabled. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	**COP Watchdog Timeout** — This write-once bit selects the timeout period of the COP. 0 Short timeout period selected. 1 Long timeout period selected.
5 STOPE	**Stop Mode Enable** — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
3 IICPS	**IIC Pin Select —** This bit selects the location of the SDA and SCL pins of the IIC module. 0 SDA on PTA2, SCL on PTA3. 1 SDA on PTB6, SCL on PTB7.
2 SCIS	**SCI Pin Select —** This bit selects the location of the RxD and TxD pins of the SCI module. 0 RxD on PTB0, TxD on PTB1. 1 RxD on PTA2, TxD on PTA3.
1 BKGDPE[1,2]	**Background Debug Mode Pin Enable** — When set, this write-once bit enables the PTA4/ACMPO/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset. 0 PTA4/ACMPO/BKGD/MS pin functions as PTA4 or ACMPO. 1 PTA4/ACMPO/BKGD/MS pin functions as BKGD/MS.
0 RSTPE	**$\overline{\text{RESET}}$ Pin Enable** — When set, this write-once bit enables the PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} pin to function as $\overline{\text{RESET}}$. 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 $\overline{\text{RESET}}$. 0 PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} pin functions as PTA5/TCLK/V_{PP} 1 PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} pin functions as $\overline{\text{RESET}}$/V_{PP}

1 When the device is power on reset, BKGEPE is reset to 1. 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 reset to 1 so that BDM communication is allowed.

2 BKGDPE can write only once from value 1 to 0. Writing from value 0 to 1 by user software is not allowed. BKGDPE can be changed back to 1 only by a POR or reset with proper condition as stated in Note 1.

5.8.3 System Options Register 2 (SOPT2)

This high-page register contains bits to configure MCU specific features on MC9RS08KB12 series devices.

	7	6	5	4	3	2	1	0
R	0	0	0	0	MTIM1EC	ADCHTS	TPMCH1PS	TPMCH0PS
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-3. System Options Register 2 (SOPT2)

Table 5-4. SOPT2 Register Field Descriptions

Field	Description
3 MTIM1EC	**MTIM1 External Clock Source Select —** This bit selects the external clock source of MTIM1. This bit is effective to select the external clock source of MTIM1 when MTIM1 selects external clock as its clock source, CLKS in the MTIMCLK resister is set to 10 or 11. 0 The external clock source of MTIM1 is TCLK. 1 The external clock source of MTIM1 is the overflow of MTIM2.
2 ADCHTS	**ADC Hardware Trigger Select —** This bit selects ADC hardware trigger source. 0 ADC hardware trigger is connected with RTI overflow. 1 ADC hardware trigger is connected with MTIM1 overflow.
1 TPMCH1PS	**TPMCH1 Pin Select—** This bit selects the location of the TPMCH1 pin of the TPM module. 0 TPMCH1 on PTA1. 1 TPMCH1 on PTB5.
0 TPMCH0PS	**TPMCH0 Pin Select—** This bit selects the location of the TPMCH0 pin of the TPM module. 0 TPMCH0 on PTA0. 1 TPMCH0 on PTB4.

5.8.4 System Device Identification Register (SDIDH, SDIDL)

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

	7	6	5	4	3	2	1	0
R	REV3	REV2	REV1	REV0	ID11	ID10	ID9	ID8
W								
Reset:	0 (Note 1)	0 (Note 1)	0 (Note 1)	0 (Note 1)	1	0	0	0

= Unimplemented or Reserved

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

Figure 5-4. System Device Identification Register — High (SDIDH)

Table 5-5. SDIDH Register Field Descriptions

Field	Description
7:4 REV[3:0]	**Revision Number** — The high-order 4 bits of address SDIDH are hard coded to reflect the current mask set revision number (0–F).
3:0 ID[11:8]	**Part Identification Number** — Each derivative in the RS08 Family has a unique identification number. The MC9RS08KB12 is hard coded to the value $0806. See also ID bits in Figure 5-5.

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

= Unimplemented or Reserved

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

Table 5-6. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	**Part Identification Number** — Each derivative in the RS08 family has a unique identification number. The MC9RS08KB12 is hard coded to the value $0806. See also ID bits in Figure 5-4.

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

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

	7	6	5	4	3	2	1	0
R	RTIF	0	RTICLKS	RTIE	0	RTIS		
W		RTIACK						
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-6. System RTI Status and Control Register (SRTISC)

Table 5-7. SRTISC Register Field Descriptions

Field	Description
7 RTIF	**Real-Time Interrupt Flag** — Read-only status bit indicates the periodic wakeup timer has timed out. 0 Periodic wakeup timer not timed out. 1 Periodic wakeup timer timed out.
6 RTIACK	**Real-Time Interrupt Acknowledge** — Write-only bit acknowledges real-time interrupt request (write 1 to clear RTIF). Writing 0 has no meaning or effect. Reads always return 0.
5 RTICLKS	**Real-Time Interrupt Clock Select** — 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	**Real-Time Interrupt Enable** — Read-write bit enables real-time interrupts. 0 Real-time interrupts disabled. 1 Real-time interrupts enabled.
2:0 RTIS	**Real-Time Interrupt Delay Selects** — These read/write bits select the period for the RTI. See Table 5-8.

Table 5-8. Real-Time Interrupt Period

RTIS	Using the 1 kHz Oscillator Source[1]	Using the External Clock Input
000	Disable and clear the RTI Counter	Disable and clear the RTI Counter
001	8 ms	$1/f_{extclk} \times 256$
010	32 ms	$1/f_{extclk} \times 1024$
011	64 ms	$1/f_{extclk} \times 2048$
100	128 ms	$1/f_{extclk} \times 4096$
101	256 ms	$1/f_{extclk} \times 8192$
110	512 ms	$1/f_{extclk} \times 16384$
111	1.024 s	$1/f_{extclk} \times 32768$

[1] Values are shown in this column based on f_{RTI}= 1 kHz. Consult electricals specification from MC9RS08KB12 Series *Data Sheet*.

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

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

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

= Unimplemented or Reserved

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

Figure 5-7. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-9. SPMSC1 Register Field Descriptions

Field	Description
7 LVDF	**Low-Voltage Detect Flag** — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event.
6 LVDACK	**Low-Voltage Detect Acknowledge** — Write-only bit, is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return 0.
5 LVDIE	**Low-Voltage Detect Interrupt Enable** — Enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF = 1.
4 LVDRE	**Low-Voltage Detect Reset Enable** — Write-once bit enables low-voltage detect events to generate a hardware reset (provided LVDE = 1). 0 LVDF does not generate hardware resets. 1 Force an MCU reset when LVDF = 1.
3 LVDSE	**Low-Voltage Detect Stop Enable** — Provided LVDE = 1, this read/write bit determines whether the low-voltage-detect function operates when MCU is in stop mode. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	**Low-Voltage Detect Enable** — Write-once bit enables low-voltage detect logic and the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	**Bandgap Buffer Enable** — Enables an internal buffer for the bandgap-voltage reference for use by the ACMP and ADC modules on one of its internal channels. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

5.8.7 System Interrupt Pending Register 1 (SIP1)

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

	7	6	5	4	3	2	1	0
R	LVD	KBI	ACMP	ADC	IIC	MTIM2	MTIM1	RTI
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-8. System Interrupt Pending Register 1 (SIP1)

Table 5-10. SIP1 Register Field Descriptions

Field	Description
7 LVD	**Low Voltage Detect Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt from LVD.Clearing the LVDF flag of SPMSC1 register clears this bit. Reset also clears this bit. 0 No pending LVD interrupt. 1 Pending LVD interrupt.
6 KBI	**Keyboard Interrupt Pending** — This read-only bit indicates whether a pending interrupt from the KBI module. Clearing the KBF flag of the KBISC register clears this bit. Reset also clears this bit. 0 No pending KBI interrupt. 1 Pending KBI interrupt.
5 ACMP	**Analog Comparator Interrupt Pending** — This read-only bit indicates whether a pending interrupt from the ACMP module. Clearing the ACF flag of the ACMPSC register clears this bit. Reset also clears this bit. 0 No pending ACMP interrupt. 1 Pending a ACMP interrupt.
4 ADC	**ADC Interrupt Pending —** This read-only bit indicates whether a pending interrupt from the ADC module. Clearing the COCO flag of the ADCSC1 register clears this bit. Reset also clears this bit. 0 No pending ADC interrupt. 1 Pending ADC interrupt.
3 IIC	**IIC Interrupt Pending** — This read-only bit indicates whether a pending interrupt from the IIC module. Clearing the IICIF flag of the IICS register clears this bit. Reset also clears this bit. 0 No pending IIC interrupt. 1 Pending IIC interrupt.
2 MTIM2	**Modulo Timer 2 Interrupt Pending** — This read-only bit indicates whether a pending interrupt from the MTIM2 module. Clearing the TOF flag of the MTIM2SC register clears this bit. Reset also clears this bit. 0 No pending MTIM2 interrupt. 1 Pending MTIM2 interrupt.
1 MTIM1	**Modulo Timer 1 Interrupt Pending** — This read-only bit indicates whether a pending interrupt from the MTIM1 module. Clearing the TOF flag of the MTIM1SC register clears this bit. Reset also clears this bit. 0 No pending MTIM1 interrupt. 1 Pending MTIM1 interrupt.
0 RTI	**Real-Time Interrupt Pending** — This read-only bit indicates whether a pending interrupt from the RTI. Clearing the RTIF flag of the SRTISC register clears this bit. Reset also clears this bit. 0 No pending RTI interrupt. 1 Pending RTI interrupt.

5.8.8 System Interrupt Pending Register 2 (SIP2)

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

	7	6	5	4	3	2	1	0
R	IMASK	TPMCH0	TPMCH1	TPMOV	0	SCIE	SCIR	SCIT
W								
Reset:	1	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-9. System Interrupt Pending Register 2 (SIP2)

Table 5-11. SIP2 Register Field Descriptions

Field	Description
7 IMASK	**Interrupt Mask —** This read/write bit controls whether an interrupt request will be generated to the CPU. When cleared, any peripheral interrupt flag in SIP1 or SIP2 will cause an interrupt request to be generated to the CPU. This bit is set on any reset, and also upon a CPU interrupt vector. 0 Interrupt request enabled. 1 Interrupt request disabled.
6 TPMCH0	**TPM Channel 0 Interrupt Pending —** This read-only bit indicates whether a pending interrupts from the channel 0 of TPM module. Clearing the CH0F flag of the TPMC0SC register clears this bit. Reset also clears this bit. 0 No pending TPM Channel 0 interrupt. 1 Pending TPM Channel 0 interrupt.
5 TPMCH1	**TPM Channel 1 Interrupt Pending —** This read-only bit indicates whether a pending interrupts from the channel 1 of TPM module. Clearing the CH1F flag of the TPMC1SC register clears this bit. Reset also clears this bit. 0 No pending TPM Channel 1 interrupt. 1 Pending TPM Channel 1 interrupt.
4 TPMOV	**TPM Overflow Interrupt Pending —** This read-only bit indicates whether a pending interrupts of timer overflow from the TPM module. Clearing the TOF flag of the TPMSC register clears this bit. Reset also clears this bit. 0 No pending TPM timer overflow interrupt. 1 Pending TPM timer overflow interrupt.
2 SCIE	**SCI Error Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt of error from SCI module. Clearing the OR, NF, FE and PF flags of in the SCIS1 register clears this bit. Reset also clears this bit. 0 No pending SCI error interrupt. 1 Pending SCI error interrupt.
1 SCIR	**SCI Receiver Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt of receiving from SCI module. Clearing the RDRF and IDLE flags of the SCIS1 register, the LBKDIF and RXEDGIF flags of the SCIS2 register clears this bit. 0 No pending SCI receiver interrupt. 1 Pending SCI receiver interrupt.
0 SCIT	**SCI Transmitter Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt of transmitting from SCI module. Clearing the TDRE and TC flags of the SCIS1 register clears this bit. 0 No pending SCI transmitter interrupt. 1 Pending SCI transmitter interrupt.

5.8.9 Interrupt Exit Address Register (IEA)

This register contains a hardware encoded JMP opcode used to return from a user interrupt service routine to the address of the program executing prior to servicing the interrupt. This register will always precede the interrupt return address (IRA) in the memory map. The jump operand in the IEA register will use the address saved in the IRA as its operand. When exiting the ISR, there is a double jump that will occur to get back to the program executing prior to servicing the interrupt.

	7	6	5	4	3	2	1	0
R	Hardware Encoded JMP opcode ($BC)							
W								
Reset:	1	0	1	1	1	1	0	0

= Unimplemented or Reserved

Figure 5-10. Interrupt Exit Address Register (IEA)

5.8.10 Interrupt Return Address Registers (IRA = IRAH:IRAL)

In this document, IRA refers to the concatenation of IRAH and IRAL. IRAH contains the high byte of the address where the CPU returns after servicing an interrupt as well as the CCR flags to be restored upon interrupt return. IRAL contains the low byte of the address where the CPU returns after servicing an interrupt.

	7	6	5	4	3	2	1	0
R	ZF	CF	A13	A12	A11	A10	A9	A8
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-11. Interrupt Return Address High Register (IRAH)

Table 5-12. IRAH Register Field Descriptions

Field	Description
7 ZF	**ZF** -— Contains the CCR Z flag value saved before servicing an interrupt to be restored upon interrupt return.
6 CF	**CF** -— Contains the CCR C flag value saved before servicing an interrupt to be restored upon interrupt return.
5-0 A13-A8	**A13 - A8** — IRAH contains the high byte of the address where the CPU returns after servicing an interrupt.

	7	6	5	4	3	2	1	0
R	A7	A6	A5	A4	A3	A2	A1	A0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-12. Interrupt Program Counter Low Register (IRAL)

Table 5-13. IRAL Register Field Descriptions

Field	Description
7:0 A7-A0	**A7–A0** — IRAL contains the low byte of the address where the CPU returns after servicing an interrupt.

Chapter 6
Parallel Input/Output Control

This chapter explains software controls related to parallel input/output (I/O) and pin control. See Chapter 2, "Pins and Connections," for more information about pin assignments and external hardware considerations for these pins.

All these I/O pins are shared with on-chip peripheral functions (see Table 1). The peripheral modules have priority over the I/Os. 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 the I/O controls the pins. All the I/Os are configured as inputs (PTxDDn = 0) with pullup/pulldown devices disabled (PTxPEn = 0), except for output-only pin PTA4, which defaults to the BKGD/MS function. All pins' default-low-drive strengths are selected (PTxDSn = 0) after reset.

Reading and writing 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 block diagram in Figure 6-1 illustrates the parallel I/O port function for an individual pin.

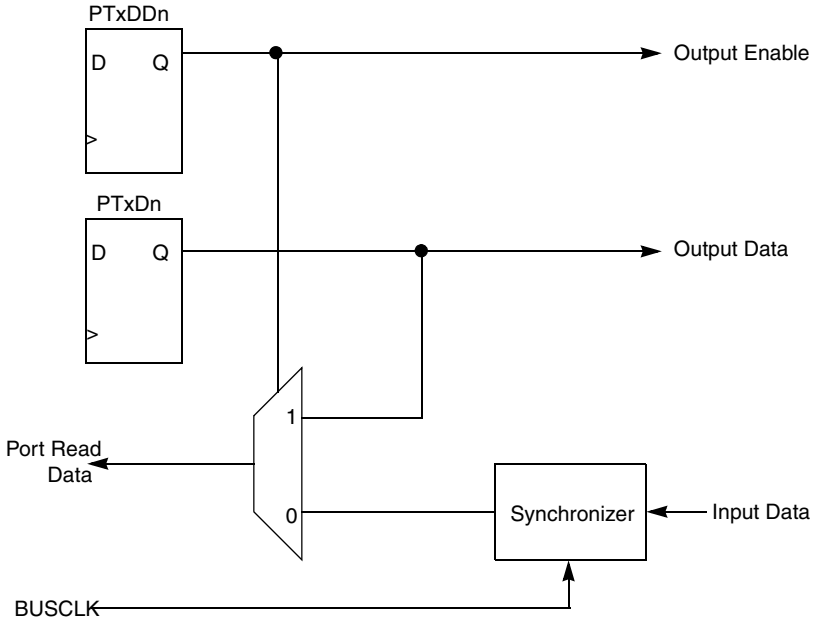


Figure 6-1. Parallel I/O Block Diagram

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 shared function controls the output buffer. However, the data direction register bit continues controlling the source for reads of the port data register.

When a shared analog function is enabled for a pin, 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 an alternative digital function and an analog function, the analog function has priority such that if the digital and analog functions are enabled, the analog function controls the pin.

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

A set of registers associated with the parallel I/O ports is located in the high-page register space that operate independently of the parallel I/O registers. These registers are used to control pullup/pulldown, slew rate and drive strength for the pins. See Section 6.3, “Pin Control Registers.”

6.1 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 revert to the state they were in prior to entering stop mode.

6.2 Parallel I/O Registers

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

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

6.2.1 Port A Registers

Port A parallel I/O function is controlled by the data and data direction registers described in this section.

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

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

Table 6-1. PTAD Register Field Descriptions

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

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

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

Table 6-2. PTADD Register Field Descriptions

Field	Description
3:0 PTADD[3:0]	**Data Direction for Port A Bits** — These read/write bits control the port A pins direction 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.

6.2.2 Port B Registers

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

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

Figure 6-4. Port B Data Register (PTBD)

Table 6-3. PTBD Register Field Descriptions

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

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

Figure 6-5. Port B Data Direction Register (PTBDD)

Table 6-4. PTBDD Register Field Descriptions

Field	Description
7:0 PTBDD[7:0]	**Data Direction for Port B Bits** — These read/write bits control the port B pins direction and what is read for PTBD reads. 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.

6.2.3 Port C Registers

Port C parallel I/O function is controlled by the data and data direction registers described in this section.

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCD3	PTCD2	PTCD1	PTCD0
W								
Reset:	0	0	0	0	0	0	0	0

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

Table 6-5. PTCD Register Field Descriptions

Field	Description
3:0 PTCD[3:0]	**Port C Data Register Bits** — For port C pins that are inputs, reads return the logic level on the pin. For port C pins 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 configured as outputs, the logic level is driven out on the corresponding MCU pin. Reset forces PTCD to all 0s, but these 0s are not driven out on the corresponding pins because reset also configures all port pins as high-impedance inputs with pullup/pulldowns disabled.

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCDD3	PTCDD2	PTCDD1	PTCDD0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-7. Port C Data Direction Register (PTCDD)

Table 6-6. PTCDD Register Field Descriptions

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

6.3 Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports that are used for pin control functions.

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

NOTE

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

6.3.1 Port A Pin Control Registers

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

6.3.1.1 Internal Pulling Device Enable

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

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

Figure 6-8. Internal Pulling Device Enable for Port A Register (PTAPE)

Table 6-7. PTAPE Register Field Descriptions

Field	Description
5,3:0 PTAPE[5,3:0]	**Internal Pulling Device Enable for Port A Bits** — Each of these control bits determines whether the internal pulling device is enabled for the associated PTA pin. For port A pins configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pulling device disabled for port A bit n. 1 Internal pulling device enabled for port A bit n.

6.3.1.2 Pullup/Pulldown Control

Pullup/pulldown control is used to select the pullup or pulldown device enabled by the corresponding PTAPE bit.

	7	6	5	4	3	2	1	0
R	0	0	PTAPUD5	0	PTAPUD3	PTAPUD2	PTAPUD1	PTAPUD0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-9. Pullup/Pulldown Device Control for Port A (PTAPUD)

Table 6-8. PTAPUD Register Field Descriptions

Field	Description
5,3:0 PTAPUD[5,3:0]	**Pullup/Pulldown Device Control for Port A Bits** — 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 bit n. 1 Internal pulldown device is selected for port A bit n.

6.3.1.3 Output Slew Rate Control Enable

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

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

Figure 6-10. Slew Rate Enable for Port A Register (PTASE)

Table 6-9. PTASE Register Field Descriptions

Field	Description
4:0 PTASE[4:0]	**Output Slew Rate Enable for Port A Bits** — Each of these control bits determines whether the output slew rate control is enabled for the associated PTA pin. For port A pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.

6.3.1.4 Port A Drive Strength Selection Register (PTADS)

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

Figure 6-11. Output Drive Strength Selection for Port A (PTADS)

Table 6-10. PTADS Register Field Descriptions

Field	Description
4:0 PTADS[4:0]	**Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. 0 Low output drive enabled for port A bit n. 1 High output drive enabled for port A bit n.

6.3.2 Port B Pin Control Registers

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

6.3.2.1 Internal Pulling Device Enable

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

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

Figure 6-12. Internal Pulling Device Enable for Port B Register (PTBPE)

Table 6-11. PTBPE Register Field Descriptions

Field	Description
7:0 PTBPE[7:0]	**Internal Pulling Device Enable for Port A Bits** — Each of these control bits determines whether the internal pulling device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pulling device disabled for port B bit n. 1 Internal pulling device enabled for port B bit n.

6.3.2.2 Pullup/Pulldown Control

Pullup/pulldown control is used to select the pullup or pulldown device enabled by the corresponding PTBPE bit.

	7	6	5	4	3	2	1	0
R W	PTBPUD7	PTBPUD6	PTBPUD5	PTBPUD4	PTBPUD3	PTBPUD2	PTBPUD1	PTBPUD0
Reset:	0	0	0	0	0	0	0	0

Table 6-12. Pullup/Pulldown Device Control for Port B (PTBPUD)

Table 6-13. PTBPUD Register Field Descriptions

Field	Description
7:0 PTBPUD[7:0]	**Pullup/Pulldown Device Control for Port B Bits** — 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. 0 Internal pullup device is selected for port B bit n. 1 Internal pulldown device is selected for port B bit n.

6.3.2.3 Output Slew Rate Control Enable

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

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

Figure 6-13. Slew Rate Enable for Port B Register (PTBSE)

Table 6-14. PTBSE Register Field Descriptions

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

6.3.2.4 Port B Drive Strength Selection Register (PTBDS)

	7	6	5	4	3	2	1	0
R W	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-14. Output Drive Strength Selection for Port B (PTBDS)

Table 6-15. PTBDS Register Field Descriptions

Field	Description
7:0 PTBDS[7:0]	**Output Drive Strength Selection for Port B Bits** — Each of these control bits selects between low and high output drive for the associated PTB pin. 0 Low output drive enabled for port B bit n. 1 High output drive enabled for port B bit n.

6.3.3 Port C Pin Control Registers

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

6.3.3.1 Internal Pulling Device Enable

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-15. Internal Pulling Device Enable for Port C Register (PTCPE)

Table 6-16. PTCPE Register Field Descriptions

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

6.3.3.2 Pullup/Pulldown Control

Pullup/pulldown control is used to select the pullup or pulldown device enabled by the corresponding PTCPE bit.

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCPUD3	PTCPUD2	PTCPUD1	PTCPUD0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-16. Pullup/Pulldown Device Control for Port C (PTCPUD)

Table 6-17. PTCPUD Register Field Descriptions

Field	Description
3:0 PTCPUD[3:0]	**Pullup/Pulldown Device Control for Port C Bits** — 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 bit n. 1 Internal pulldown device is selected for port C bit n.

6.3.3.3 Output Slew Rate Control Enable

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCSE3	PTCSE2	PTCSE1	PTCSE0
W								
Reset:	0	0	0	0	1	1	1	1

Figure 6-17. Slew Rate Enable for Port C Register (PTCSE)

Table 6-18. PTCSE Register Field Descriptions

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

6.3.3.4 Port C Drive Strength Selection Register (PTCDS)

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-18. Output Drive Strength Selection for Port C (PTCDS)

Table 6-19. PTCDS Register Field Descriptions

Field	Description
3:0 PTCDS[3:0]	**Output Drive Strength Selection for Port C Bits** — Each of these control bits is selected between low and high output drive for the associated PTC pin. 0 Low output drive enabled for port C bit n. 1 High output drive enabled for port C bit n.

Chapter 7
Keyboard Interrupt (RS08KBIV1)

7.1 Introduction

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

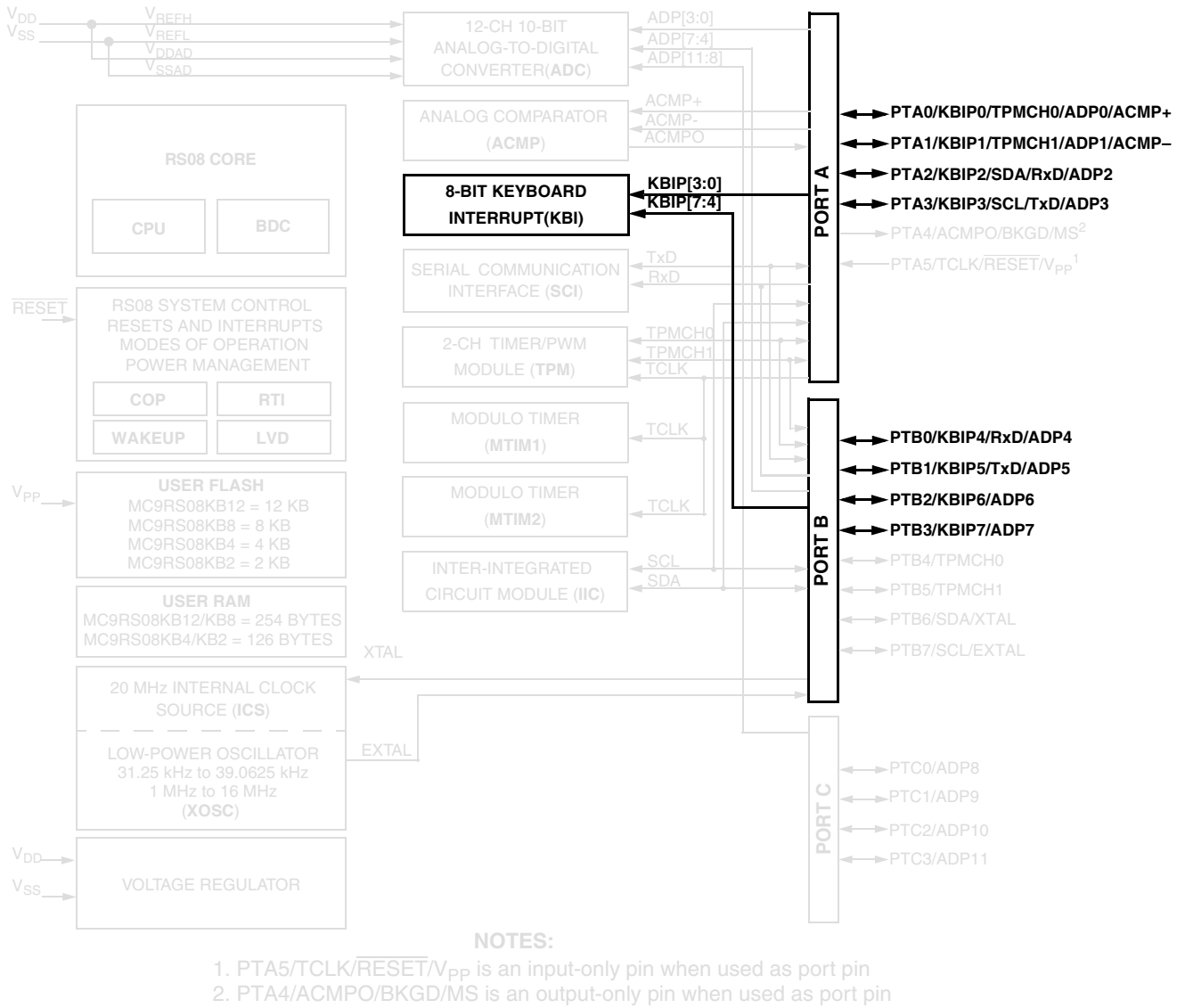


Figure 7-1. MC9RS08KB12 Series Block Diagram Highlighting KBI Block and Pins

7.1.1 Features

The KBI features include:

- Each keyboard interrupt pins has an individual pin enable bit
- Each keyboard interrupt pins 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 (KBPEn = 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 (KBPEn = 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 continues to operate normally.

7.1.3 Block Diagram

Figure 7-2 shows the block diagram for the keyboard interrupt module.

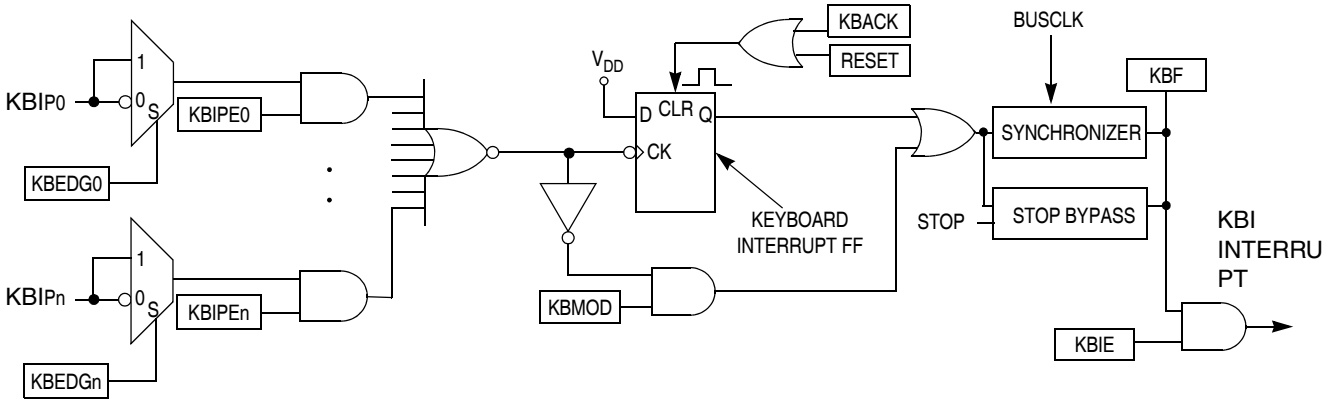


Figure 7-2. Keyboard Interrupt (KBI) Block Diagram

7.2 External Signal Description

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

Table 7-1 shows KBI signal properties.

Table 7-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

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 Chapter 4, “Memory,” for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

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								
PIN NAME	I	KBIP7	KBIP6	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0

7.3.1 KBI Status and Control Register (KBISC)

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBMOD
W						KBACK		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

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

Table 7-3. KBISC Register Field Descriptions

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

7.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin-enable-control bits.

	7	6	5	4	3	2	1	0
R W	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
Reset:	0	0	0	0	0	0	0	0

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

Table 7-4. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPEn	**Keyboard Pin Enables** — 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.

	7	6	5	4	3	2	1	0
R W	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
Reset:	0	0	0	0	0	0	0	0

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

Table 7-5. KBIES Register Field Descriptions

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

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 falling or rising; the level can be 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).

7.4.1 Edge Only Sensitivity

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

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

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 remains set if any enabled KBI pin is asserted while attempting to clear by writing a 1 to KBACK.

7.4.3 KBI Pullup/Pulldown Resistors

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

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

Chapter 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.

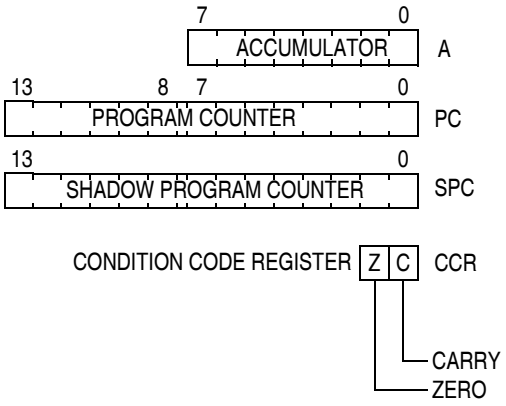


Figure 8-1. CPU Registers

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.

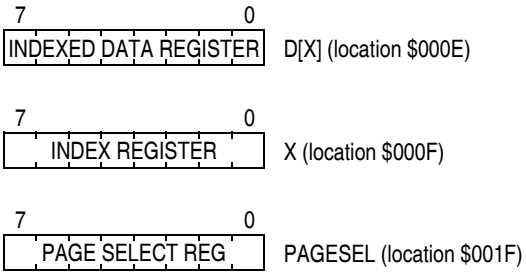


Figure 8-2. Memory Mapped Registers

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.

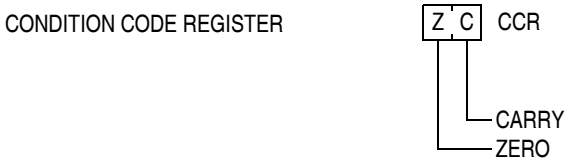


Figure 8-3. Condition Code Register (CCR)

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:   deca              ;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 will 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 will 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
rel	=	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
↕	=	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

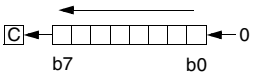
Everything in the source forms columns, *except expressions in italic characters,* is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic is always a literal expression. All commas, pound signs (#), parentheses, and plus signs (+) are 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 6)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
ADC #*opr8i* ADC *opr8a* ADC ,X [1] ADC X	Add with Carry	A ← (A) + (M) + (C) A ← (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 [1] ADD X	Add without Carry	A ← (A) + (M)	↕	↕	IMM DIR TNY IX DIR	AB BB 6*x* 6E 6F	ii dd	2 3 3 3 3
AND #*opr8i* AND *opr8a* AND ,X [1] AND X	Logical AND	A ← (A) & (M) A ← (A) & (X)	↕	—	IMM DIR IX DIR	A4 B4 B4 B4	ii dd 0E 0F	2 3 3 3
ASLA [1]	Arithmetic Shift Left		↕	↕	INH	48		1
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + $0002 + *rel*, if (C) = 0	—	—	REL	34	rr	3
BCLR *n*,*opr8a* BCLR *n*,D[X] BCLR *n*,X	Clear Bit n in Memory	Mn ← 0	—	—	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7) IX (b0) IX (b1) IX (b2) IX (b3) IX (b4) IX (b5) IX (b6) IX (b7) DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F 11 13 15 17 19 1B 1D 1F 11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd 0E 0E 0E 0E 0E 0E 0E 0E 0F 0F 0F 0F 0F 0F 0F 0F	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + $0002 + *rel*, if (C) = 1	—	—	REL	35	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + $0002 + *rel*, if (Z) = 1	—	—	REL	37	rr	3
BGND	Background	Enter Background Debug Mode	—	—	INH	BF		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 2 of 6)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BHS *rel* (1)	Branch if Higher or Same (Same as BCC)	PC ← (PC) + $0002 + *rel*, if (C) = 0	—	—	REL	34	rr	3
BLO *rel* (1)	Branch if Lower (Same as BCS)	PC ← (PC) + $0002 + *rel*, if (C) = 1	—	—	REL	35	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + $0002 + *rel*, if (Z) = 0	—	—	REL	36	rr	3
BRA *rel*	Branch Always	PC ← (PC) + $0002 + *rel*	—	—	REL	30	rr	3
BRN *rel* (1)	Branch Never	PC ← (PC) + $0002	—	—	REL	30	00	3
BRCLR *n*,*opr8a*,*rel*	Branch if Bit *n* in Memory Clear	PC ← (PC) + $0003 + *rel*, 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
BRCLR *n*,D[X],*rel*					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
BRCLR *n*,X,*rel*					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

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 6)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BRSET *n*,*opr8a*,*rel*	Branch if Bit *n* in Memory Set	PC ← (PC) + $0003 + *rel*, 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
BRSET *n*,D[X],*rel*					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
BRSET *n*,X,*rel*					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
BSET *n*,*opr8a*	Set Bit *n* in Memory	Mn ← 1	—	—	DIR (b0)	10	dd	5
					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 *n*,D[X]					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
BSET *n*,X					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

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.

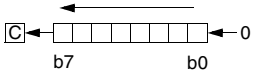
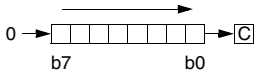
Table 8-1. Instruction Set Summary (Sheet 4 of 6)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BSR *rel*	Branch Subroutine	PC ← (PC) + 2 Push PC to shadow PC PC ← (PC) + *rel*	—	—	REL	AD	rr	3
CBEQA #*opr8i*,*rel* CBEQ *opr8a*,*rel* CBEQ ,X,*rel* (1),(2) CBEQ X,*rel* (1)	Compare and Branch if Equal	PC ← (PC) + $0003 + *rel*, if (A) – (M) = $00 PC ← (PC) + $0003 + *rel*, if (A) – (M) = $00 PC ← (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 ← 0	—	0	INH	38		1
CLR *opr8a* CLR *opr5a* CLR ,X (1) CLRA CLRX (1)	Clear	M ← $00 A ← $00 X ← $00	1	—	DIR SRT IX INH INH	3F 8*x* / 9*x* 8E 4F 8F	dd	3 2 2 1 2
CMP #*opr8i* CMP *opr8a* CMP ,X (1) CMP X (1)	Compare Accumulator with Memory	(A) – (M) (A) – (X)	↕	↕	IMM DIR IX INH	A1 B1 B1 B1	ii dd 0E 0F	2 3 3 3
COMA	Complement (One's Complement)	A ← ($\overline{A}$)	↕	1	INH	43		1
DBNZ *opr8a*,*rel* DBNZ ,X,rel (1) DBNZA *rel* DBNZX rel (1)	Decrement and Branch if Not Zero	A ← (A) – $01 or M ← (M) - $01 PC ← (PC) + $0003 + *rel* if (result) ≠ 0 for DBNZ direct PC ← (PC) + $0002 + *rel* if (result) ≠ 0 for DBNZA X ← (X) – $01 PC ← (PC) + $0003 + *rel* if (result) ≠ 0	—	—	DIR IX INH INH	3B 3B 4B 3B	dd rr 0E rr rr 0F rr	7 7 4 7
DEC *opr8a* DEC *opr4a* DEC ,X (1) DECA DEC X	Decrement	M ← (M) – $01 A ← (A) – $01 X ← (X) – $01	↕	—	DIR TNY IX INH DIR	3A 5*x* 5E 4A 5F	dd	5 4 4 1 4
EOR #*opr8i* EOR *opr8a* EOR ,X (1) EOR X	Exclusive OR Memory with Accumulator	A ← (A ⊕ M) A ← (A ⊕ X)	↕	—	IMM DIR IX DIR	A8 B8 B8 B8	ii dd 0E 0F	2 3 3 3
INC *opr8a* INC *opr4a* INC ,X (1) INCA INCX (1)	Increment	M ← (M) + $01 A ← (A) + $01 X ← (X) + $01	↕	—	DIR TNY IX INH INH	3C 2*x* 2E 4C 2F	dd	5 4 4 1 4
JMP *opr16a*	Jump	PC ← Effective Address	—	—	EXT	BC	hh ll	4
JSR *opr16a*	Jump to Subroutine	PC ← (PC) + 3 Push PC to shadow PC PC ← Effective Address	—	—	EXT	BD	hh ll	4
LDA #*opr8i* LDA *opr8a* LDA *opr5a* LDA ,X (1)	Load Accumulator from Memory	A ← (M)	↕	—	IMM DIR SRT IX	A6 B6 C*x*/D*x* CE	ii dd	2 3 3 3

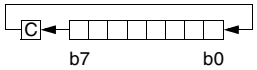
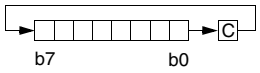
1. This is a pseudo instruction supported by the normal RS08 instruction set.

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 5 of 6)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
LDX #*opr8i* (1) LDX *opr8a* (1) LDX ,X (1)	Load Index Register from Memory	$0F ← (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 *oprr8a,oprr8a*
MOV #*opr8i,oprr8a*
MOV D[X],*opr8a*
MOV *opr8a*,D[X]
MOV #*opr8i*,D[X]
	Move	$(M)_{destination} ← (M)_{source}$	↕	—	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 #*opr8i* ORA *opr8a* ORA ,X (1) ORA X	Inclusive OR Accumulator and Memory	A ← (A) | (M) A ← (A) | (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 #*opr8i* SBC *opr8a* SBC ,X (1) SBC X	Subtract with Carry	A ← (A) – (M) – (C) A ← (A) – (X) – (C)	↕	↕	IMM DIR IX DIR	A2 B2 B2 B2	ii dd 0E 0F	2 3 3 3
SEC	Set Carry Bit	C ← 1	—	1	INH	39		1
SHA	Swap Shadow PC High with A	A ⇔ SPCH	—	—	INH	45		1
SLA	Swap Shadow PC Low with A	A ⇔ SPCL	—	—	INH	42		1
STA *opr8a* STA *opr5a* STA ,X (1) STA X	Store Accumulator in Memory	M ← (A)	↕	—	DIR SRT IX SRT	B7 E*x* / F*x* EE EF	dd	3 2 2 2
STX *opr8a* (1)	Store Index Register in Memory	M ← (X)	↕	—	DIR	4E	0F dd	5
STOP	Put MCU into stop mode		—	—	INH	AE		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 6 of 6)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
SUB #*opr8i* SUB *opr8a* SUB *opr4a* SUB ,X (1) SUB X	Subtract	A ← (A) – (M) A ← (A) – (X)	↕	↕	IMM DIR TNY IX DIR	A0 B0 7*x* 7E 7F	ii dd	2 3 3 3 3
TAX(1)	Transfer A to X	X ← (A)	↕	—	INH	EF		2
TST opr8a (1) TSTA (1) TST ,X (1) TSTX (1)	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(1)	Transfer X to A	A ← (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

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

INH	**Inherent**	**REL**	**Relative**
IMM	**Immediate**	**SRT**	**Short**
DIR	**Direct**	**TNY**	**Tiny**
EXT	**Extended**		
DD	**Direct-Direct**	**IMD**	**Immediate-Direct**

High Byte of Opcode in Hexadecimal **B**

Gray box is decoded as illegal instruction

Low Byte of Opcode in Hexadecimal **0**

3 SUB 2 DIR

RS08 Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode

Chapter 9
16-Bit Timer/PWM (S08TPMV3)

9.1 Introduction

The TPM uses one input/output (I/O) pin per channel, TPMCHn where n is the channel number (such as 0–1). The TPM shares its I/O pins with general-purpose I/O port pins (refer to the Chapter 2, “Pins and Connections.”)

The TPM module pins, TPMCH0 and TPMCH1 can be repositioned under software control using TPMCH0PS in SOPT1 (Table 9-1). TPMCH0PS and TPMCH1PS in SOPT1 select general-purpose I/O ports to be associated with TPM.

Table 9-1. TPM Position Options

TPMCH0PS in SOPT1	Port Pin for TPMCH0
0 (default)	PTA0
1	PTB4

TPMCH1PS in SOPT1	Port Pin for TPMCH1
0 (default)	PTA1
1	PTB5

Figure 9-1 shows the MC9RS08KB12 series block diagram with the TPM block and pins highlighted.

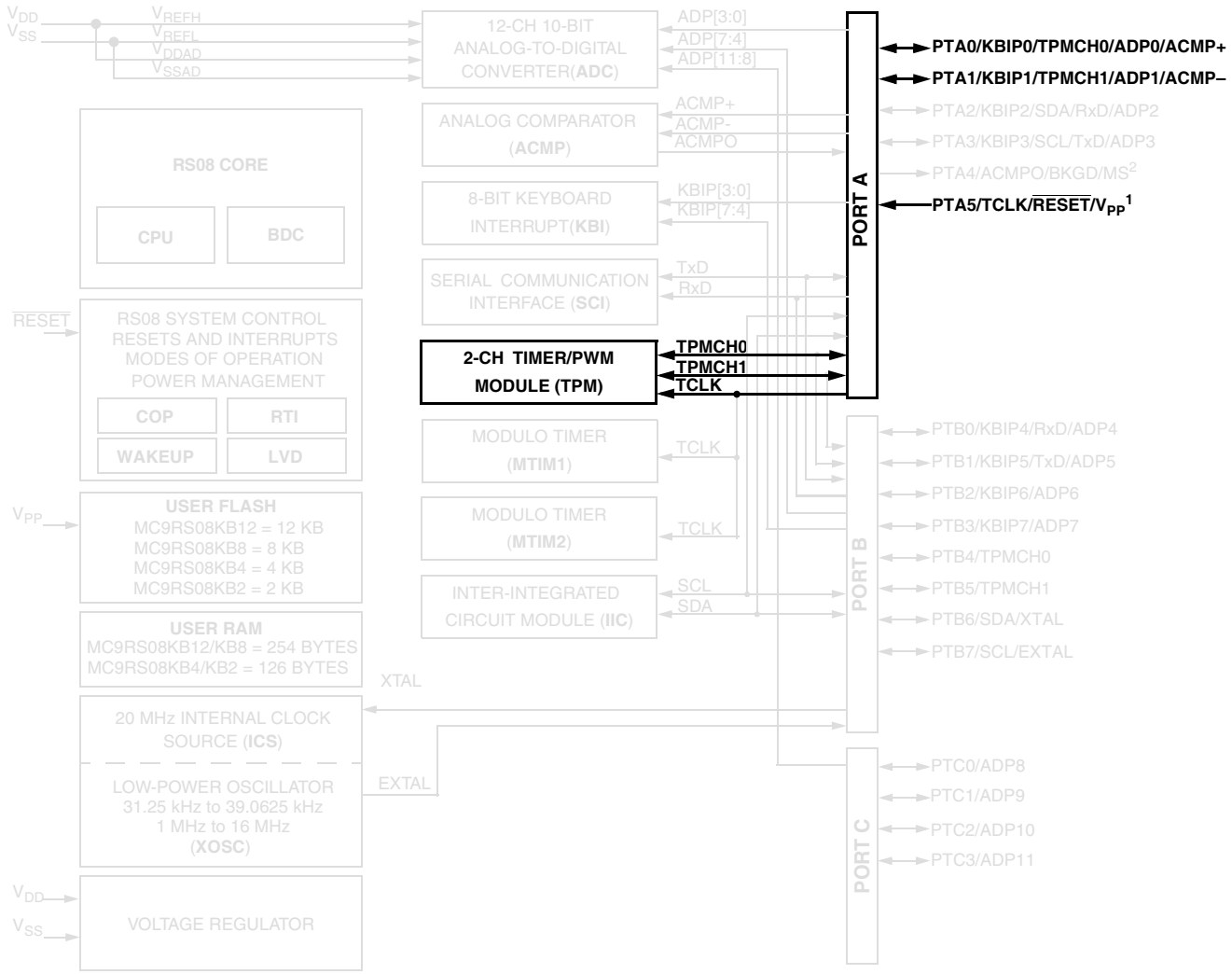


Figure 9-1. MC9RS08KB12 Series Block Diagram Highlighting TPM Block and Pins

9.1.1 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

9.1.2 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.

9.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMCHn (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 9-2 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.

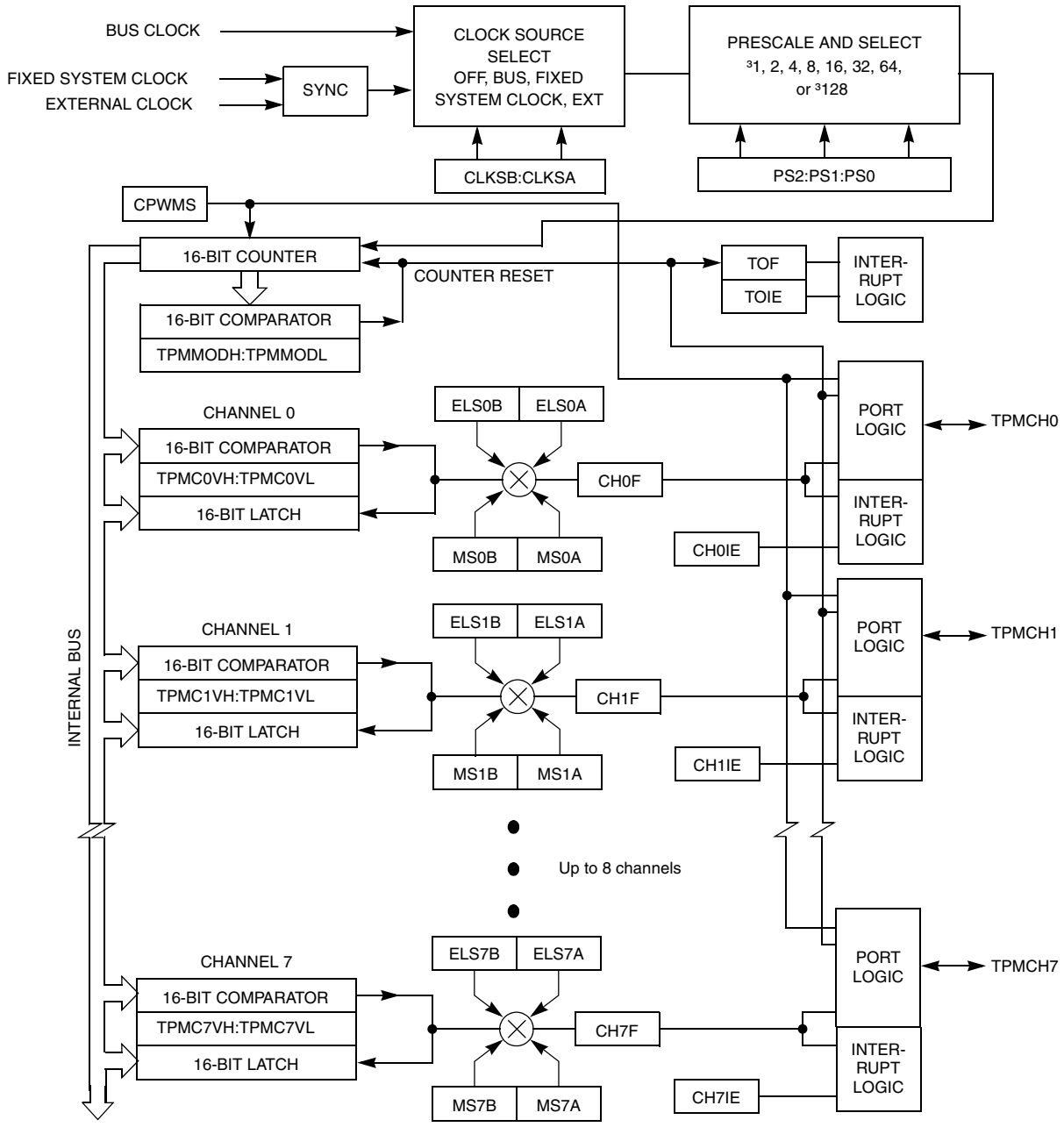


Figure 9-2. TPM Block Diagram

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.

9.2 Signal Description

Table 9-2 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 9-2. Signal Properties

Name	Function
EXTCLK[1]	External clock source which may be selected to drive the TPM counter.
TPMCHn[2]	I/O pin associated with TPM channel n

1 When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

2 n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

9.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 9-2 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.

9.2.1.1 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).

9.2.1.2 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 (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the 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
TPMCnVH:TPMCnVL = 0x0005

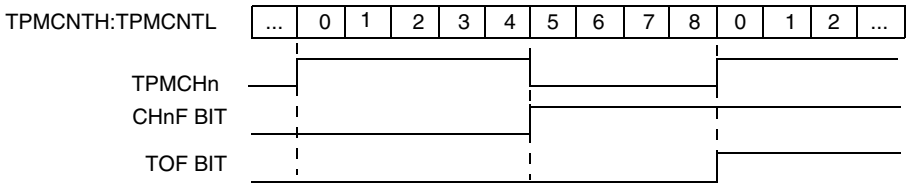


Figure 9-3. High-True Pulse of an Edge-Aligned PWM

TPMMODH:TPMMODL = 0x0008
TPMCnVH:TPMCnVL = 0x0005

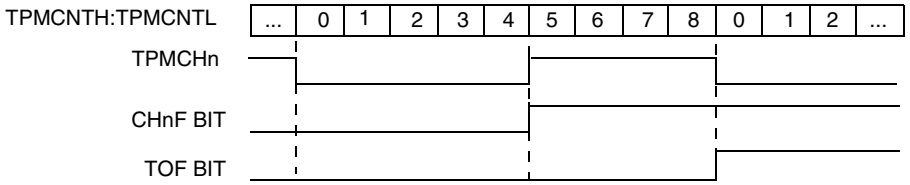


Figure 9-4. Low-True Pulse of an Edge-Aligned PWM

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
TPMCnVH:TPMCnVL = 0x0005

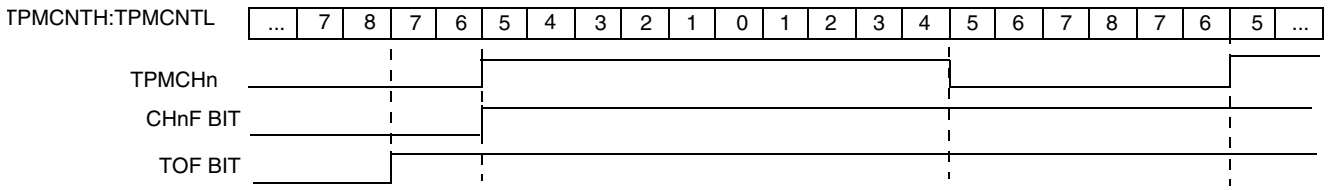


Figure 9-5. High-True Pulse of a Center-Aligned PWM

TPMMODH:TPMMODL = 0x0008
TPMCnVH:TPMCnVL = 0x0005

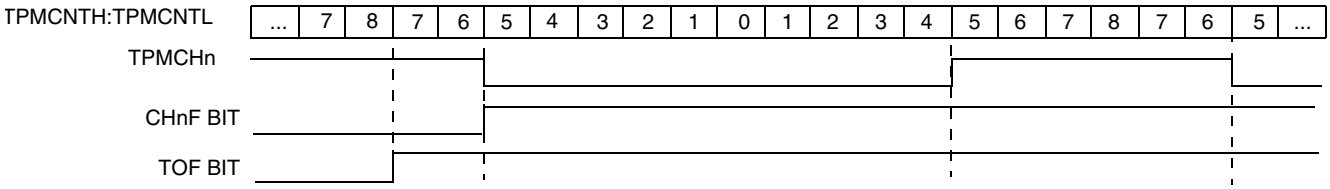


Figure 9-6. Low-True Pulse of a Center-Aligned PWM

9.3 Register Definition

This section consists of register descriptions in address order.

9.3.1 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.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 9-7. TPM Status and Control Register (TPMSC)

Table 9-3. 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[B:A]	Clock source selects. As shown in Table 9-4, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based or FLL-based system clock. When there is no PLL or FLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL or FLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL or FLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 9-5. 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 9-4. TPM-Clock-Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Table 9-5. Prescale Factor Selection

PS2:PS1:PS0	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

9.3.2 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.

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W	Any write to TPMCNTH clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 9-8. TPM Counter Register High (TPMCNTH)

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMCNTL clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 9-9. 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.

In BDM mode, writing any value to TPMSC, TPMCNTH or TPMCNTL registers resets the read coherency mechanism of the TPMCNTH:L registers, regardless of the data involved in the write.

9.3.3 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 CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), then the registers are updated after both bytes were written, and the TPM counter changes from (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 0xFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 9-10. TPM Counter Modulo Register High (TPMMODH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 9-11. TPM Counter Modulo Register Low (TPMMODL)

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

9.3.4 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.

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0							
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-12. TPM Channel n Status and Control Register (TPMCnSC)

Table 9-6. 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 Table 9-7.

Table 9-6. TPMCnSC Field Descriptions (continued)

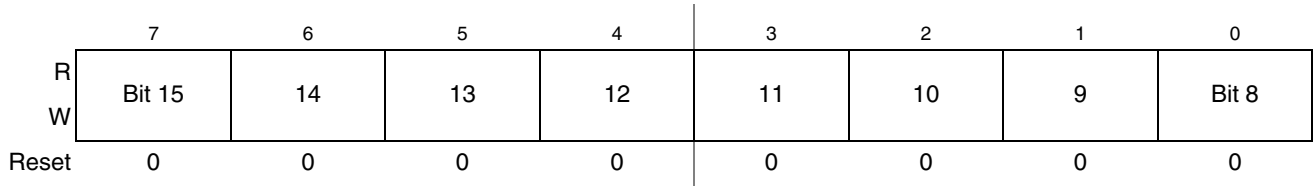
Field	Description
4 MSnA	Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 9-7 for a summary of channel mode and setup controls. **Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 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 9-7, 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 9-7. 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	00	Output compare	Software compare only
		01		Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

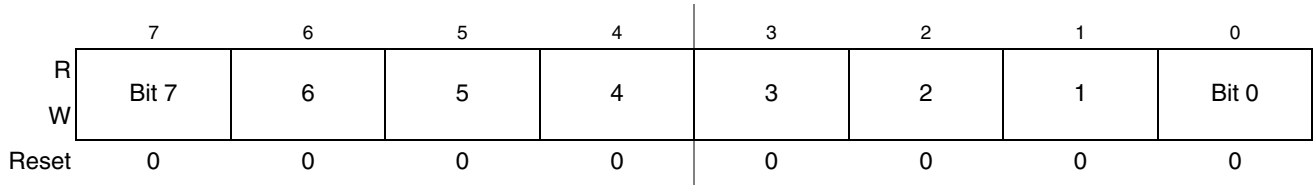
9.3.5 TPM Channel Value Registers (TPMCnVH:TPMCnVL)

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.



	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 9-13. TPM Channel Value Register High (TPMCnVH)



	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 9-14. 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 (unless reset by writing to TPMCnSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the 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 CLKSB:CLKSA bits and the selected mode, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written.
- If (CLKSB:CLKSA not = 0:0 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKSB:CLKSA not = 0:0 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (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 0xFFFE 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.

9.4 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.

9.4.1 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.

9.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 9-4. After any MCU reset, CLKSB:CLKSA=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

Table 9-8. TPM Clock Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL and FLL or the PLL and FLL are not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL or FLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

9.4.1.2 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).

9.4.1.3 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).

9.4.1.4 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.

9.4.2 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.

9.4.2.1 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.

In input capture mode, the TPMCnVH and TPMCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (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.

9.4.2.2 Output Compare Mode

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

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

9.4.2.3 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 9-15). 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.

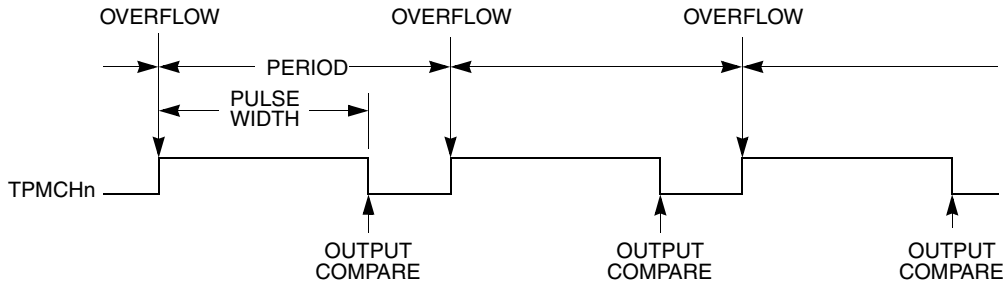


Figure 9-15. PWM Period and Pulse Width (ELSnA=0)

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 CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (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 0xFFFE to 0xFFFF.

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

pulse width = 2 x (TPMCnVH:TPMCnVL)

period = 2 x (TPMMODH:TPMMODL); TPMMODH:TPMMODL=0x0001-0x7FFF

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

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 9-16). 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.

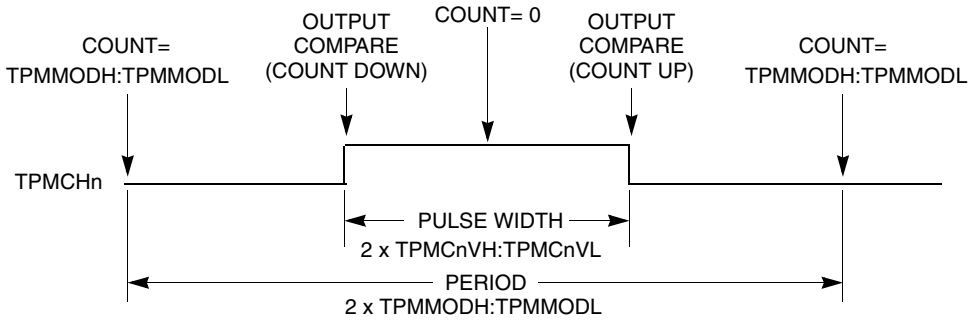


Figure 9-16. CPWM Period and Pulse Width (ELSnA=0)

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 CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (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 0xFFFE 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.

9.5 Reset Overview

9.5.1 General

The TPM is reset whenever any MCU reset occurs.

9.5.2 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).

9.6 Interrupts

9.6.1 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 9-9 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 9-9. 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.

9.6.2 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.

9.6.2.1 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.

9.6.2.1.1 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.

9.6.2.1.2 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.

9.6.2.2 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).

9.6.2.2.1 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 9.6.2, "Description of Interrupt Operation."

9.6.2.2.2 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 9.6.2, "Description of Interrupt Operation."

9.6.2.2.3 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 9.6.2, "Description of Interrupt Operation."

1. Write to TPMxCNTH:L registers (Section 9.3.2, "TPM-Counter Registers (TPMCNTH:TPMCNTL),") [SE110-TPM case 7]

   Any write to TPMxCNTH or TPMxCNTL registers in TPM v3 clears the TPM counter (TPMxCNTH:L) and the prescaler counter. Instead, in the TPM v2 only the TPM counter is cleared in this case.
2. Read of TPMxCNTH:L registers (Section 9.3.2, "TPM-Counter Registers (TPMCNTH:TPMCNTL),")
   - In TPM v3, any read of TPMxCNTH:L registers during BDM mode returns the value of the TPM counter that is frozen. In TPM v2, if only one byte of the TPMxCNTH:L registers was read before the BDM mode became active, then any read of TPMxCNTH:L registers during BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the frozen TPM counter value.

— This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxSC, TPMxCNTH or TPMxCNTL. Instead, in these conditions the TPM v2 does not clear this read coherency mechanism.

3. Read of TPMxCnVH:L registers (Section 9.3.5, “TPM Channel Value Registers (TPMCnVH:TPMCnVL).”)
   — In TPM v3, any read of TPMxCnVH:L registers during BDM mode returns the value of the TPMxCnVH:L register. In TPM v2, if only one byte of the TPMxCnVH:L registers was read before the BDM mode became active, then any read of TPMxCnVH:L registers during BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the value in the TPMxCnVH:L registers.
   — This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxCnSC. Instead, in this condition the TPM v2 does not clear this read coherency mechanism.
4. Write to TPMxCnVH:L registers
   — Input Capture Mode (Section 9.4.2.1, “Input Capture Mode.”)

     In this mode the TPM v3 does not allow the writes to TPMxCnVH:L registers. Instead, the TPM v2 allows these writes.
   — Output Compare Mode (Section 9.4.2.2, “Output Compare Mode.”)

     In this mode and if (CLKSB:CLKSA not = 0:0), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written. Instead, the TPM v2 always updates these registers when their second byte is written.

     The following procedure can be used in the TPM v3 to verify if the TPMCnVH:L registers were updated with the new value that was written to these registers (value in their write buffer).

     ...

     write the new value to TPMCnVH:L;

     read TPMCnVH and TPMCnVL registers;

     while (the read value of TPMCnVH:L is different from the new value written to TPMCnVH:L)

     begin

      read again TPMCnVH and TPMCnVL;

     end

     ...

     In this point, the TPMCnVH:L registers were updated, so the program can continue and, for example, write to TPMC0SC without cancelling the previous write to TPMCnVH:L registers.
   — Edge-Aligned PWM (Section 9.4.2.3, “Edge-Aligned PWM Mode.”)

     In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from $FFFE to $FFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to $0000.

— Center-Aligned PWM (Section 9.4.2.4, “Center-Aligned PWM Mode.”)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from $FFFE to $FFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to (TPMxMODH:L - 1).

5. Center-Aligned PWM (Section 9.4.2.4, “Center-Aligned PWM Mode.”)

— TPMxCnVH:L = TPMxMODH:L [SE110-TPM case 1]

In this case, the TPM v3 produces 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

— TPMxCnVH:L = (TPMxMODH:L - 1) [SE110-TPM case 2]

In this case, the TPM v3 produces almost 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

— TPMxCnVH:L is changed from 0x0000 to a non-zero value [SE110-TPM case 3 and 5]

In this case, the TPM v3 waits for the start of a new PWM period to begin using the new duty cycle setting. Instead, the TPM v2 changes the channel output at the middle of the current PWM period (when the count reaches 0x0000).

— TPMxCnVH:L is changed from a non-zero value to 0x0000 [SE110-TPM case 4]

In this case, the TPM v3 finishes the current PWM period using the old duty cycle setting. Instead, the TPM v2 finishes the current PWM period using the new duty cycle setting.

6. Write to TPMxMODH:L registers in BDM mode (Section 9.3.3, “TPM Counter Modulo Registers (TPMMODH:TPMMODL).”)

In the TPM v3 a write to TPMxSC register in BDM mode clears the write coherency mechanism of TPMxMODH:L registers. Instead, in the TPM v2 this coherency mechanism is not cleared when there is a write to TPMxSC register.

7. Update of EPWM signal when CLKSB:CLKSA = 00

In the TPM v3 if CLKSB:CLKSA = 00, then the EPWM signal in the channel output is not update (it is frozen while CLKSB:CLKSA = 00). Instead, in the TPM v2 the EPWM signal is updated at the next rising edge of bus clock after a write to TPMCnSC register.

The Figure 9-10 and Figure 9-11 show when the EPWM signals generated by TPM v2 and TPM v3 after the reset (CLKSB:CLKSA = 00) and if there is a write to TPMCnSC register.

EPWM mode
TPMMODH:TPMMODL = 0x0007
TPMCnVH:TPMCnVL = 0x0005

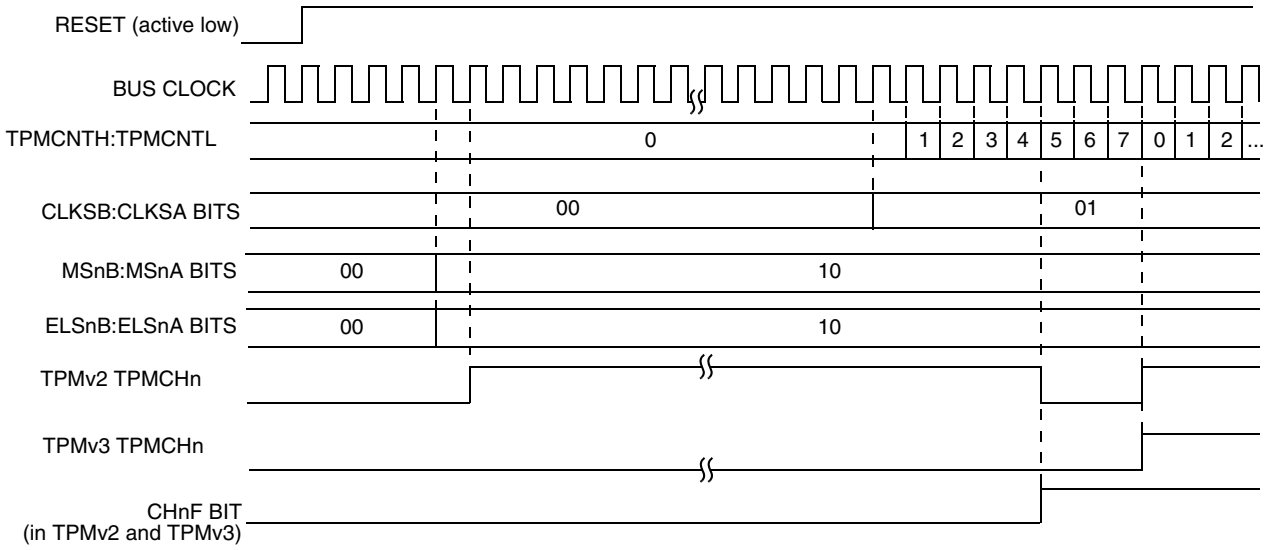


Table 9-10. Generation of high-true EPWM signal by TPM v2 and v3 after the reset

EPWM mode
TPMMODH:TPMMODL = 0x0007
TPMCnVH:TPMCnVL = 0x0005

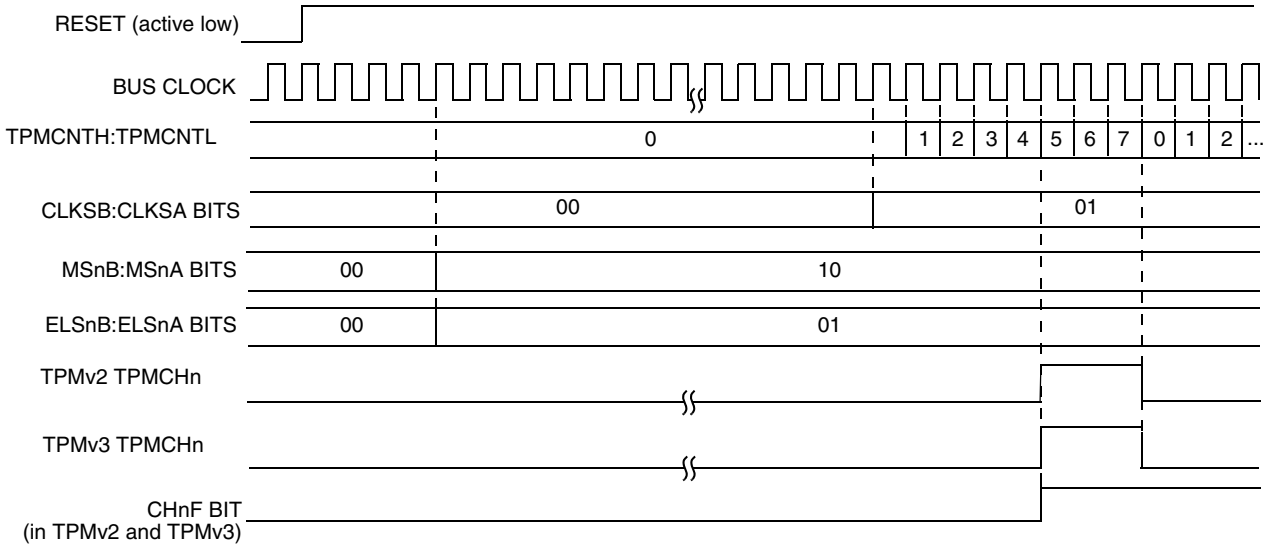


Table 9-11. Generation of low-true EPWM signal by TPM v2 and v3 after the reset

The following procedure can be used in TPM v3 (when the channel pin is also a port pin) to emulate the high-true EPWM generated by TPM v2 after the reset.

...

configure the channel pin as output port pin and set the output pin;

configure the channel to generate the EPWM signal but keep ELSnB:ELSnA as 00;

configure the other registers (TPMMODH, TPMMODL, TPMCnVH, TPMCnVL, ...);

configure CLKSB:CLKSA bits (TPM v3 starts to generate the high-true EPWM signal, however TPM does not control the channel pin, so the EPWM signal is not available);

wait until the TOF is set (or use the TOF interrupt);

enable the channel output by configuring ELSnB:ELSnA bits (now EPWM signal is available);

...

Chapter 10
Serial Communications Interface (S08SCIV4)

10.1 Introduction

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI shares its I/O pins with general-purpose I/O port pins (refer to the Chapter 2, “Pins and Connections”.)

The SCI module I/O, TxD and RxD, can be repositioned under software control using SCIS in SOPT1 register (Table 10-1).

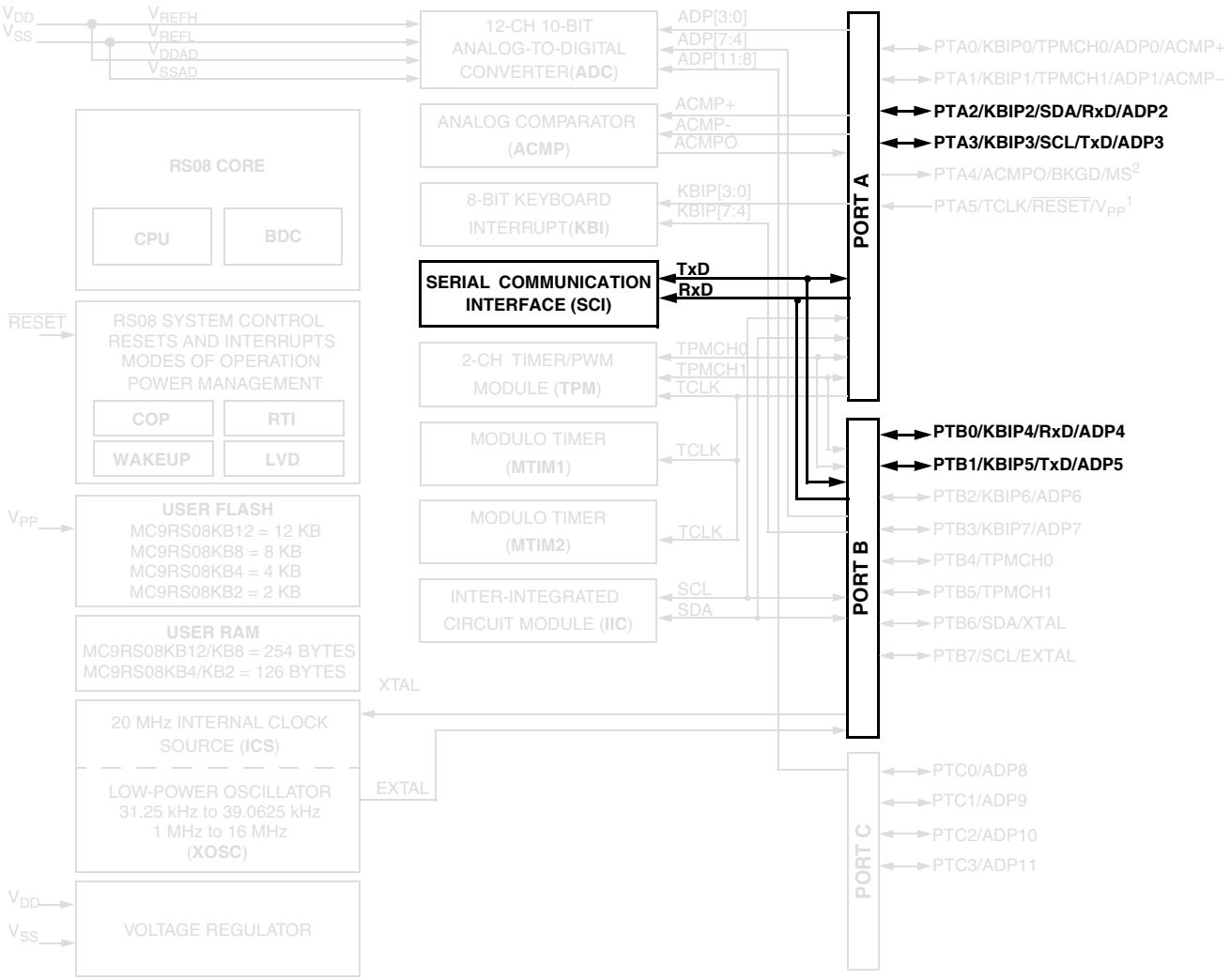
Table 10-1. SCI I/O Position Options

SCIS in SOPT1	Port pin for RxD	Port pin for TxD
0 (default)	PTB0	PTB1
1	PTA2	PTA3

NOTE

MC9RS08KB12 series MCUs DONOT support stop1 or stop2, neglect those information in this chapter. The stop mode in the other chapters is the stop3 in this chapter.

Figure 10-1 shows the MC9RS08KB12 series block diagram, highlighting the SCI block and pins.



NOTES:
1. PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} is an input-only pin when used as port pin
2. PTA4/ACMPO/BKGD/MS is an output-only pin when used as port pin

Figure 10-1. MC9RS08KB12 Series Block Diagram Highlighting SCI Block and Pins

10.1.1 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

10.1.2 Modes of Operation

See Section 10.3, “Functional Description,” For details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

10.1.3 Block Diagram

Figure 10-2 shows the transmitter portion of the SCI.

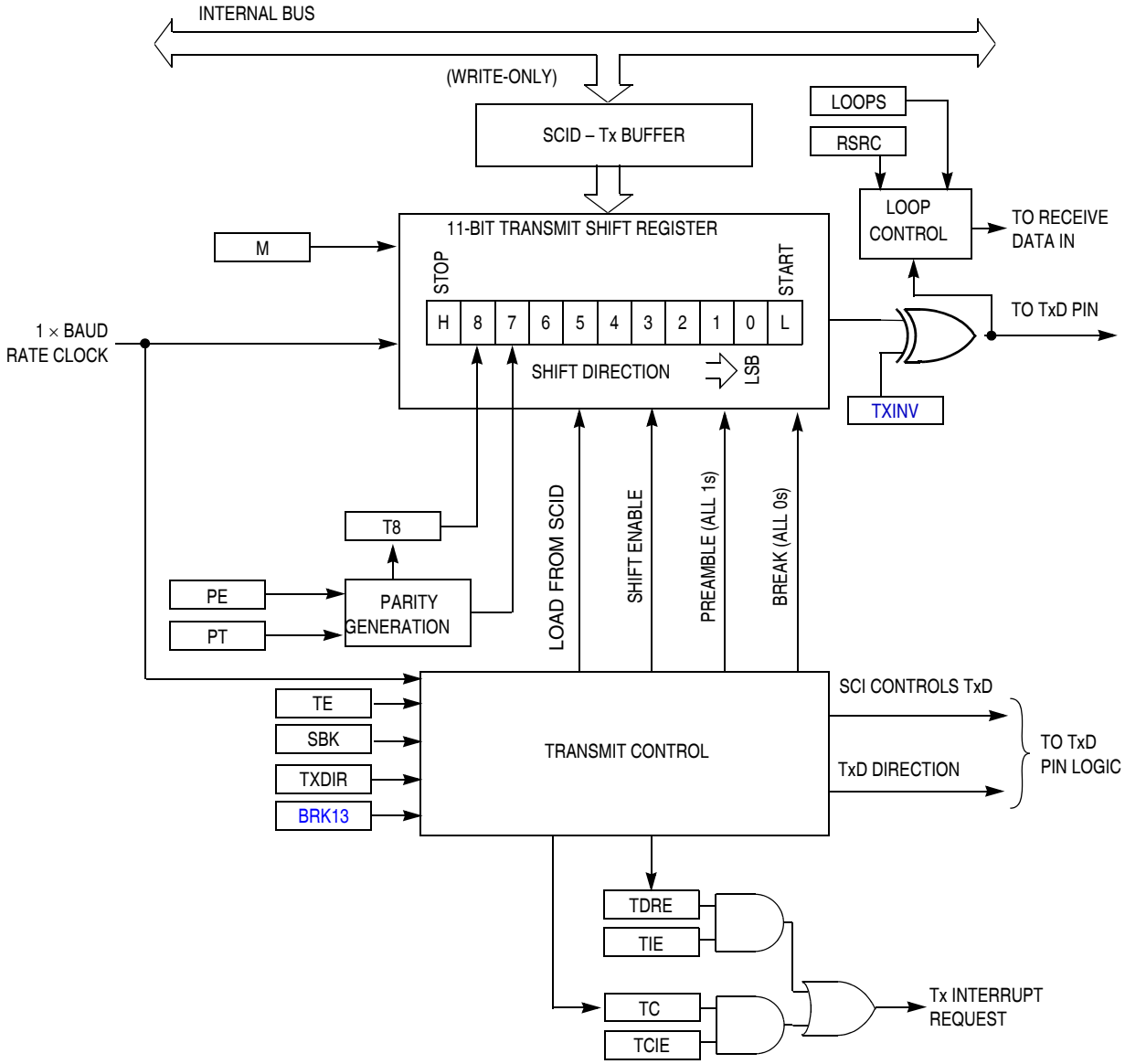


Figure 10-2. SCI Transmitter Block Diagram

Figure 10-3 shows the receiver portion of the SCI.

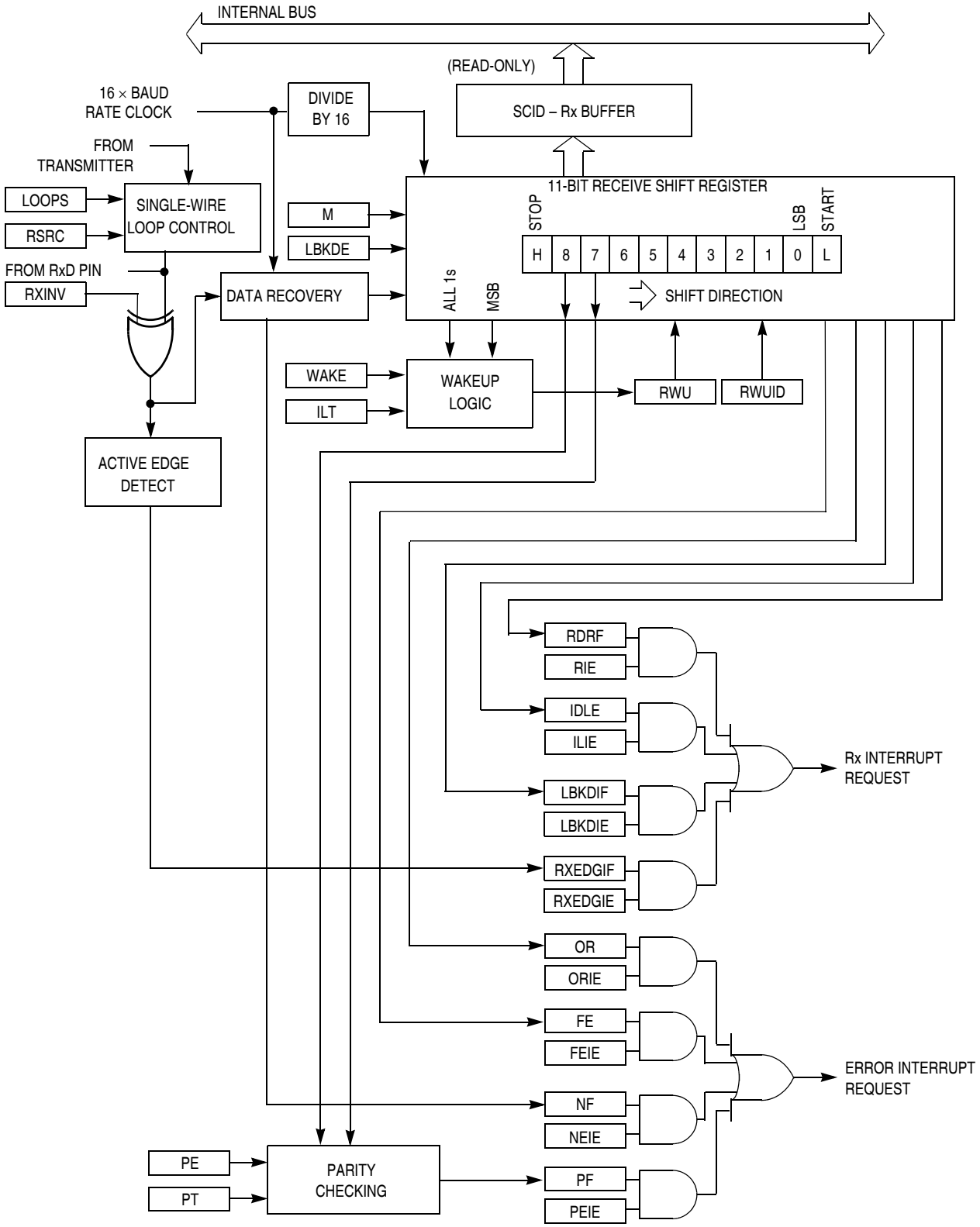


Figure 10-3. SCI Receiver Block Diagram

10.2 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.

10.2.1 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).

	7	6	5	4	3	2	1	0
R	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 10-4. SCI Baud Rate Register (SCIBDH)

Table 10-2. SCIBDH Field Descriptions

Field	Description
7 LBKDIE	**LIN Break Detect Interrupt Enable (for LBKDIF)** 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	**RxD Input Active Edge Interrupt Enable (for RXEDGIF)** 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	**Baud Rate Modulo Divisor** — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 10-3.

	7	6	5	4	3	2	1	0
R W	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Reset	0	0	0	0	0	1	0	0

Figure 10-5. SCI Baud Rate Register (SCIBDL)

Table 10-3. SCIBDL Field Descriptions

Field	Description
7:0 SBR[7:0]	**Baud Rate Modulo Divisor** — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 10-2.

10.2.2 SCI Control Register 1 (SCIC1)

This read/write register is used to control various optional features of the SCI system.

	7	6	5	4	3	2	1	0
R W	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

Figure 10-6. SCI Control Register 1 (SCIC1)

Table 10-4. SCIC1 Field Descriptions

Field	Description
7 LOOPS	**Loop Mode Select** — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 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 RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	**SCI Stops in Wait Mode** 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	**Receiver Source Select** — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 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	**9-Bit or 8-Bit Mode Select** 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.

Table 10-4. SCIC1 Field Descriptions (continued)

Field	Description
3 WAKE	**Receiver Wakeup Method Select** — Refer to Section 10.3.3.2, "Receiver Wakeup Operation" for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	**Idle Line Type Select** — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 10.3.3.2.1, "Idle-Line Wakeup" for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	**Parity Enable** — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	**Parity Type** — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

10.2.3 SCI Control Register 2 (SCIC2)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Figure 10-7. SCI Control Register 2 (SCIC2)

Table 10-5. SCIC2 Field Descriptions

Field	Description
7 TIE	**Transmit Interrupt Enable (for TDRE)** 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	**Transmission Complete Interrupt Enable (for TC)** 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	**Receiver Interrupt Enable (for RDRF)** 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	**Idle Line Interrupt Enable (for IDLE)** 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.

Table 10-5. SCIC2 Field Descriptions (continued)

Field	Description
3 TE	**Transmitter Enable** 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 Section 10.3.2.1, “Send Break and Queued Idle” 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.
2 RE	**Receiver Enable** — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	**Receiver Wakeup Control** — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 10.3.3.2, “Receiver Wakeup Operation” for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	**Send Break** — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 10.3.2.1, “Send Break and Queued Idle” for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

10.2.4 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.

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 10-8. SCI Status Register 1 (SCIS1)

Table 10-6. SCIS1 Field Descriptions

Field	Description
7 TDRE	**Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIS1 with TDRE = 1 and then write to the SCI data register (SCID). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	**Transmission Complete Flag** — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SCIS1 with TC = 1 and then doing one of the following three things: • Write to the SCI data register (SCID) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCIC2
5 RDRF	**Receive Data Register Full Flag** — 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). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	**Idle Line Flag** — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. 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. 0 No idle line detected. 1 Idle line was detected.
3 OR	**Receiver Overrun Flag** — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from 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). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	**Noise Flag** — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIS1 and then read the SCI data register (SCID). 0 No noise detected. 1 Noise detected in the received character in SCID.

Table 10-6. SCIS1 Field Descriptions (continued)

Field	Description
1 FE	**Framing Error Flag** — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read 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	**Parity Error Flag** — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIS1 and then read the SCI data register (SCID). 0 No parity error. 1 Parity error.

10.2.5 SCI Status Register 2 (SCIS2)

This register has one read-only status flag.

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 10-9. SCI Status Register 2 (SCIS2)

Table 10-7. SCIS2 Field Descriptions

Field	Description
7 LBKDIF	**LIN Break Detect Interrupt Flag** — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	**RxD Pin Active Edge Interrupt Flag** — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV[1]	**Receive Data Inversion** — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	**Receive Wake Up Idle Detect**— 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	**Break Character Generation Length** — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 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 10-7. SCIS2 Field Descriptions (continued)

Field	Description
1 LBKDE	**LIN Break Detection Enable**— LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 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	**Receiver Active Flag** — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

[1] Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

10.2.6 SCI Control Register 3 (SCIC3)

	7	6	5	4	3	2	1	0
R	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 10-10. SCI Control Register 3 (SCIC3)

Table 10-8. SCIC3 Field Descriptions

Field	Description
7 R8	**Ninth Data Bit for Receiver** — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the 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	**Ninth Data Bit for Transmitter** — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the 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 should 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	**TxD Pin Direction in Single-Wire Mode** — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

Table 10-8. SCIC3 Field Descriptions (continued)

Field	Description
4 TXINV[1]	**Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	**Overrun Interrupt Enable** — 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	**Noise Error Interrupt Enable** — 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	**Framing Error Interrupt Enable** — 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	**Parity Error Interrupt Enable** — 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.

[1] Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

10.2.7 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 10-11. SCI Data Register (SCID)

10.3 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.

10.3.1 Baud Rate Generation

As shown in Figure 10-12, the clock source for the SCI baud rate generator is the bus-rate clock.

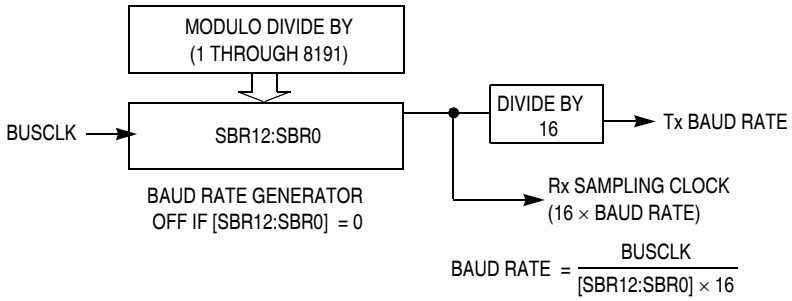


Figure 10-12. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ±4.5 percent for 8-bit data format and about ±4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

10.3.2 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 10-2.

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in 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.

10.3.2.1 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 10-9. 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

10.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 10-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in 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 10.3.5.1, “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 10.3.4, "Interrupts and Status Flags" for more details about flag clearing.

10.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

10.3.3.2 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.

10.3.3.2.1 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.

10.3.3.2.2 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.

10.3.4 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).

10.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

10.3.5.1 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.

10.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2. . An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

10.3.5.3 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.

10.3.5.4 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 11
Modulo Timer (S08MTIMV1)

11.1 Introduction

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

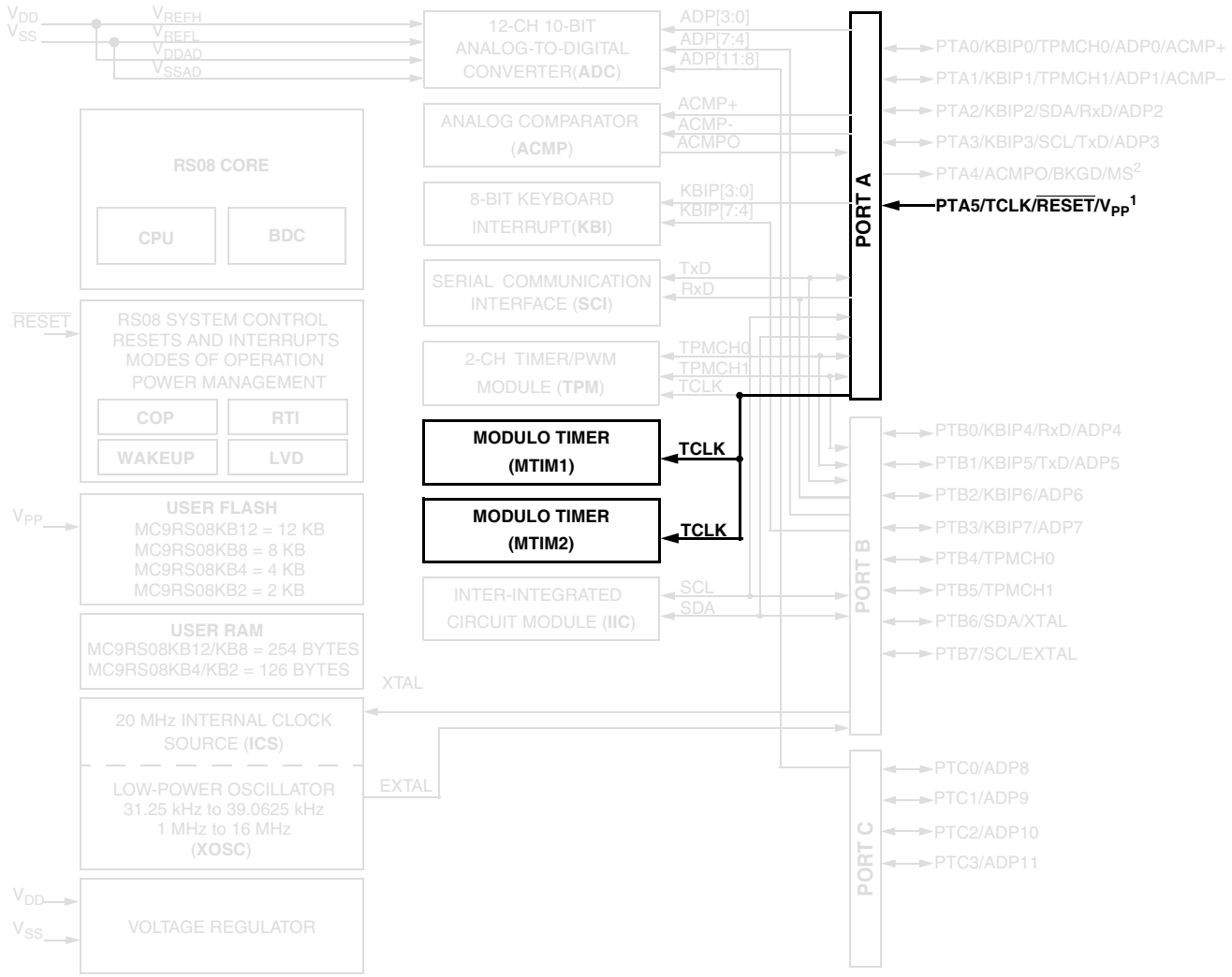
The central component of the MTIM is the 8-bit counter that can operate as a free-running counter or a modulo counter. A timer overflow interrupt can be enabled to generate periodic interrupts for time-based software loops.

There are two MTIM modules in MC9RS08KB12 series. Each has three optional reference clocks in MC9RS08KB12 series. They are TCLK clock, ICSFFCLK clock, and bus clock from ICS module. Additionally, the overflow of MTIM2 counter can be the input of MTIM1 when MTIM1EC bit in the SOPT2 register is set and TCLK clock is selected for MTIM1.

NOTE

MC9RS08KB12 series MCUs DO NOT support stop1 or stop2, neglect those information in this chapter. The stop mode in the other chapters is the stop3 in this chapter.

Figure 11-1 shows the MC9RS08KB12 series block diagram with the MTIM blocks and pins highlighted.



NOTES:
1. PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} is an input-only pin when used as port pin
2. PTA4/ACMPO/BKGD/MS is an output-only pin when used as port pin

Figure 11-1. MC9RS08KB12 Series Block Diagram Highlighting MTIM Blocks and Pins

11.1.1 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

11.1.2 Modes of Operation

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

11.1.2.1 MTIM in Wait Mode

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

11.1.2.2 MTIM in Stop Modes

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

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

11.1.2.3 MTIM in Active Background Mode

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

11.1.3 Block Diagram

The block diagram for the modulo timer module is shown Figure 11-2.

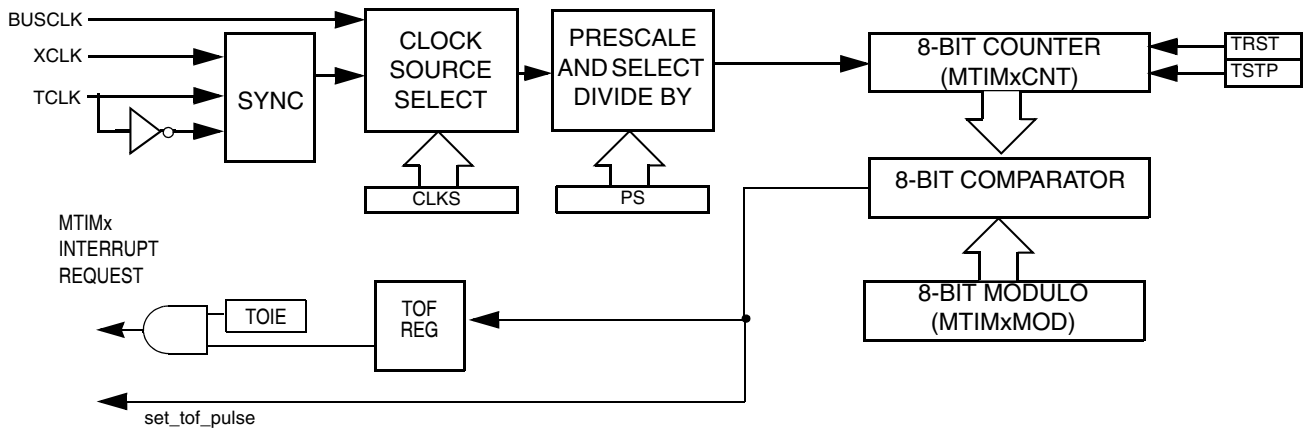


Figure 11-2. Modulo Timer (MTIM) Block Diagram

11.2 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 11-1.

Table 11-1. 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.

11.3 Register Definition

Figure 11-3 is a summary of MTIM registers.

Name		7	6	5	4	3	2	1	0
MTIMxSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMxCLK	R	0	0	CLKS		PS			
	W								
MTIMxCNT	R	COUNT							
	W								
MTIMxMOD	R	MOD							
	W								

Figure 11-3. MTIMx **Register Summary**

Each MTIM includes four registers:

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

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

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

11.3.1 MTIMx Status and Control Register (MTIMxSC)

MTIMxSC 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 11-4. MTIMx Status and Control Register

Table 11-2. MTIMx Status and Control Register Field Descriptions

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

11.3.2 MTIMx Clock Configuration Register (MTIMxCLK)

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

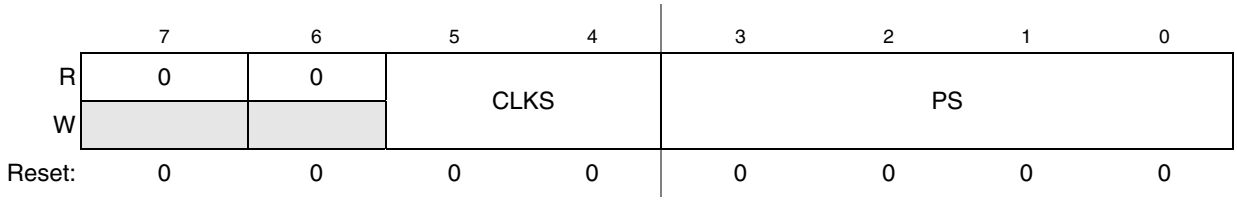


Figure 11-5. MTIMx Clock Configuration Register

Table 11-3. MTIMx Clock Configuration Register Field Description

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

11.3.3 MTIMx Counter Register (MTIMxCNT)

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

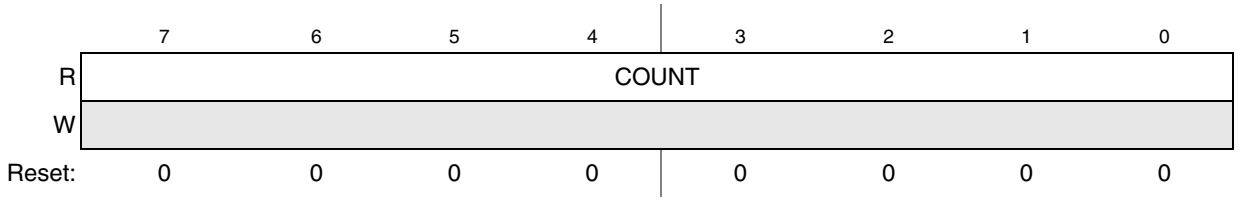


Figure 11-6. MTIMx Counter Register

Table 11-4. MTIMx Counter Register Field Description

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

11.3.4 MTIMx Modulo Register (MTIMxMOD)

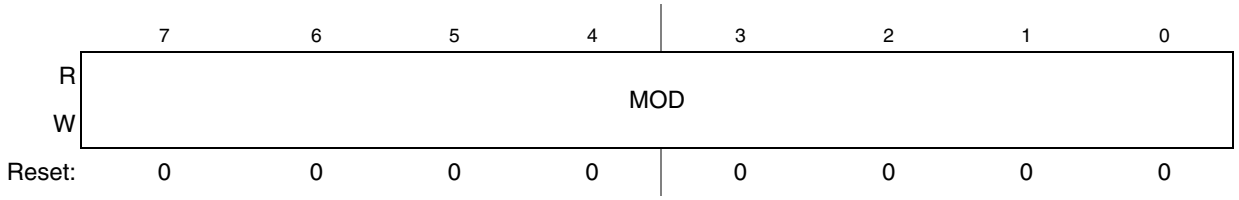


Figure 11-7. MTIMx Modulo Register

Table 11-5. MTIMx **Modulo Register Field Descriptions**

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

11.4 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 (MTIMxCNT) 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 (MTIMxSC) and clear the MTIM stop bit (TSTP).

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

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in MTIMxSC 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 (MTIMxMOD) 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 MTIMxMOD 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 MTIMxSC 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 MTIMxMOD 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 MTIMxSC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

11.4.1 MTIM Operation Example

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

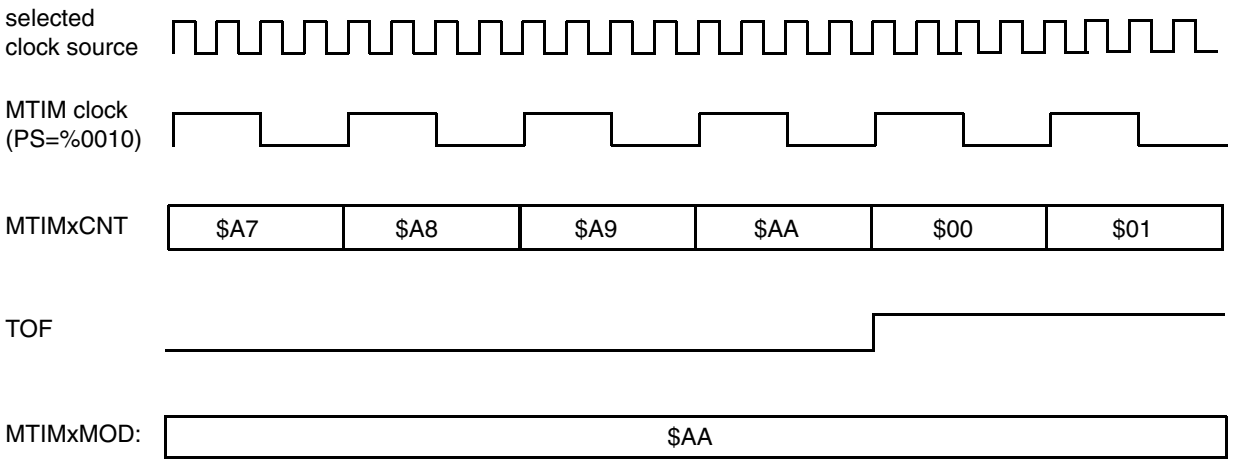


Figure 11-8. MTIM counter overflow example

In the example of Figure 11-8, the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMxMOD register is set to $AA. When the counter, MTIMxCNT, 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 12
10-Bit Analog-to-Digital Converter (S08ADC10V1)

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.

NOTE

MC9RS08KB12 series MCUs do not support stop1 or stop2, neglect those information in this chapter. The stop mode in the other chapters is the stop3 in this chapter.

Figure 12-1 shows the MC9RS08KB12 series with the ADC module and pins highlighted.

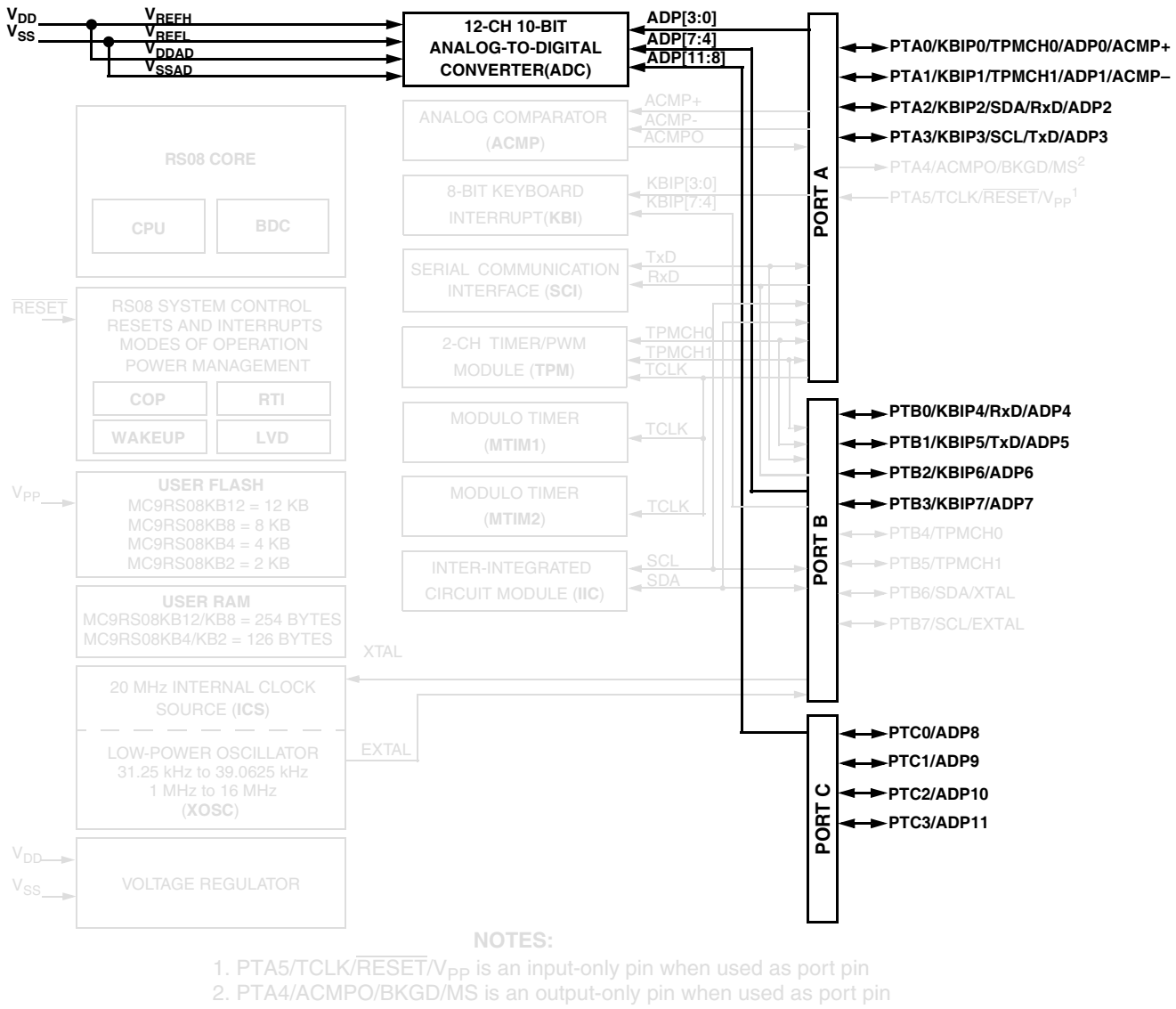


Figure 12-1. MC9RS08KB12 Series Block Diagram Highlighting ADC Block and Pins

12.1.1 Module Configurations

This section provides device-specific information for configuring the ADC on MC9RS08KB12 series devices.

12.1.1.1 Analog Supply and Voltage Reference Connections

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

12.1.1.2 Alternate Clock

The ADC module is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock, ALTCLK. The alternate clock for the MC9RS08KB12 devices is the external reference clock (ICSERCLK).

The selected clock source must run at a frequency such that the ADC conversion clock (ADCK) runs at a frequency within its specified range (f_{ADCK}) after being divided down from the ALTCLK input as determined by the ADIV bits.

ALTCLK is active while the MCU is in wait mode provided the conditions described above are met. This allows ALTCLK to be used as the conversion clock source for the ADC while the MCU is in wait mode. ALTCLK cannot be used as the ADC conversion clock source while the MCU is in stop mode.

12.1.1.3 Hardware Trigger

The ADC hardware trigger ADHWT is the output from the real-time interrupt (RTI) counter or the output of the modulo timer (MTIM1).

When the ADCHTS bit in the SOPT2 register is cleared, the ADC hardware trigger ADHWT is connected to the output from the RTI. ICSERCLK or a nominal 1 kHz clock source within the RTI block can clock the RTI counter.

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

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

When the ADCHTS bit in the SOPT2 register is set, the ADC hardware trigger ADHWT is connected to the output of MTIM1. The bus clock, fixed-frequency clock or an external clock source can clock the MTIM1 counter.

NOTE

To get quick hardware trigger, configure ADHWT to the output of MTIM1 or the output of RTI clocked by high frequency external clock source.

12.1.1.4 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.

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

where:

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

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

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

12.1.1.5 Channel Assignment

The ADC on MC9RS08KB12 series devices contains only two analog pin-enable registers, APCTL1 and APCTL2.

The ADC channel assignments for the MC9RS08KB12 series devices are shown in the table below. Reserved channels convert to an unknown value. Channels which are connected to an I/O pin have an associated pin control bit as shown.

Table 12-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control
00000	AD0	PTA0//ADP0	ADPC0
00001	AD1	PTA1/ADP1	ADPC1
00010	AD2	PTA2/ADP2	ADPC2
00011	AD3	PTA3/ADP3	ADPC3
00100	AD4	PTB0/ADP4	ADPC4
00101	AD5	PTB1/ADP5	ADPC5
00110	AD6	PTB2/ADP6	ADPC6
00111	AD7	PTB3/ADP7	ADPC7
01000	AD8	PTC0/ADP8	ADPC8
01001	AD9	PTC1/ADP9	ADPC9
01010	AD10	PTC2/ADP10	ADPC10
01011	AD11	PTC3/ADP11	ADPC11
01100	AD12	V_{REFL}	N/A
01101	AD13	V_{REFL}	N/A
01110	AD14	V_{REFL}	N/A
01111	AD15	V_{REFL}	N/A

ADCH	Channel	Input	Pin Control
10000	AD16	V_{REFL}	N/A
10001	AD17	V_{REFL}	N/A
10010	AD18	V_{REFL}	N/A
10011	AD19	V_{REFL}	N/A
10100	AD20	V_{REFL}	N/A
10101	AD21	V_{REFL}	N/A
10110	AD22	Reserved	N/A
10111	AD23	Reserved	N/A
11000	AD24	Reserved	N/A
11001	AD25	Reserved	N/A
11010	AD26	Temperature Sensor	N/A
11011	AD27	Internal Bandgap	N/A
11100	AD28	Reserved	N/A
11101	AD29	V_{REFH}	N/A
11110	AD30	V_{REFL}	N/A
11111	module disabled	None	N/A

12.1.1.6 Low-Power Mode Operation

The ADC can run in stop mode but requires LVDSE and LVDE in SPMSC1 to be set.

12.1.2 Features

Features of the ADC module include:

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

12.1.3 Block Diagram

Figure 12-2 provides a block diagram of the ADC module

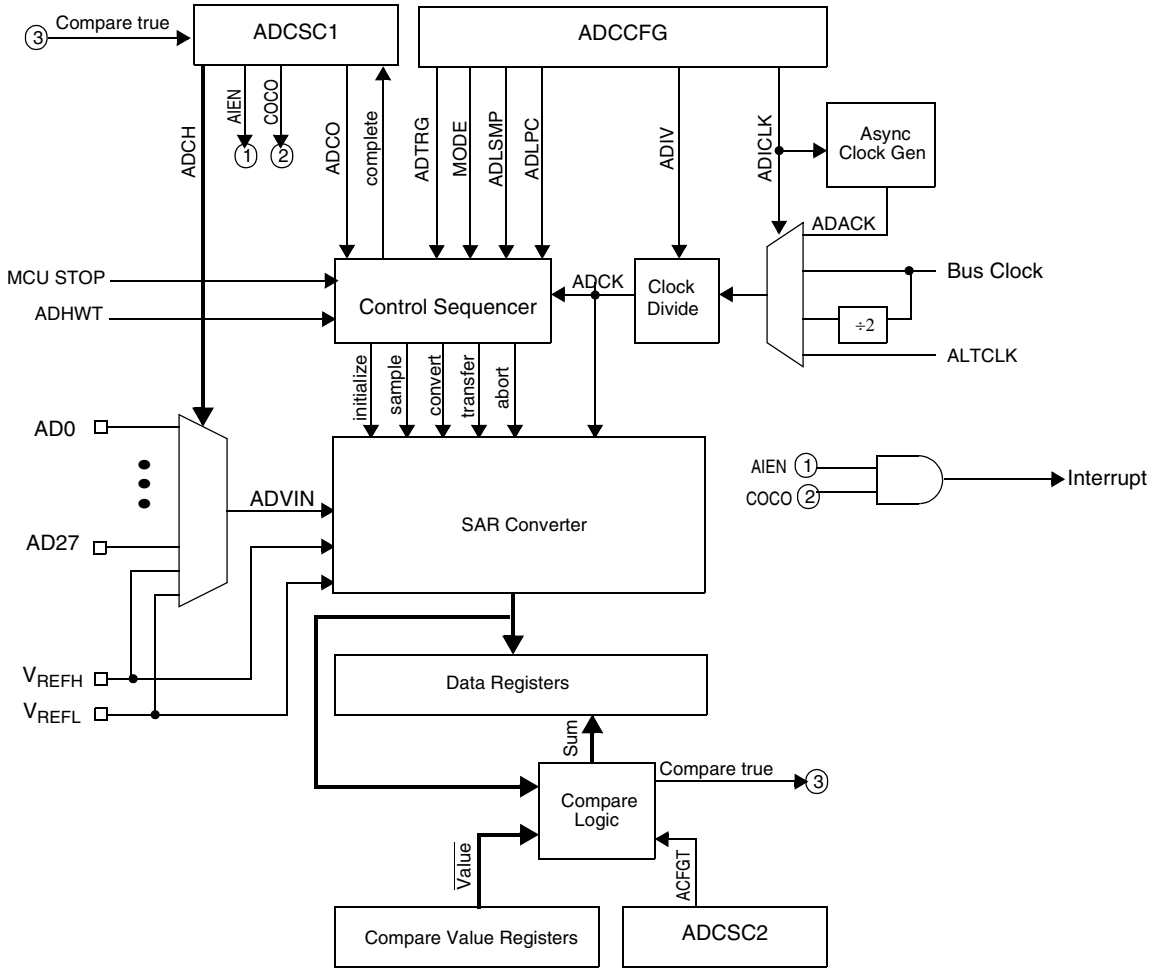


Figure 12-2. ADC Block Diagram

12.2 External Signal Description

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

Table 12-2. Signal Properties

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

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

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

12.3.1 Status and Control Register 1 (ADCSC1)

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

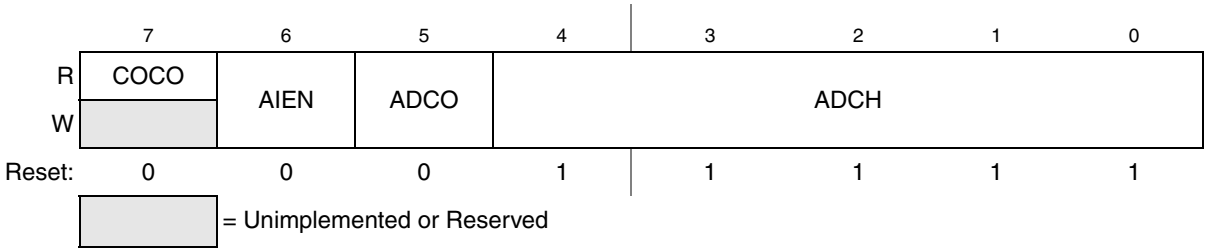


Figure 12-3. Status and Control Register (ADCSC1)

Table 12-3. ADCSC1 Register Field Descriptions

Field	Description
7 COCO	**Conversion Complete Flag** — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADCSC1 is written or whenever ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	**Interrupt Enable** — AIEN is used to enable conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	**Continuous Conversion Enable** — ADCO is used to enable continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	**Input Channel Select** — The ADCH bits form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Figure 12-4. The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Figure 12-4. Input Channel Select

ADCH	Input Select
00000	AD0
00001	AD1
00010	AD2
00011	AD3
00100	AD4
00101	AD5
00110	AD6
00111	AD7

ADCH	Input Select
10000	AD16
10001	AD17
10010	AD18
10011	AD19
10100	AD20
10101	AD21
10110	AD22
10111	AD23

Figure 12-4. Input Channel Select (continued)

ADCH	Input Select
01000	AD8
01001	AD9
01010	AD10
01011	AD11
01100	AD12
01101	AD13
01110	AD14
01111	AD15

ADCH	Input Select
11000	AD24
11001	AD25
11010	AD26
11011	AD27
11100	Reserved
11101	V_{REFH}
11110	V_{REFL}
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.

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R[1]	R[1]
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

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

Table 12-4. ADCSC2 Register Field Descriptions (continued)

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

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.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	ADR9	ADR8
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

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

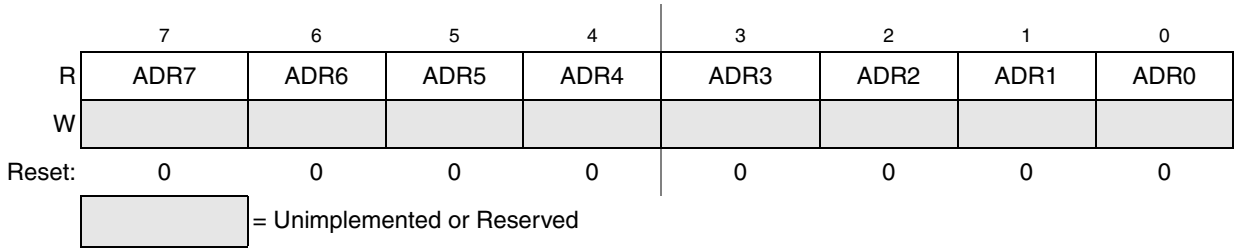


Figure 12-7. Data Result Low Register (ADCRL)

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.

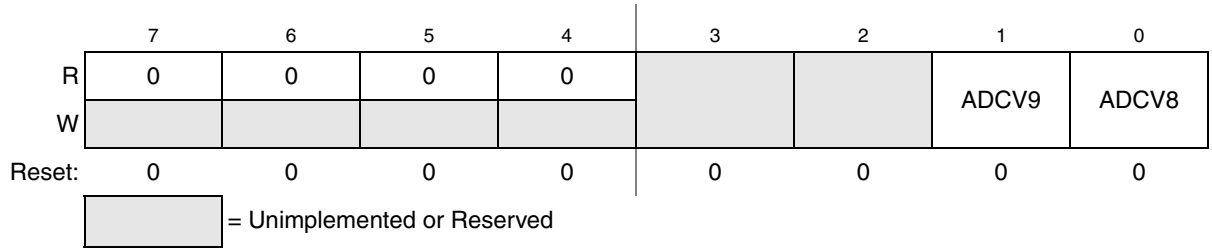


Figure 12-8. Compare Value High Register (ADCCVH)

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 either 10-bit or 8-bit mode.

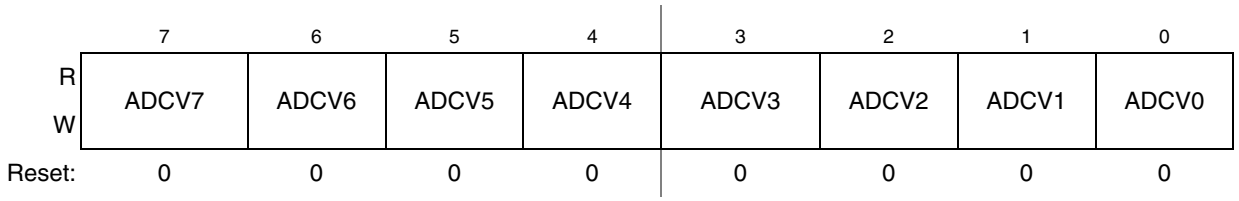


Figure 12-9. Compare Value Low Register(ADCCVL)

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.

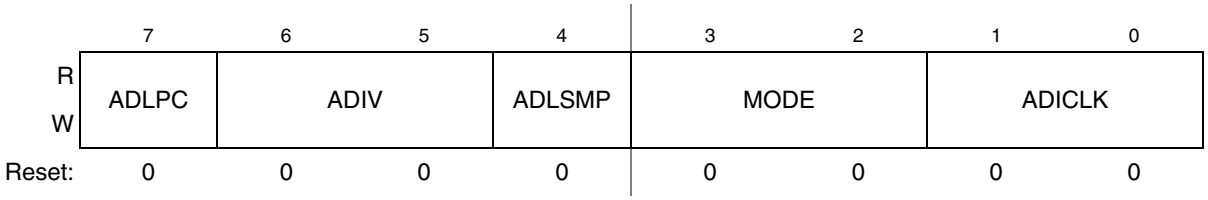


Figure 12-10. Configuration Register (ADCCFG)

Table 12-5. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	**Low Power Configuration** — ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: {FC31}The power is reduced at the expense of maximum clock speed.
6:5 ADIV	**Clock Divide Select** — ADIV select the divide ratio used by the ADC to generate the internal clock ADCK. Table 12-6 shows the available clock configurations.
4 ADLSMP	**Long Sample Time Configuration** — ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	**Conversion Mode Selection** — MODE bits are used to select between 10- or 8-bit operation. See Table 12-7.
1:0 ADICLK	**Input Clock Select** — ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 12-8.

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

	7	6	5	4	3	2	1	0
R W	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
Reset:	0	0	0	0	0	0	0	0

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

Table 12-9. APCTL1 Register Field Descriptions

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

Table 12-9. APCTL1 Register Field Descriptions (continued)

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

12.3.9 Pin Control 2 Register (APCTL2)

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

	7	6	5	4	3	2	1	0
R W	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
Reset:	0	0	0	0	0	0	0	0

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

Table 12-10. APCTL2 Register Field Descriptions

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

Table 12-10. APCTL2 Register Field Descriptions (continued)

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

12.3.10 Pin Control 3 Register (APCTL3)

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

	7	6	5	4	3	2	1	0
R W	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
Reset:	0	0	0	0	0	0	0	0

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

Table 12-11. APCTL3 Register Field Descriptions

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

Table 12-11. APCTL3 Register Field Descriptions (continued)

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

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

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

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

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

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

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

12.4.2 Input Select and Pin Control

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

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

12.4.3 Hardware Trigger

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

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

12.4.4 Conversion Control

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

12.4.4.1 Initiating Conversions

A conversion is initiated:

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

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

12.4.4.2 Completing Conversions

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

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

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

12.4.4.3 Aborting Conversions

Any conversion in progress will be aborted when:

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

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

12.4.4.4 Power Control

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

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

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 is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The

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

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

The maximum total conversion time for different conditions is summarized in Table 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 µs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	0	5 µs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 µs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	1	5 µs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

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

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

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

NOTE

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

12.4.5 Automatic Compare Function

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

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

NOTE

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

12.4.6 MCU Wait Mode Operation

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

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

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

12.4.7 MCU Stop3 Mode Operation

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

12.4.7.1 Stop3 Mode With ADACK Disabled

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

12.4.7.2 Stop3 Mode With ADACK Enabled

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

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

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

NOTE

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in Section 12.4.4.2, "Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

12.4.8 MCU Stop1 and Stop2 Mode Operation

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

12.5 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 12-6, Table 12-7, and Table 12-8 for information used in this example.

NOTE

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

12.5.1 ADC Module Initialization Example

12.5.1.1 Initialization Sequence

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

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

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

12.5.1.2 Pseudo — Code Example

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

ADCCFG = 0x98 (%10011000)

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

ADCSC2 = 0x00 (%00000000)

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

ADCSC1 = 0x41 (%01000001)

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

ADCRH/L = 0xxx

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

ADCCVH/L = 0xxx

Holds compare value when compare function enabled

APCTL1=0x02

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

APCTL2=0x00

All other AD pins remain general purpose I/O pins

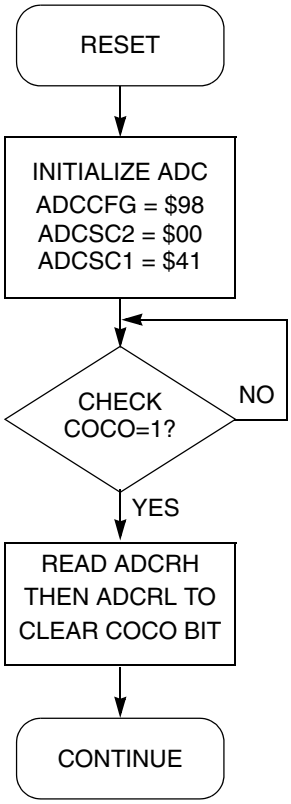


Figure 12-14. Initialization Flowchart for Example

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 they should be used for best results.

12.6.1.1 Analog Supply Pins

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

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

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

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 may be shared on the same pin as V_{DDAD} on some devices. The low reference is V_{REFL}, which may be shared on the same pin as V_{SSAD} on some devices.

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

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

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 should be done to achieve lowest operating current.

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

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

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

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 7kΩ 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 kept below 5 kΩ.

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

12.6.2.2 Pin Leakage Error

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

12.6.2.3 Noise-Induced Errors

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

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

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

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

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

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:

$$1_{LSB} = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 12-2}$$

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

12.6.2.5 Linearity Errors

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

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

12.6.2.6 Code Jitter, Non-Monotonicity and Missing Codes

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

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

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 12.6.2.3 will reduce this error.

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

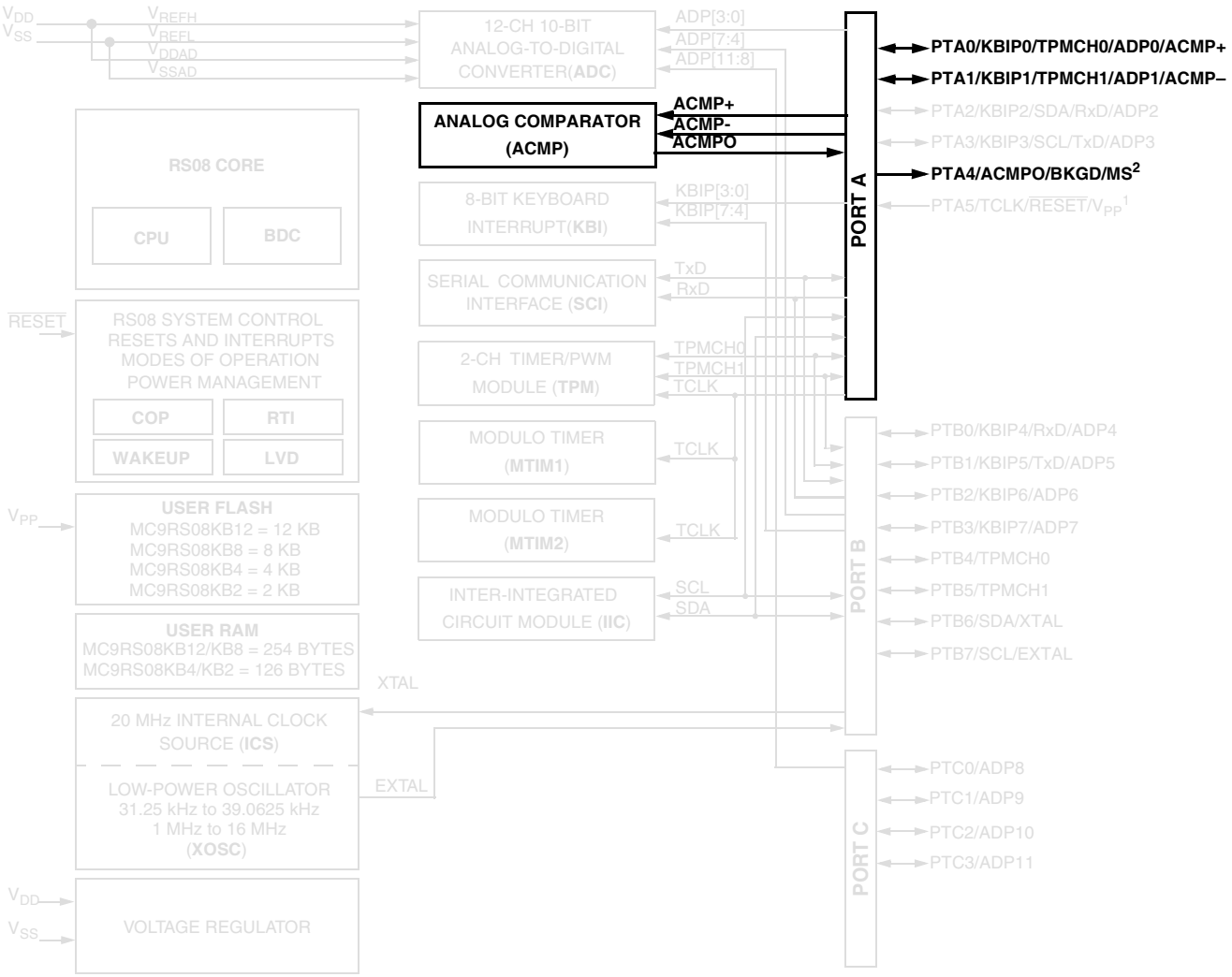
In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and to have no missing codes.

Chapter 13
Analog Comparator (RS08ACMPV1)

13.1 Introduction

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

Figure 13-1 shows the MC9RS08KB12 series block diagram with the ACMP block and pins highlighted.



NOTES:
1. PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} is an input-only pin when used as port pin
2. PTA4/ACMPO/BKGD/MS is an output-only pin when used as port pin

Figure 13-1. MC9RS08KB12 Series Block Diagram Highlighting ACMP Block and Pins

13.1.1 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

13.1.2 Modes of Operation

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

13.1.2.1 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 should be disabled by software if not required as an interrupt source during wait mode.

13.1.2.2 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.

13.1.2.3 Operation in Active Background Mode

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

13.1.3 Block Diagram

The block diagram for the analog comparator module is shown in Figure 13-2.

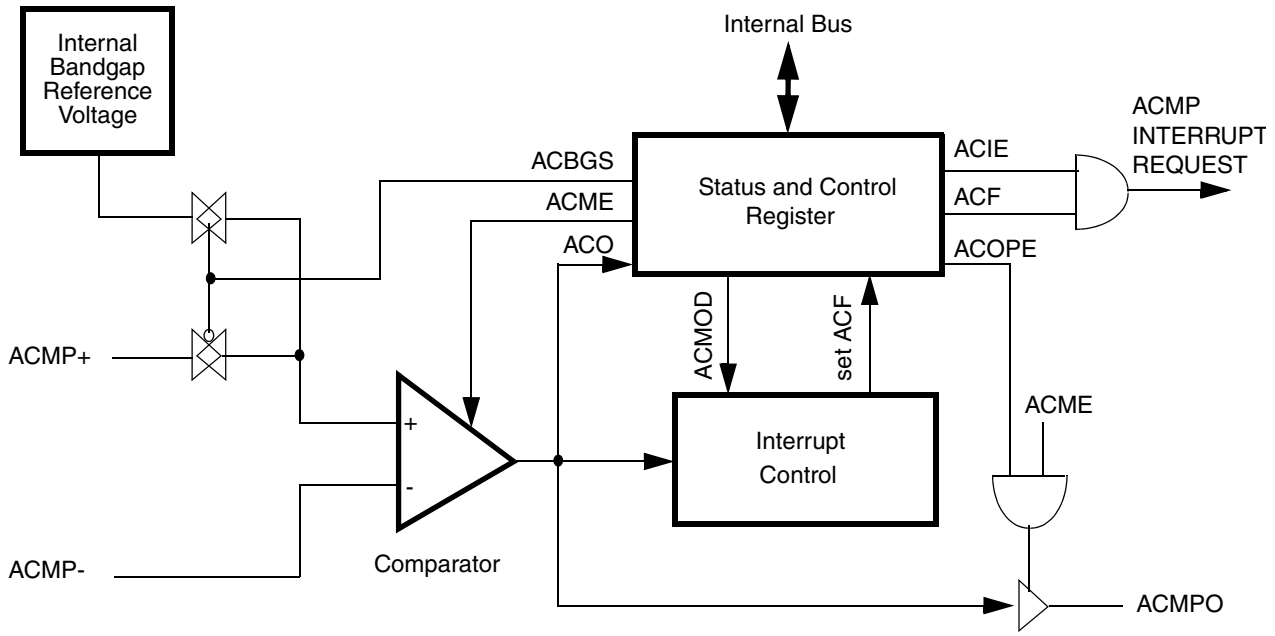


Figure 13-2. Analog Comparator (ACMP) Block Diagram

13.2 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 13-2, 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 13-2, the ACMPO pin can be enabled to drive an external pin.

The signal properties of ACMP are shown in Table 13-1.

Table 13-1. 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

13.3 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.

13.3.1 ACMP Status and Control Register (ACMPSC)

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

	7	6	5	4	3	2	1	0
R	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD	
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-3. ACMP Status and Control Register (ACMPSC)

Table 13-2. ACMPSC Field Descriptions

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

13.4 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 impedence analog pins which are sensitive to noise. Noisy V_{DD} and/or pin toggling adjacent to the analog inputs may cause the comparator offset/hysteresis performance to exceed the specified values. Maximum source impedence is restricted to the value specified in Table 13-2. To achieve maximum performance device is recommended to enter WAIT/STOP mode for ACMP measurement and adjacent pin toggling must be avoided.

Chapter 14
Inter-Integrated Circuit (S08IICV2)

14.1 Introduction

The inter-integrated circuit (IIC) provides communication among several devices. The interface can operate up to 100 kbps with maximum bus loading and timing. The device can operate at higher baud rates, up to a maximum of clock/20, with reduced bus loading. A maximum bus capacitance of 400 pF limits the communication length and the number of connected devices.

NOTE

The SDA and SCL must not be driven above V_{DD}. These pins are pseudo open-drain containing a protection diode to V_{DD}.

NOTE

MC9RS08KB12 series MCUs DO NOT support stop1 or stop2, neglect those information in this chapter. The stop mode in the other chapters is the stop3 in this chapter.

14.1.1 Module Configuration

The IIC module pins, SDA and SCL can be repositioned under software control using IICPS in SOPT1 (Table 14-1). IICPS in SOPT1 selects the general-purpose I/O ports associated with IIC operation.

Table 14-1. IIC Position Options

IICPS in SOPT	Port Pin for SDA	Port Pin for SCL
0 (default)	PTA2	PTA3
1	PTB6	PTB7

Figure 14-1 shows the MC9RS08KB12 series block diagram with the IIC block and pins highlighted.

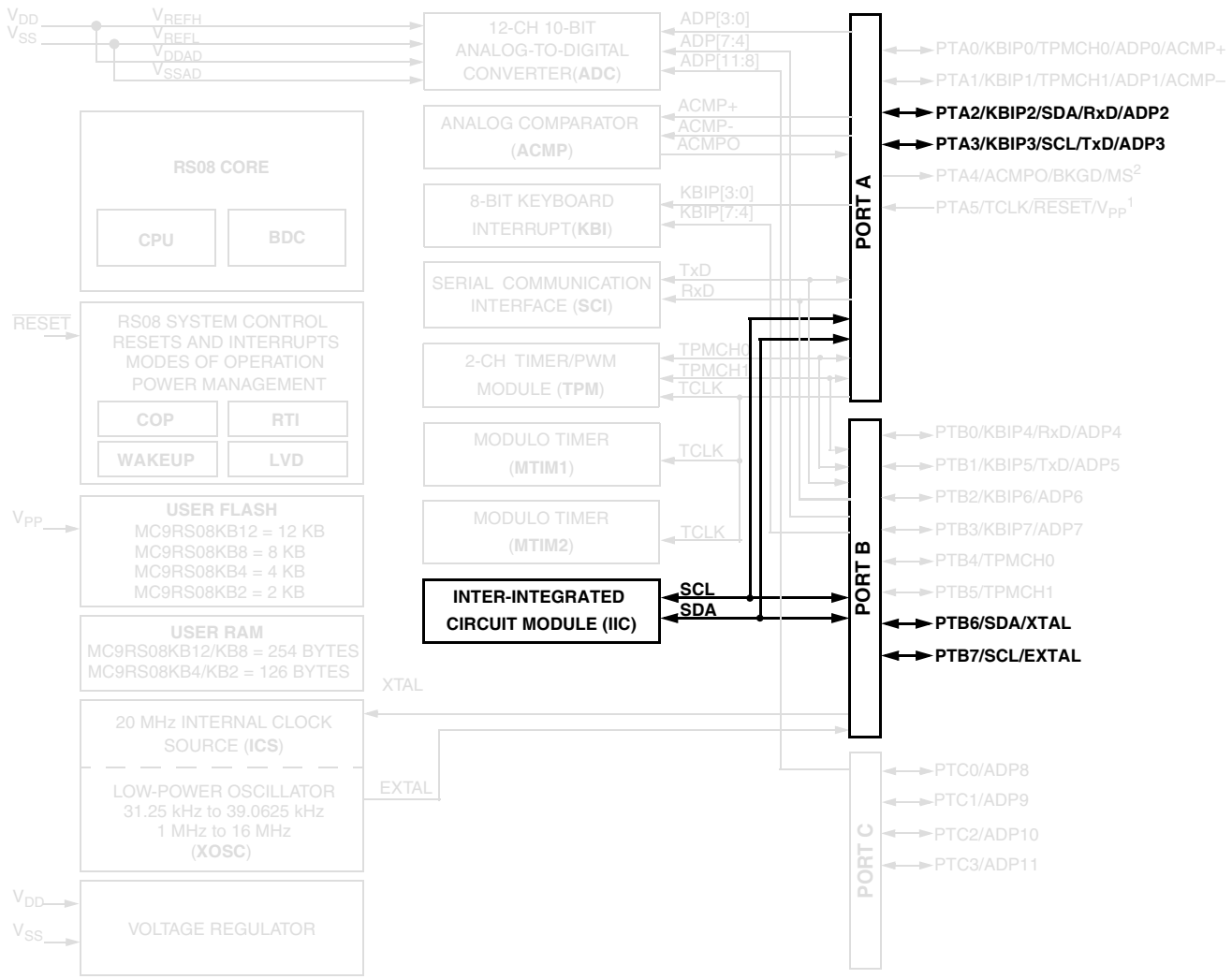


Figure 14-1. MC9RS08KB12 Series Block Diagram Highlighting the IIC Block and Pins

14.1.2 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- General call recognition
- 10-bit address extension

14.1.3 Modes of Operation

A brief description of the IIC in the various MCU modes is given here.

- **Run mode** — This is the basic mode of operation. To conserve power in this mode, disable the module.
- **Wait mode** — The module continues to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- **Stop mode** — The IIC is inactive in stop3 mode for reduced power consumption. The stop instruction does not affect IIC register states. Stop2 resets the register contents.

14.1.4 Block Diagram

Figure 14-2 is a block diagram of the IIC.

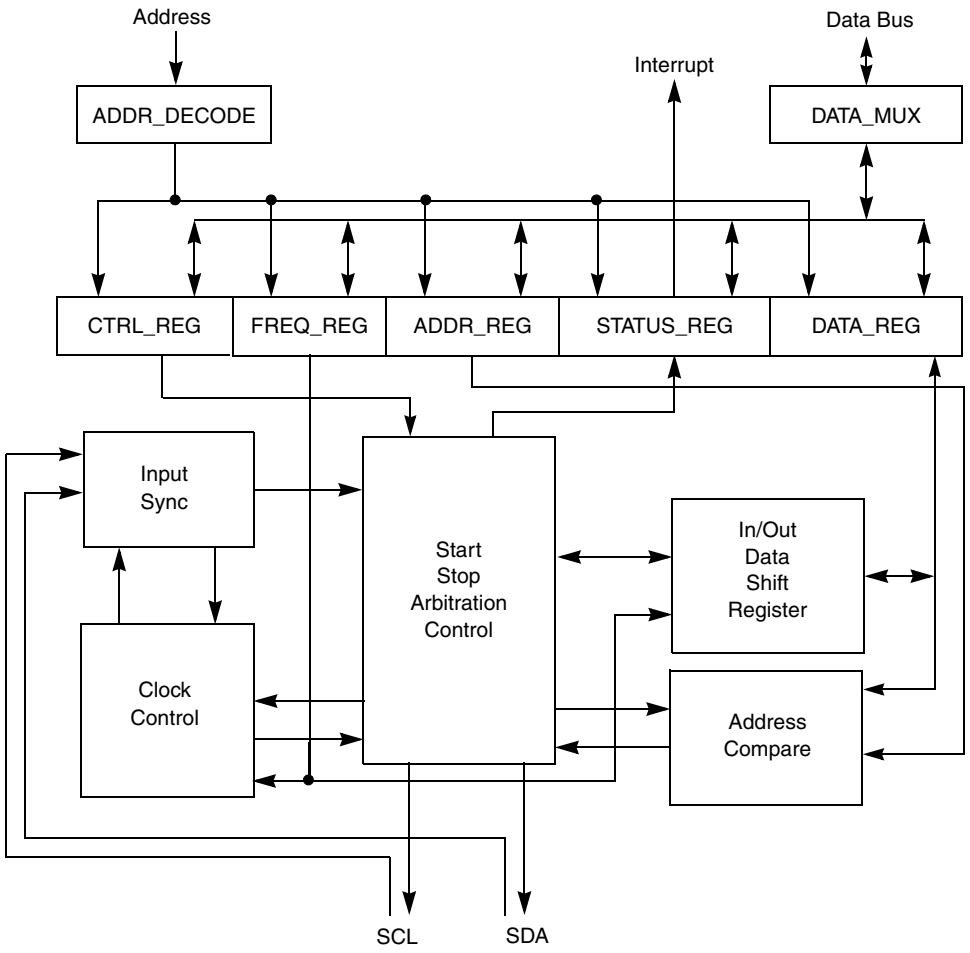


Figure 14-2. IIC Functional Block Diagram

14.2 External Signal Description

This section describes each user-accessible pin signal.

14.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

14.2.2 SDA — Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

14.3 Register Definition

This section consists of the IIC register descriptions in address order.

Refer to the direct-page register summary in the memory chapter of this document for the absolute address assignments for all IIC registers. This section refers to registers and control bits only by their names. A

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

14.3.1 IIC Address Register (IICA)

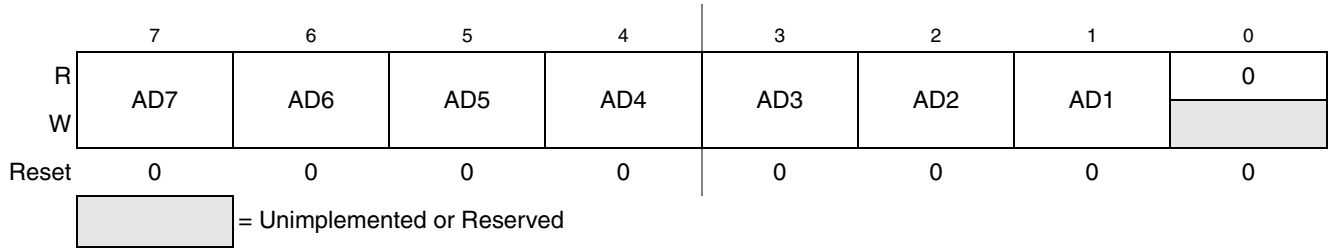


Figure 14-3. IIC Address Register (IICA)

Table 14-2. IICA Field Descriptions

Field	Description
7–1 AD[7:1]	**Slave Address.** The AD[7:1] field contains the slave address to be used by the IIC module. This field is used on the 7-bit address scheme and the lower seven bits of the 10-bit address scheme.

14.3.2 IIC Frequency Divider Register (IICF)

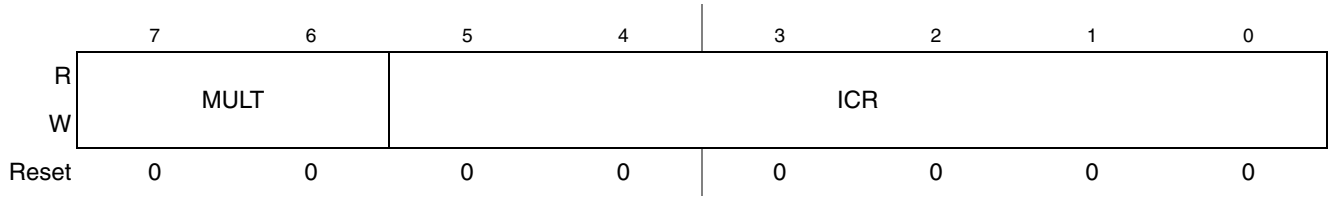


Figure 14-4. IIC Frequency Divider Register (IICF)

Table 14-3. IICF Field Descriptions

Field	Description
7–6 MULT	**IIC Multiplier Factor**. The MULT bits define the multiplier factor, mul. This factor, along with the SCL divider, generates the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below. 00 mul = 01 01 mul = 02 10 mul = 04 11 Reserved
5–0 ICR	**IIC Clock Rate**. The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits determine the IIC baud rate, the SDA hold time, the SCL Start hold time, and the SCL Stop hold time. Table 14-5 provides the SCL divider and hold values for corresponding values of the ICR. The SCL divider multiplied by multiplier factor mul generates IIC baud rate. $\text{IIC baud rate} = \frac{\text{bus speed (Hz)}}{\text{mul} \times \text{SCLdivider}}$ ***Eqn. 14-1*** SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data). SDA hold time = bus period (s) × mul × SDA hold value ***Eqn. 14-2*** SCL start hold time is the delay from the falling edge of SDA (IIC data) while SCL is high (Start condition) to the falling edge of SCL (IIC clock). SCL Start hold time = bus period (s) × mul × SCL Start hold value ***Eqn. 14-3*** SCL stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA SDA (IIC data) while SCL is high (Stop condition). SCL Stop hold time = bus period (s) × mul × SCL Stop hold value ***Eqn. 14-4***

For example, if the bus speed is 8 MHz, the table below shows the possible hold time values with different ICR and MULT selections to achieve an IIC baud rate of 100kbps.

Table 14-4. Hold Time Values for 8 MHz Bus Speed

MULT	ICR	Hold Times (μs)		
		SDA	SCL Start	SCL Stop
0x2	0x00	3.500	3.000	5.500
0x1	0x07	2.500	4.000	5.250
0x1	0x0B	2.250	4.000	5.250
0x0	0x14	2.125	4.250	5.125
0x0	0x18	1.125	4.750	5.125

Table 14-5. IIC Divider and Hold Values

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SDA Hold (Stop) Value
00	20	7	6	11
01	22	7	7	12
02	24	8	8	13
03	26	8	9	14
04	28	9	10	15
05	30	9	11	16
06	34	10	13	18
07	40	10	16	21
08	28	7	10	15
09	32	7	12	17
0A	36	9	14	19
0B	40	9	16	21
0C	44	11	18	23
0D	48	11	20	25
0E	56	13	24	29
0F	68	13	30	35
10	48	9	18	25
11	56	9	22	29
12	64	13	26	33
13	72	13	30	37
14	80	17	34	41
15	88	17	38	45
16	104	21	46	53
17	128	21	58	65
18	80	9	38	41
19	96	9	46	49
1A	112	17	54	57
1B	128	17	62	65
1C	144	25	70	73
1D	160	25	78	81
1E	192	33	94	97
1F	240	33	118	121

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
20	160	17	78	81
21	192	17	94	97
22	224	33	110	113
23	256	33	126	129
24	288	49	142	145
25	320	49	158	161
26	384	65	190	193
27	480	65	238	241
28	320	33	158	161
29	384	33	190	193
2A	448	65	222	225
2B	512	65	254	257
2C	576	97	286	289
2D	640	97	318	321
2E	768	129	382	385
2F	960	129	478	481
30	640	65	318	321
31	768	65	382	385
32	896	129	446	449
33	1024	129	510	513
34	1152	193	574	577
35	1280	193	638	641
36	1536	257	766	769
37	1920	257	958	961
38	1280	129	638	641
39	1536	129	766	769
3A	1792	257	894	897
3B	2048	257	1022	1025
3C	2304	385	1150	1153
3D	2560	385	1278	1281
3E	3072	513	1534	1537
3F	3840	513	1918	1921

14.3.3 IIC Control Register (IICC1)

	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-5. IIC Control Register (IICC1)

Table 14-6. IICC1 Field Descriptions

Field	Description
7 IICEN	**IIC Enable.** The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled 1 IIC is enabled
6 IICIE	**IIC Interrupt Enable.** The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled 1 IIC interrupt request enabled
5 MST	**Master Mode Select.** The MST bit changes from a 0 to a 1 when a start signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a stop signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	**Transmit Mode Select.** The TX bit selects the direction of master and slave transfers. In master mode, this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. When addressed as a slave, this bit should be set by software according to the SRW bit in the status register. 0 Receive 1 Transmit
3 TXAK	**Transmit Acknowledge Enable.** This bit specifies the value driven onto the SDA during data acknowledge cycles for master and slave receivers. 0 An acknowledge signal is sent out to the bus after receiving one data byte 1 No acknowledge signal response is sent
2 RSTA	**Repeat start.** Writing a 1 to this bit generates a repeated start condition provided it is the current master. This bit is always read as cleared. Attempting a repeat at the wrong time results in loss of arbitration.

14.3.4 IIC Status Register (IICS)

	7	6	5	4	3	2	1	0
R	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
W								
Reset	1	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-6. IIC Status Register (IICS)

Table 14-7. IICS Field Descriptions

Field	Description
7 TCF	**Transfer Complete Flag.** This bit is set on the completion of a byte transfer. This bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IICD register in receive mode or writing to the IICD in transmit mode. 0 Transfer in progress 1 Transfer complete
6 IAAS	**Addressed as a Slave.** The IAAS bit is set when the calling address matches the programmed slave address or when the GCAEN bit is set and a general call is received. Writing the IICC register clears this bit. 0 Not addressed 1 Addressed as a slave
5 BUSY	**Bus Busy.** The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a start signal is detected and cleared when a stop signal is detected. 0 Bus is idle 1 Bus is busy
4 ARBL	**Arbitration Lost.** This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software by writing a 1 to it. 0 Standard bus operation 1 Loss of arbitration
2 SRW	**Slave Read/Write.** When addressed as a slave, the SRW bit indicates the value of the R/W command bit of the calling address sent to the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IICIF	**IIC Interrupt Flag.** The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit: • One byte transfer completes • Match of slave address to calling address • Arbitration lost 0 No interrupt pending 1 Interrupt pending
0 RXAK	**Receive Acknowledge**. When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected. 0 Acknowledge received 1 No acknowledge received

14.3.5 IIC Data I/O Register (IICD)

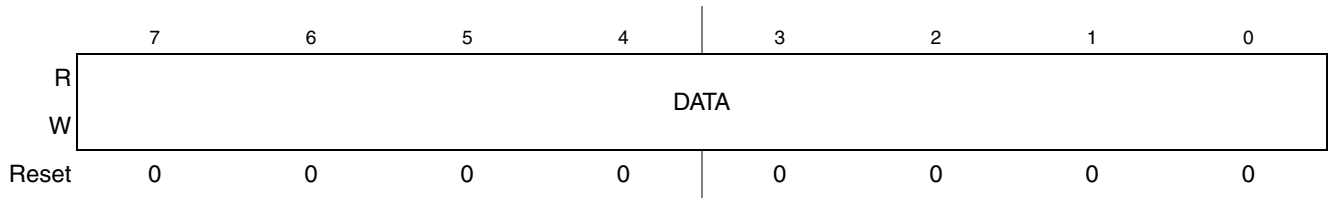


Figure 14-7. IIC Data I/O Register (IICD)

Table 14-8. IICD Field Descriptions

Field	Description
7–0 DATA	**Data** — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data.

NOTE

When transitioning out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

The TX bit in IICC must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, reading the IICD does not initiate the receive.

Reading the IICD returns the last byte received while the IIC is configured in master receive or slave receive modes. The IICD does not reflect every byte transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

14.3.6 IIC Control Register 2 (IICC2)

	7	6	5	4	3	2	1	0
R	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-8. IIC Control Register (IICC2)

Table 14-9. IICC2 Field Descriptions

Field	Description
7 GCAEN	**General Call Address Enable.** The GCAEN bit enables or disables general call address. 0 General call address is disabled 1 General call address is enabled
6 ADEXT	**Address Extension.** The ADEXT bit controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
2–0 AD[10:8]	Slave Address. The AD[10:8] field contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set.

14.4 Functional Description

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

14.4.1 IIC Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts:

- Start signal
- Slave address transmission
- Data transfer
- Stop signal

The stop signal should not be confused with the CPU stop instruction. The IIC bus system communication is described briefly in the following sections and illustrated in Figure 14-9.

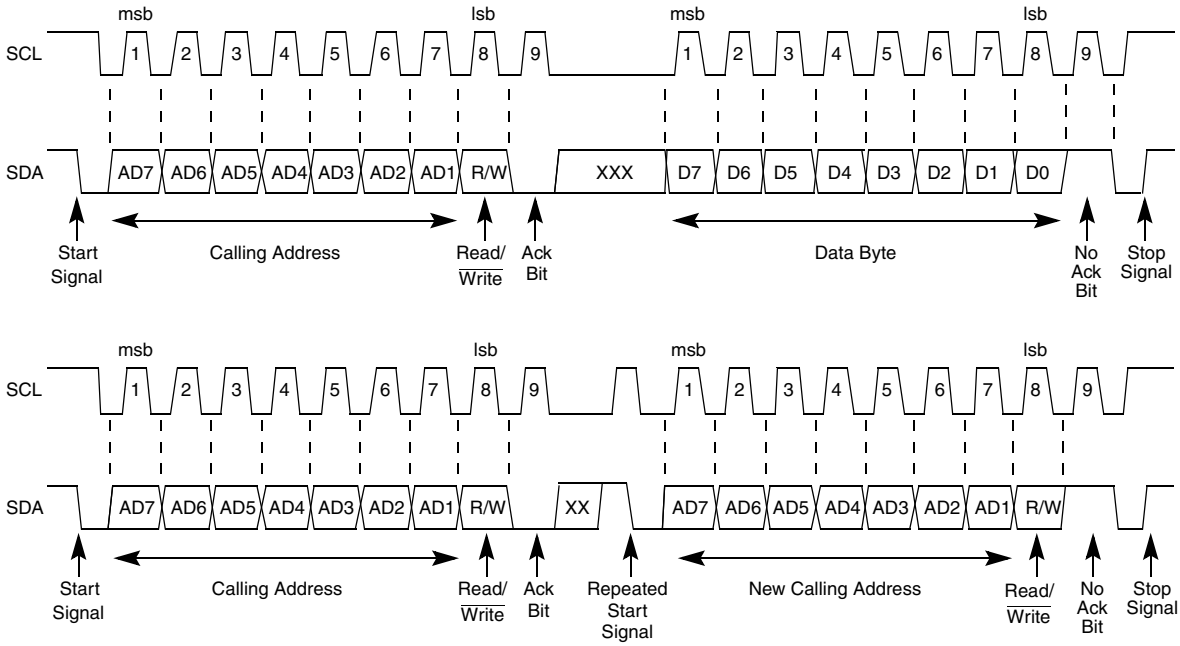


Figure 14-9. IIC Bus Transmission Signals

14.4.1.1 Start Signal

When the bus is free, no master device is engaging the bus (SCL and SDA lines are at logical high), a master may initiate communication by sending a start signal. As shown in Figure 14-9, a start signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

14.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the start signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 14-9).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC reverts to slave mode and operates correctly even if it is being addressed by another master.

14.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{W}$ bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 14-9. There is one clock pulse on SCL for each data bit, the msb being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a stop signal.
- Commences a new calling by generating a repeated start signal.

14.4.1.4 Stop Signal

The master can terminate the communication by generating a stop signal to free the bus. However, the master may generate a start signal followed by a calling command without generating a stop signal first. This is called repeated start. A stop signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 14-9).

The master can generate a stop even if the slave has generated an acknowledge at which point the slave must release the bus.

14.4.1.5 Repeated Start Signal

As shown in Figure 14-9, a repeated start signal is a start signal generated without first generating a stop signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

14.4.1.6 Arbitration Procedure

The IIC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a stop condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

14.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 14-10). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

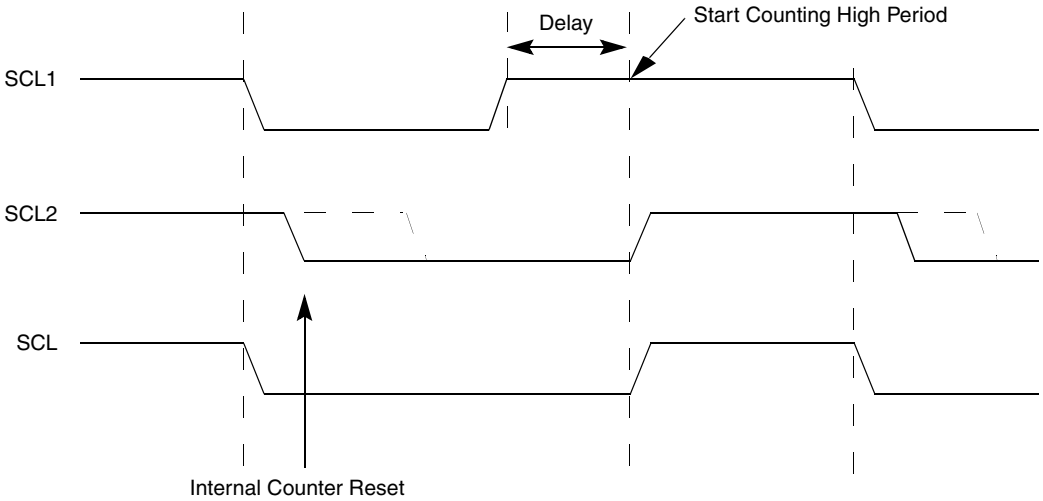


Figure 14-10. IIC Clock Synchronization

14.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such a case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

14.4.1.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

14.4.2 10-bit Address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

14.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 14-10). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P

Table 14-10. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

14.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit (see Table 14-11). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

After a repeated start condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices) or the 11110XX slave address (for 7-bit devices) does not match.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Sr	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 1	A3	Data	A	...	Data	A	P

Table 14-11. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

14.4.3 General Call Address

General calls can be requested in 7-bit address or 10-bit address. If the GCAEN bit is set, the IIC matches the general call address as well as its own slave address. When the IIC responds to a general call, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the IICD register after the first byte transfer to determine whether the address matches is its own slave address or a general call. If the value is 00, the match is a general call. If the GCAEN bit is clear, the IIC ignores any data supplied from a general call address by not issuing an acknowledgement.

14.5 Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

14.6 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 14-12 occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. You can determine the interrupt type by reading the status register.

Table 14-12. Interrupt Summary

Interrupt Source	Status	Flag	Local Enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration Lost	ARBL	IICIF	IICIE

14.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the ninth clock to indicate the completion of byte transfer.

14.6.2 Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the status register is set. The CPU is interrupted, provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

14.6.3 Arbitration Lost Interrupt

The IIC is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A start cycle is attempted when the bus is busy.
- A repeated start cycle is requested in slave mode.
- A stop condition is detected when the master did not request it.

This bit must be cleared by software writing a 1 to it.

14.7 Initialization/Application Information

Module Initialization (Slave)

1. Write: IICC2
   — to enable or disable general call
   — to select 10-bit or 7-bit addressing mode
2. Write: IICA
   — to set the slave address
3. Write: IICC1
   — to enable IIC and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in Figure 14-12

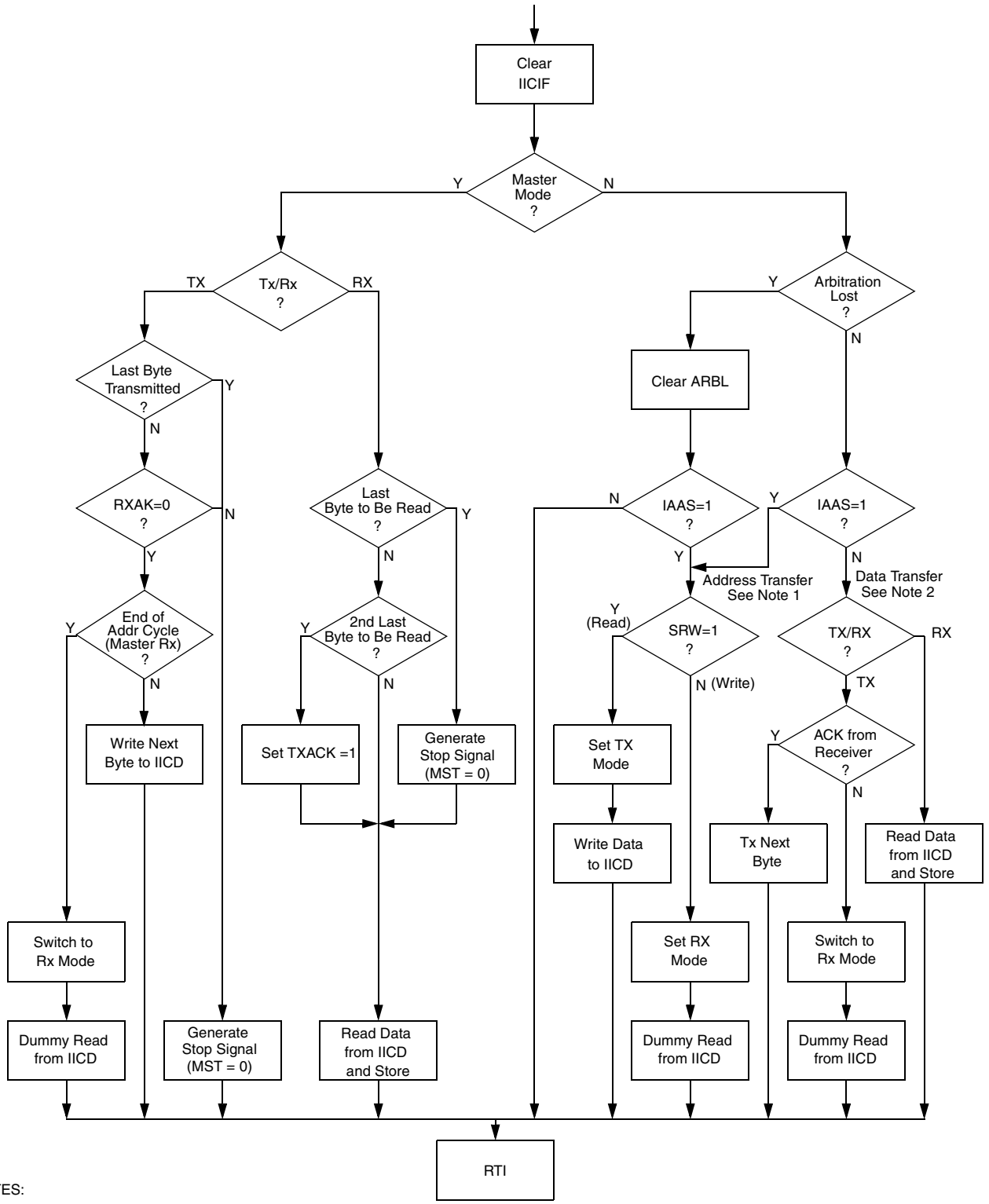
Module Initialization (Master)

1. Write: IICF
   — to set the IIC baud rate (example provided in this chapter)
2. Write: IICC1
   — to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in Figure 14-12
5. Write: IICC1
   — to enable TX

Register Model

IICA	AD[7:1]							0
	When addressed as a slave (in slave mode), the module responds to this address							
IICF	MULT		ICR					
	Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER))							
IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
	Module configuration							
IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
	Module status flags							
IICD	DATA							
	Data register; Write to transmit IIC data read to read IIC data							
IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
	Address configuration							

Figure 14-11. IIC Module Quick Start



NOTES:

1. If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.
2. When 10-bit addressing is used to address a slave, the slave sees an interrupt following the first byte of the extended address. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as a valid data transfer.

Figure 14-12. Typical IIC Interrupt Routine

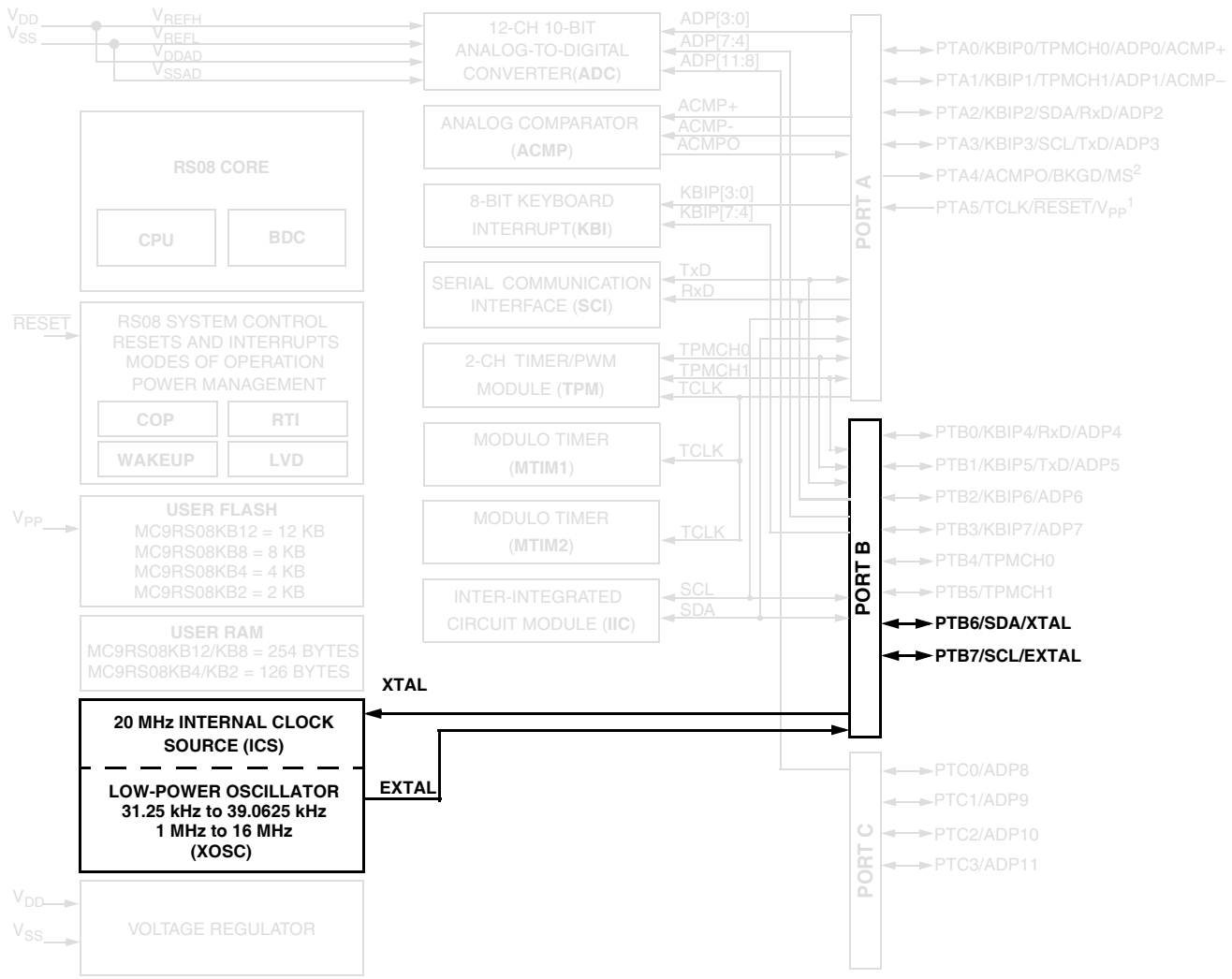
Chapter 15
Internal Clock Source (S08ICSV1)

15.1 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 controllable by an internal reference clock or an external reference clock. The module can provide FLL clock, internal reference clock, or external 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 a lower final output clock frequency to be derived. ICSOUT is two times the bus frequency.

Figure 15-1 shows the MC9RS08KB12 series block diagram with the ICS block and pins highlighted.



NOTES:
1. PTA5/TCLK/$\overline{\text{RESET}}$/V_{PP} is an input-only pin when used as port pin
2. PTA4/ACMPO/BKGD/MS is an output-only pin when used as port pin

Figure 15-1. MC9RS08KB12 Series Block Diagram Highlighting ICS Block and Pins

15.1.1 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
- External reference clock up to 5MHz can be used to control the FLL
  - 3 bit select for reference divider is provided
- Internal reference clock has 9 trim bits available
- Internal or external reference clock can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  - 2 bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8
    - BDC clock is provided as a constant divide by 2 of the DCO output
- Control signals for a low power oscillator as the external reference clock are provided
  - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL engaged internal mode is automatically selected out of reset

15.1.2 Modes of Operation

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

15.1.2.1 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. The BDC clock is supplied from the FLL.

15.1.2.2 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. The BDC clock is supplied from the FLL.

15.1.2.3 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. The BDC clock is supplied from the FLL.

15.1.2.4 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. The BDC clock is not available.

15.1.2.5 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. The BDC clock is supplied from the FLL.

15.1.2.6 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. The BDC clock is not available.

15.1.2.7 Stop (STOP)

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

15.1.3 Block Diagram

This section contains the ICS block diagram.

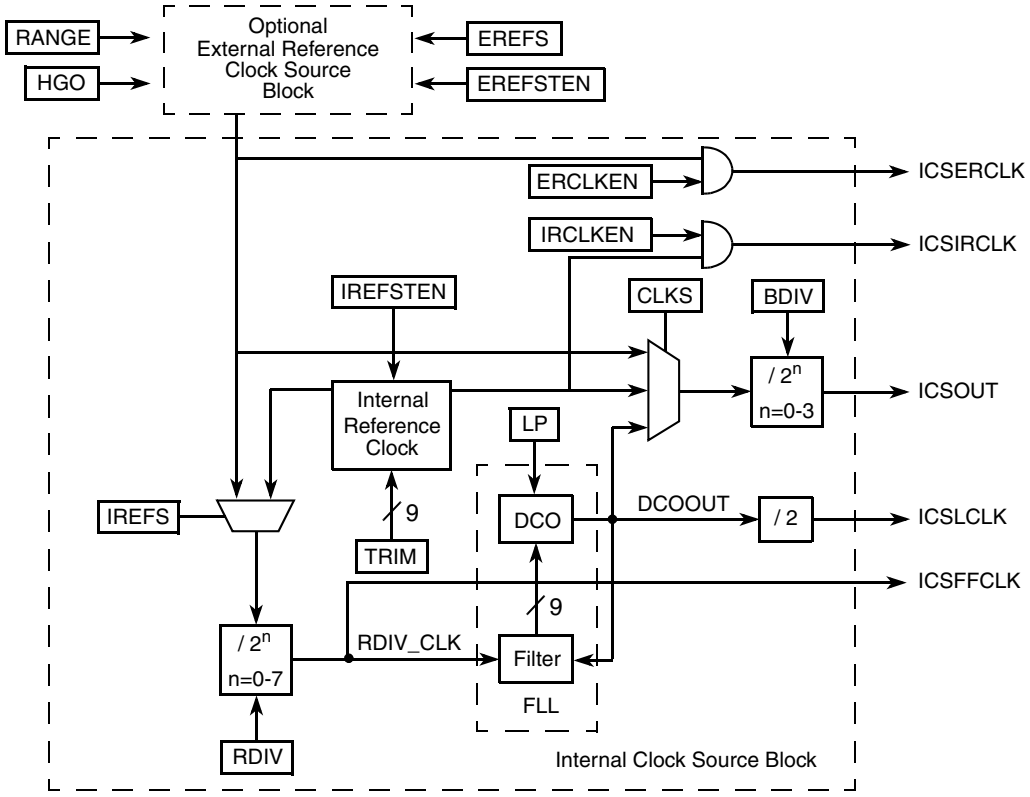


Figure 15-2. Internal Clock Source (ICS) Block Diagram

15.2 External Signal Description

No ICS signal connects off chip.

15.3 Register Definition

15.3.1 ICS Control Register 1 (ICSC1)

	7	6	5	4	3	2	1	0
R W	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
Reset:	0	0	0	0	0	1	0	0

Figure 15-3. ICS Control Register 1 (ICSC1)

Table 15-1. ICS Control Register 1 Field Descriptions

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

15.3.2 ICS Control Register 2 (ICSC2)

	7	6	5	4	3	2	1	0
R W	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
Reset:	0	1	0	0	0	0	0	0

Figure 15-4. ICS Control Register 2 (ICSC2)

Table 15-2. ICS Control Register 2 Field Descriptions

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

15.3.3 ICS Trim Register (ICSTRM)

	7	6	5	4	3	2	1	0
R W	TRIM							
POR:	1	0	0	0	0	0	0	0
Reset:	U	U	U	U	U	U	U	U

Figure 15-5. ICS Trim Register (ICSTRM)

Table 15-3. ICS Trim Register Field Descriptions

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

15.3.4 ICS Status and Control (ICSSC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	CLKST		OSCINIT	FTRIM
W								
POR:	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	U

Figure 15-6. ICS Status and Control Register (ICSSC)

Table 15-4. ICS Status and Control Register Field Descriptions

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

15.4 Functional Description

15.4.1 Operational Modes

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

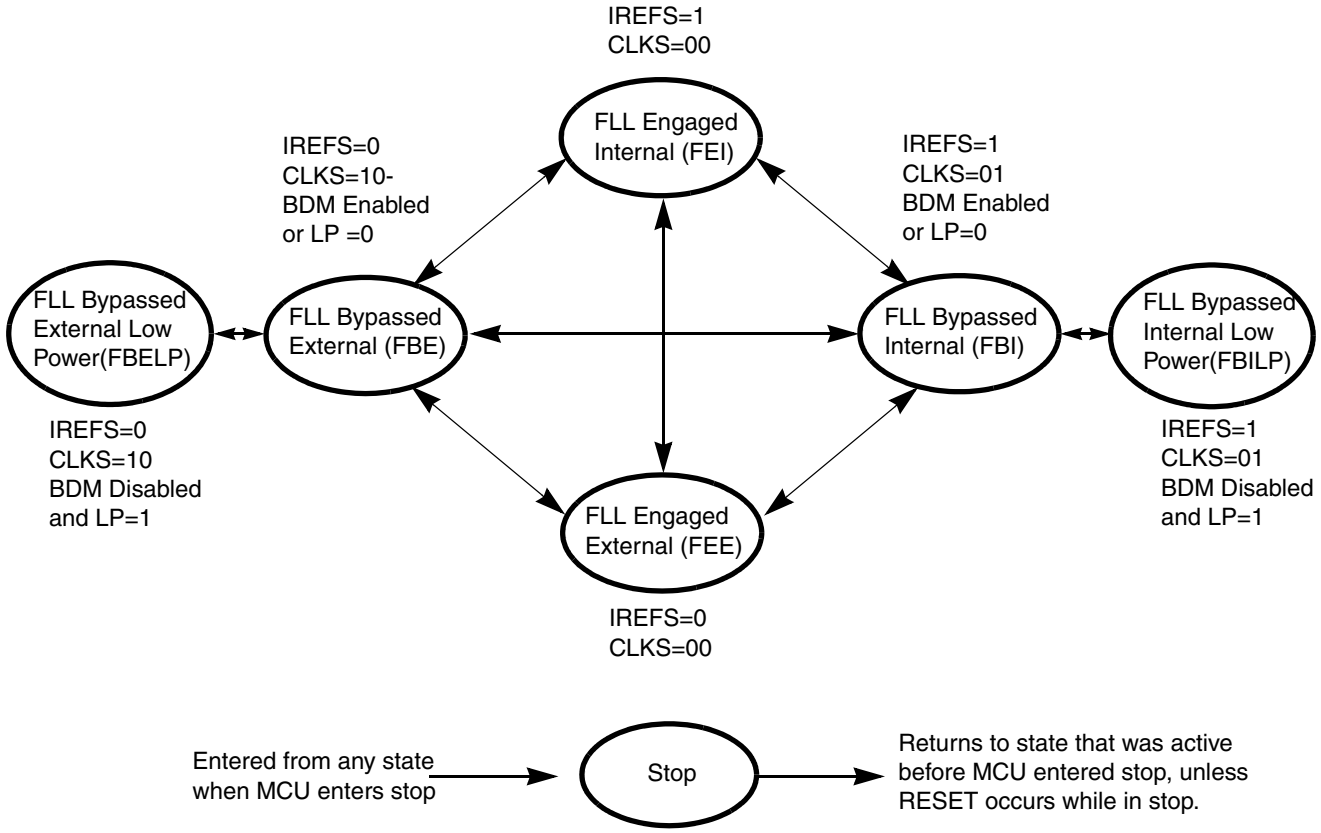


Figure 15-7. Clock Switching Modes

15.4.1.1 FLL Engaged Internal (FEI)

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

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

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

15.4.1.2 FLL Engaged External (FEE)

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

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

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

15.4.1.3 FLL Bypassed Internal (FBI)

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

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

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

15.4.1.4 FLL Bypassed Internal Low Power (FBILP)

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

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

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

15.4.1.5 FLL Bypassed External (FBE)

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

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

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

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

15.4.1.6 FLL Bypassed External Low Power (FBELP)

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

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

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

15.4.1.7 Stop

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

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

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

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

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

15.4.2 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.

15.4.3 Bus Frequency Divider

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

15.4.4 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.

15.4.5 Internal Reference Clock

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

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

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

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

15.4.6 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 ICSERCLK, 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 ICSERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the Device Overview chapter).

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

15.4.7 Fixed Frequency Clock

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

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

Chapter 16
Development Support

16.1 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.

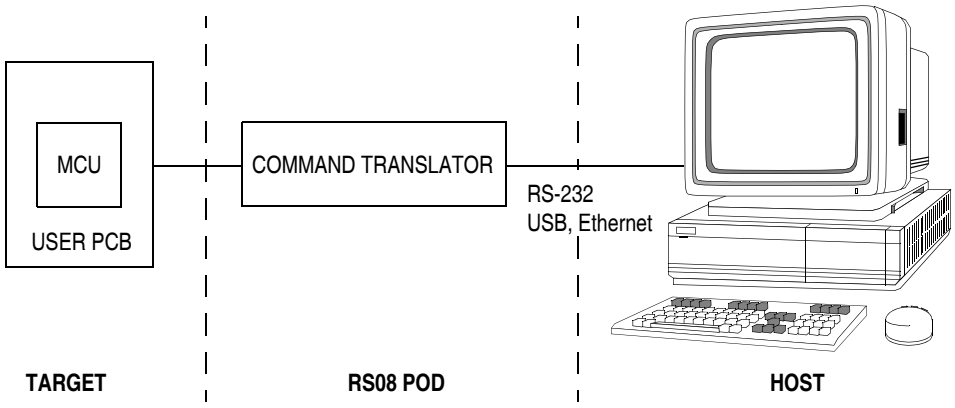


Figure 16-1. Connecting MCU to Host for Debugging

16.1.1 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

16.2 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 16-2 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 $\overline{\text{RESET}}$ signal is included in the connector to allow a direct hardware method for the host to force or monitor (if $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$ as shown in Figure 16-2. For V_{PP} requirements see the FLASH specifications in theMCU electricals appendix.

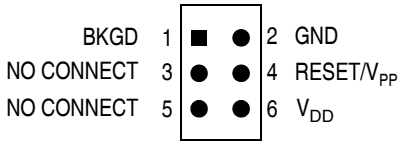


Figure 16-2. Standard RS08 BDM Tool Connector

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 16.2.2, “Communication Details."

16.2.1 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 16.2.2, “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 should be done in normal user mode without using BDM.

16.2.2 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 16-3 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.

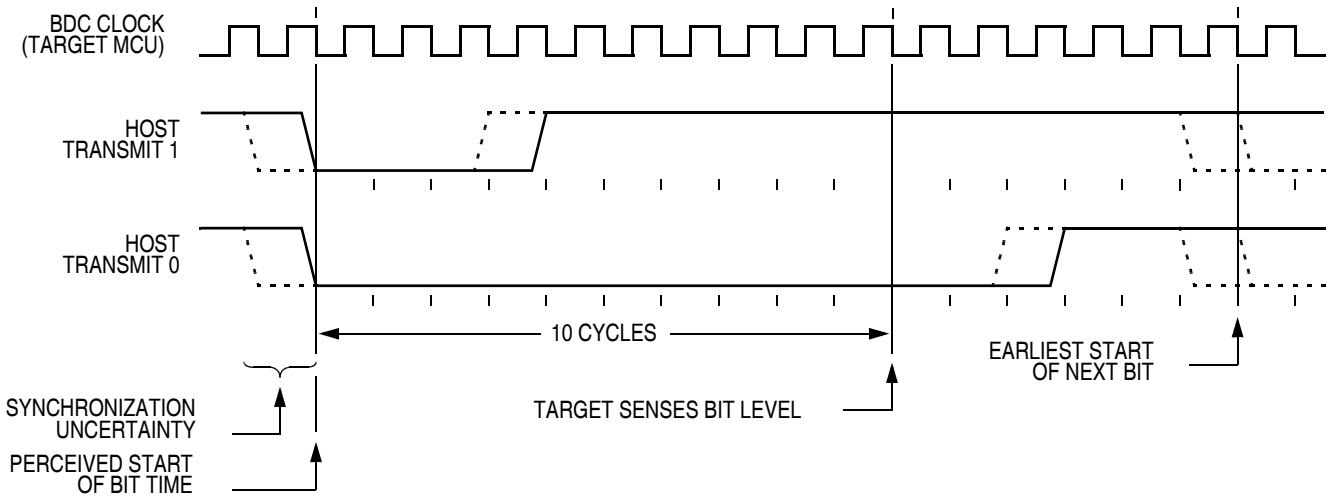


Figure 16-3. BDC Host-to-Target Serial Bit Timing

Figure 16-4 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 should sample the bit level approximately 10 cycles after it started the bit time.

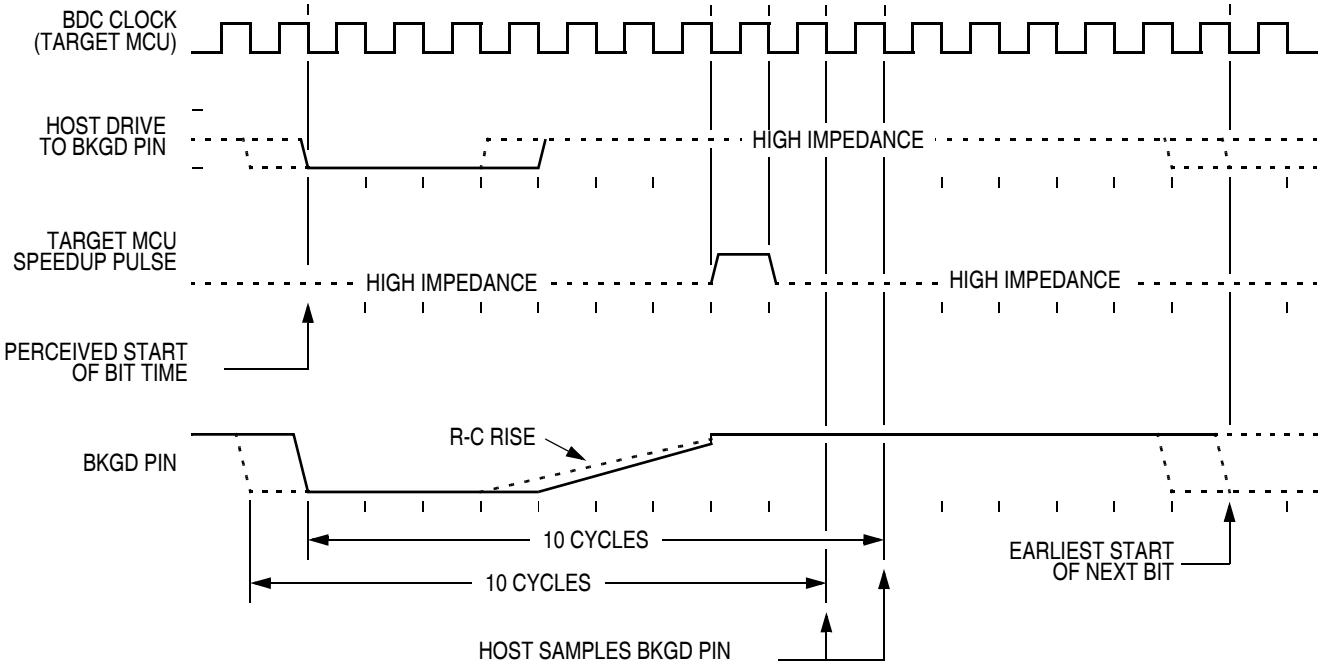


Figure 16-4. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 16-5 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.

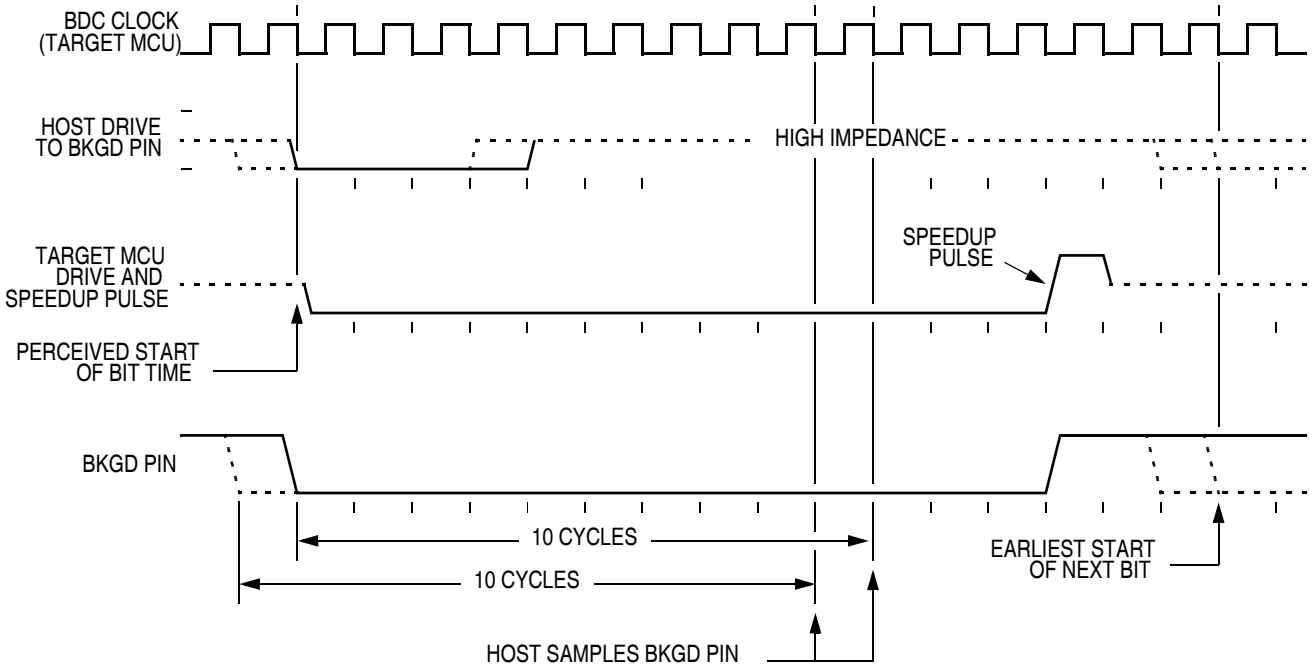


Figure 16-5. BDM Target-to-Host Serial Bit Timing (Logic 0)

16.2.3 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 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.

16.3 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.

16.3.1 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 16-6. BDC Status and Control Register (BDCSCR)

Table 16-1. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	**Enable BDM (Permit Active Background Mode)** — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 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	**Background Mode Active Status** — This is a read-only status bit. 0 BDM not active (user application program running). 1 BDM active and waiting for serial commands.

Table 16-1. BDCSCR Register Field Descriptions (continued)

Field	Description
5 BKPTEN	**BDC Breakpoint Enable** — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored 0 BDC breakpoint disabled. 1 BDC breakpoint enabled.
4 FTS	**Force/Tag Select** — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction. 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode).
2 WS	**Wait or Stop Status** — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands. 0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active). 1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode.
1 WSF	**Wait or Stop Failure Status** — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.) 0 Memory access did not conflict with a wait or stop instruction. 1 Memory access command failed because the CPU entered wait or stop mode.

16.3.2 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 16.5, “BDC Hardware Breakpoint.”

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
W																
Any Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 16-7. BDC Breakpoint Match Register (BDCBKPT)

16.4 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 16-2 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 16-2 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[1]
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 16-2. RS08 BDC Command Summary

Command Mnemonic	Active Background Mode/ Non-Intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a(1)	Request a timed reference pulse to determine target BDC communication speed
BDC_RESET	Any CPU mode	18(2)	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(3)	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(4)	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 16-2. 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[5]	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[6]	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[7]	Read the 14-bit shadow program counter (SPC) RD16=0:0:SPC
WRITE_SPC	Active background mode	4F/WD16/d[8]	Write 14-bit shadow program counter (SPC) WD16 = x:x:SPC, the two most significant bits shown by “x” are ignored by target

1. The SYNC command is a special operation which does not have a command code.
2. 18 was HCS08 BDC command for TAGGO.
3. Each RD requires a delay between host read data byte and next read, command ends when target detects a soft-reset.
4. Each WD requires a delay between host write data byte and next byte, command ends when target detects a soft-reset.
5. HCS08 BDC had separate READ_CCR and READ_PC commands, the RS08 BDC combined this commands.
6. HCS08 BDC had separate WRITE_CCR and WRITE_PC commands, the RS08 BDC combined this commands.
7. 6F is READ_SP (read stack pointer) for HCS08 BDC.
8. 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):

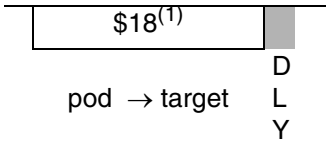
- Waits for BKGD to return to a logic 1
- Delays 16 cycles to allow the host to stop driving the high speedup pulse

- 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.

16.4.1 BDC_RESET

Reset the MCU — Any CPU Mode

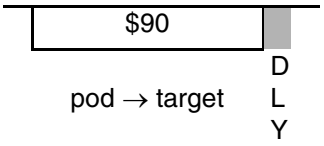


1. $18 is the HCS08 BDC command for TAGGO

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.

16.4.2 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.

16.4.3 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.

16.4.4 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

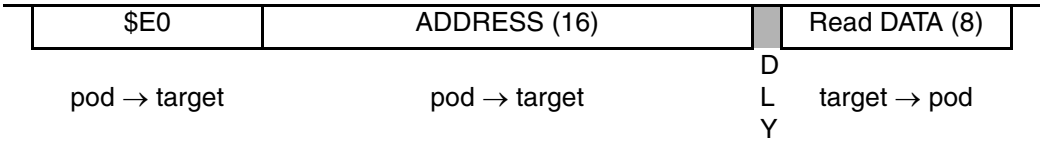
bits BDMACT, WS, and WSF, which are read-only status indicators, and ENBDM, which cannot be cleared while BDM is active.

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.

16.4.5 READ_BYTE

Read Data from Target Memory Location — Non-intrusive

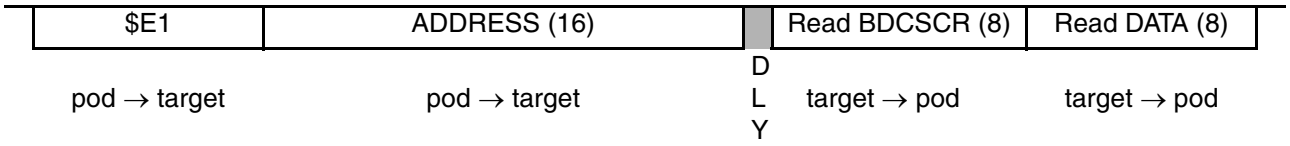


This command is used to read the contents of a memory location in the target MCU without checking the BDC status to ensure the data is valid. In systems where the target is currently in active background mode or is known to be executing a program which has no STOP or WAIT instructions, READ_BYTE is faster 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.

16.4.6 READ_BYTE_WS

Read Data from Target and Report Status — Non-intrusive

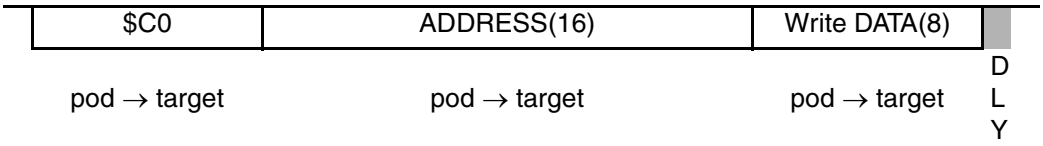


This is the command normally used by a host debug system to perform general-purpose memory read operations. In addition to returning the data from the requested target memory location, this command returns the contents of the BDC status and control register. The status information can be used to determine 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.

16.4.7 WRITE_BYTE

Write Data to Target Memory Location — Non-intrusive

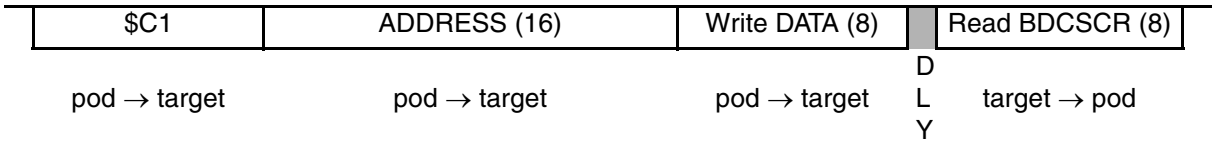


This command is used to write the contents of a memory location in the target MCU without checking the BDC status to ensure the write was completed successfully. In systems where the target is currently in active background mode or is known to be executing a program that has no STOP or WAIT instructions, 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.

16.4.8 WRITE_BYTE_WS

Write Data to Target and Report Status — Non-intrusive

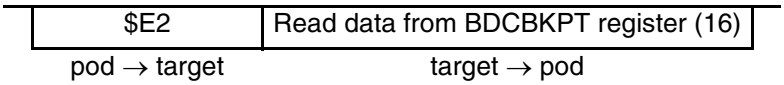


This is the command normally used by a host debug system to perform general-purpose memory write operations. In addition to performing the requested write to a target memory location, this command returns the contents of the BDC status and control register. The status information can be used to determine 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.

16.4.9 READ_BKPT

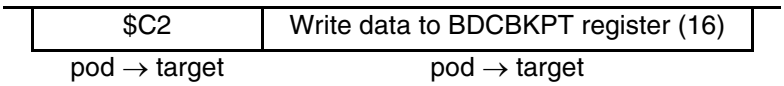
Read 16-Bit BDC Breakpoint Register (BDCBKPT) — Non-intrusive



This command is used to read the 16-bit BDCBKPT address match register in the hardware breakpoint logic in the BDC.

16.4.10 WRITE_BKPT

Write 16-Bit BDC Breakpoint Register (BDCBKPT) — Non-intrusive

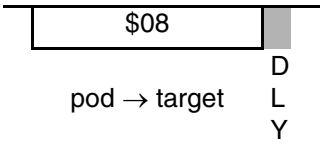


This command is used to write a 16-bit address value into the BDCBKPT register in the BDC. This establishes the address of a breakpoint. The BKPTEN bit in the BDCSCR determines whether the breakpoint is enabled. If BKPTEN = 1 and the FTS control bit in the BDCSCR is set (force), a successful match between 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.

16.4.11 GO

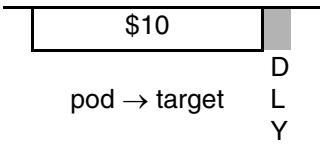
Start Execution of User Program Starting at Current PC — Active Background Mode



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.

16.4.12 TRACE1

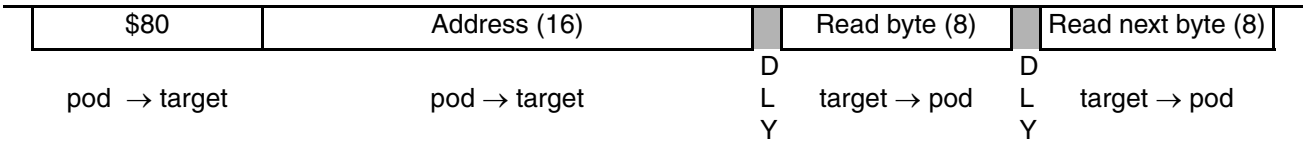
Run One User Instruction Starting at the Current PC — Active Background Mode



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.

16.4.13 READ_BLOCK

Read a block of data from target memory — Active Background Mode

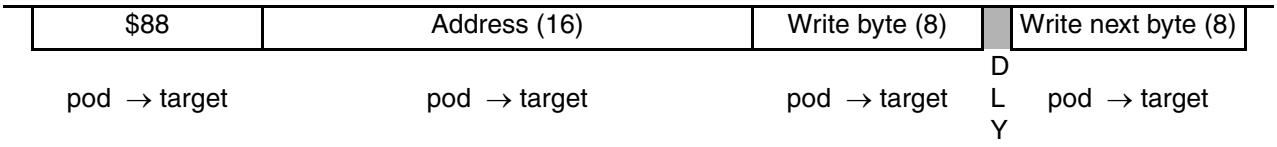


This command is used to read the contents of a block of memory starting at the location provided in the command. Because this command is only available in active background mode, the CPU cannot enter wait or stop; therefore the command does not return the contents of the BDMSCR. This command will continue 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.

16.4.14 WRITE_BLOCK

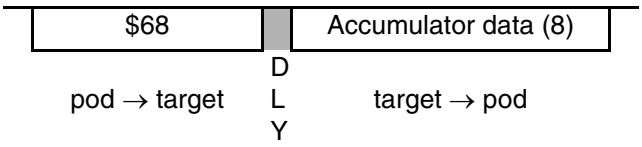
Write a block of data to target memory — Active Background Mode



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.

16.4.15 READ_A

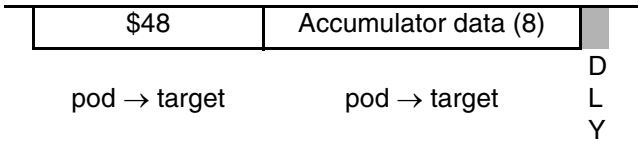
Read Accumulator A of the Target CPU — Active Background Mode



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.

16.4.16 WRITE_A

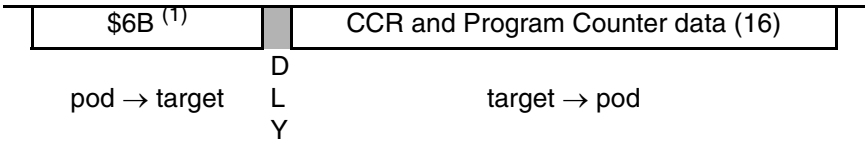
Write Accumulator A of the Target CPU — Active Background Mode



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.

16.4.17 READ_CCR_PC

Reads the CCR and the Program Counter of the Target CPU — Active Background Mode



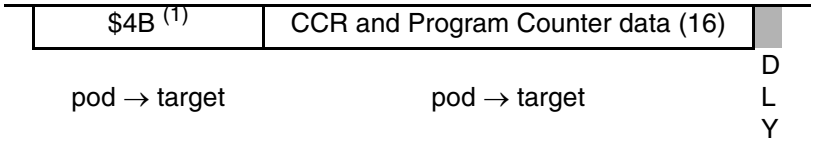
1. $6B is the HCS08 BDC command for READ_PC, the HCS08 CCR was read using READ_CCR

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.

16.4.18 WRITE_CCR_PC

Write the CCR and Program Counter of the Target CPU — Active Background Mode

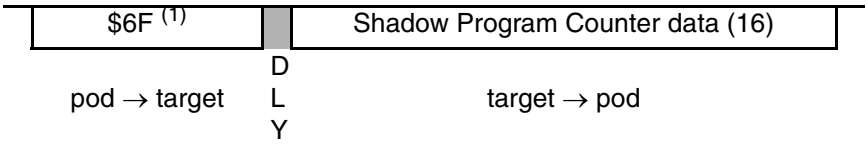


1. $4B is the HCS08 BDC command for WRITE_PC, the HCS08 CCR was written using WRITE_CCR

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.

16.4.19 READ_SPC

Reads the Shadow Program Counter of the Target CPU — Active Background Mode

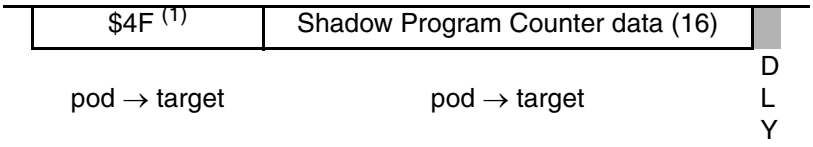


1. $6F is the HCS08 BDC command for READ_SP (stack pointer)

Read the contents of the 14-bit shadow program counter (PC) of the target CPU.

16.4.20 WRITE_SPC

Write the Shadow Program Counter of the Target CPU — Active Background Mode



1. $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.

16.5 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.

16.6 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.

16.7 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 x 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

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

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

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2008-2012. All rights reserved.

MC9RS08KB12
Rev. 4 , 1/2012

freescale™
semiconductor

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View MC9RS08KB4CFK on WIN SOURCE
- NXP / Nexperia Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management