

# **MC9S08MP16 Reference Manual**

**Devices Supported:**  
**MC9S08MP16**  
**MC9S08MP12**

Document Number: MC9S08MP16RM  
Rev. 1  
09/2009

## **How to Reach Us:**

### **Home Page:**

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

### **Web Support:**

<http://www.freescale.com/support>

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
1-800-521-6274 or +1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

Europe, Middle East, and Africa:  
Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
8129 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### **Japan:**

Freescale Semiconductor Japan Ltd. Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### **Asia/Pacific:**

Freescale Semiconductor China Ltd.  
Exchange Building 23F  
No. 118 Jianguo Road  
Chaoyang District  
Beijing 100022  
China  
+86 10 5879 8000  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or +1-303-675-2140  
Fax: +1-303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performances may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.

MC9S08MP16RM  
Rev. 1  
09/2009

# Contents

Section Number	Title	Page
<b>Chapter 1 Device Overview</b>		
1.1	Devices in the MC9S08MP16 Series .....	19
1.2	Features .....	19
1.3	MCU Block Diagram .....	21
1.4	System Clock Distribution .....	23
<b>Chapter 2 Pins and Connections</b>		
2.1	Device Pin Assignment .....	27
2.2	Recommended System Connections .....	31
2.2.1	Power .....	32
2.2.2	Oscillator (XOSC) .....	32
2.2.3	RESET Pin .....	32
2.2.4	Background / Mode Select (BKGD/MS) .....	33
2.2.5	General-Purpose I/O and Peripheral Ports .....	34
<b>Chapter 3 Modes of Operation</b>		
3.1	Introduction .....	37
3.2	Features .....	37
3.3	Run Mode .....	37
3.4	Active Background Mode .....	37
3.5	Wait Mode .....	38
3.6	Stop Modes .....	38
3.6.1	Stop3 Mode .....	39
3.6.2	Stop2 Mode .....	40
3.6.3	On-Chip Peripheral Modules in Stop Modes .....	40
<b>Chapter 4 Memory</b>		
4.1	MC9S08MP16 Series Memory Map .....	43
4.2	Reset and Interrupt Vector Assignments .....	44
4.3	Register Addresses and Bit Assignments .....	45
4.4	RAM .....	55
4.5	FLASH .....	55
4.5.1	Features .....	56
4.5.2	Program and Erase Times .....	56

<b>Section Number</b>	<b>Title</b>	<b>Page</b>
4.5.3	Program and Erase Command Execution .....	57
4.5.4	Burst Program Execution.....	58
4.5.5	Access Errors .....	60
4.5.6	FLASH Block Protection.....	60
4.5.7	Vector Redirection .....	61
4.6	Security.....	61
4.7	FLASH Registers and Control Bits.....	63
4.7.1	FLASH Clock Gating .....	63
4.7.2	FLASH Clock Divider Register (FCDIV) .....	63
4.7.3	FLASH Options Register (FOPT and NVOPT).....	64
4.7.4	FLASH Configuration Register (FCNFG) .....	65
4.7.5	FLASH Protection Register (FPROT and NVPROT) .....	66
4.7.6	FLASH Status Register (FSTAT).....	67
4.7.7	FLASH Command Register (FCMD).....	67

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

5.1	Introduction .....	69
5.2	Features .....	69
5.3	MCU Reset .....	69
5.4	Computer Operating Properly (COP) Watchdog.....	70
5.5	Interrupts .....	71
5.5.1	Interrupt Stack Frame .....	72
5.5.2	Interrupt Vectors, Sources, Priority Masks and Local Masks.....	73
5.6	Low-Voltage Detect (LVD) System .....	75
5.6.1	Power-On Reset Operation .....	75
5.6.2	Low-Voltage Detection (LVD) Reset Operation.....	75
5.6.3	Low-Voltage Warning (LVW) Interrupt Operation.....	75
5.7	Peripheral Clock Gating .....	75
5.8	Reset, Interrupt, and System Control Registers and Control Bits .....	76
5.8.1	System Reset Status Register (SRS) .....	76
5.8.2	System Background Debug Force Reset Register (SBDFR).....	77
5.8.3	System Options Register 1 (SOPT1) .....	78
5.8.4	System Options Register 2 (SOPT2) .....	79
5.8.5	System Options Register 3 (SOPT3) .....	80
5.8.6	System Device Identification Register (SDIDH, SDIDL).....	81
5.8.7	System Power Management Status and Control 1 Register (SPMSC1).....	82
5.8.8	System Power Management Status and Control 2 Register (SPMSC2).....	83
5.8.9	System Clock Gating Control 1 Register (SCGC1).....	84
5.8.10	System Clock Gating Control 2 Register (SCGC2).....	85

Section Number	Title	Page
	<b>Chapter 6 Parallel Input/Output Control</b>	
6.1	Port Data and Data Direction .....	87
6.2	Pull-up, Slew Rate, and Drive Strength .....	88
	6.2.1 Port Internal Pull-up Enable .....	88
	6.2.2 Port Slew Rate Enable .....	88
	6.2.3 Port Drive Strength Select .....	88
6.3	Pin Behavior in Stop Modes.....	89
6.4	Parallel I/O and Pin Control Registers .....	89
	6.4.1 Port A Registers .....	89
	6.4.2 Port B Registers .....	91
	6.4.3 Port C Registers .....	93
	6.4.4 Port D Registers .....	95
	6.4.5 Port E Registers .....	97
	6.4.6 Port F Registers.....	99
	<b>Chapter 7 Keyboard Interrupt (S08KBIv2)</b>	
7.1	Introduction .....	103
	7.1.1 KBI Clock Gating .....	103
	7.1.2 Features .....	104
	7.1.3 Modes of Operation .....	104
	7.1.4 Block Diagram.....	104
7.2	External Signal Description .....	105
7.3	Register Definition .....	106
	7.3.1 KBI Interrupt Status and Control Register (KBIXSC) .....	106
	7.3.2 KBI Interrupt Pin Select Register (KBIXPE) .....	106
	7.3.3 KBI Interrupt Edge Select Register (KBIXES) .....	107
7.4	Functional Description .....	107
	7.4.1 Edge Only Sensitivity .....	107
	7.4.2 Edge and Level Sensitivity .....	108
	7.4.3 Pull-up/Pull-down Resistors .....	108
	7.4.4 Keyboard Interrupt Initialization .....	108
	<b>Chapter 8 Central Processor Unit (S08CPUv5)</b>	
8.1	Introduction .....	109
	8.1.1 Features .....	109
8.2	Programmer's Model and CPU Registers .....	110
	8.2.1 Accumulator (A) .....	110
	8.2.2 Index Register (H:X) .....	110
	8.2.3 Stack Pointer (SP) .....	111

Section Number	Title	Page
8.2.4	Program Counter (PC) .....	111
8.2.5	Condition Code Register (CCR) .....	111
8.3	Addressing Modes .....	113
8.3.1	Inherent Addressing Mode (INH) .....	113
8.3.2	Relative Addressing Mode (REL) .....	113
8.3.3	Immediate Addressing Mode (IMM) .....	113
8.3.4	Direct Addressing Mode (DIR) .....	113
8.3.5	Extended Addressing Mode (EXT) .....	114
8.3.6	Indexed Addressing Mode .....	114
8.4	Special Operations .....	115
8.4.1	Reset Sequence .....	115
8.4.2	Interrupt Sequence .....	115
8.4.3	Wait Mode Operation .....	116
8.4.4	Stop Mode Operation .....	116
8.4.5	BGND Instruction .....	117
8.5	HCS08 Instruction Set Summary .....	118

## Chapter 9 Analog-to-Digital Converter (S08ADC12V1)

9.1	Introduction .....	129
9.2	Module Configurations .....	129
9.2.1	ADC Clock Gating .....	129
9.2.2	Analog Supply and Voltage Reference Connections .....	129
9.2.3	Configurations for Stop Modes .....	129
9.2.4	Channel Assignments .....	129
9.2.5	Alternate Clock .....	130
9.2.6	Hardware Trigger .....	130
9.2.7	Temperature Sensor .....	131
9.2.8	Features .....	133
9.2.9	ADC Module Block Diagram .....	133
9.3	External Signal Description .....	134
9.3.1	Analog Power ( $V_{DDA}$ ) .....	135
9.3.2	Analog Ground ( $V_{SSA}$ ) .....	135
9.3.3	Voltage Reference High ( $V_{REFH}$ ) .....	135
9.3.4	Voltage Reference Low ( $V_{REFL}$ ) .....	135
9.3.5	Analog Channel Inputs ( $ADx$ ) .....	135
9.4	Register Definition .....	135
9.4.1	Status and Control Register 1 (ADCSC1) .....	136
9.4.2	Status and Control Register 2 (ADCSC2) .....	137
9.4.3	Data Result High Register (ADCRH) .....	138
9.4.4	Data Result Low Register (ADCRL) .....	138
9.4.5	Compare Value High Register (ADCCVH) .....	139

Section Number	Title	Page
9.4.6	Compare Value Low Register (ADCCVL) .....	139
9.4.7	Configuration Register (ADCCCFG).....	139
9.4.8	Pin Control 1 Register (APCTL1) .....	141
9.4.9	Pin Control 2 Register (APCTL2) .....	142
9.4.10	Pin Control 3 Register (APCTL3) .....	143
9.5	Functional Description .....	144
9.5.1	Clock Select and Divide Control .....	144
9.5.2	Input Select and Pin Control .....	145
9.5.3	Hardware Trigger.....	145
9.5.4	Conversion Control.....	145
9.5.5	Automatic Compare Function.....	148
9.5.6	MCU Wait Mode Operation.....	149
9.5.7	MCU Stop3 Mode Operation.....	149
9.5.8	MCU Stop2 Mode Operation.....	150
9.6	Initialization Information .....	150
9.6.1	ADC Module Initialization Example .....	150
9.7	Application Information .....	152
9.7.1	External Pins and Routing .....	152
9.7.2	Sources of Error .....	154

## Chapter 10 Cyclic Redundancy Check Generator (S08CRCV3)

10.1	Introduction .....	159
10.1.1	Features .....	159
10.1.2	Features .....	161
10.1.3	Modes of Operation .....	161
10.1.4	Block Diagram .....	162
10.2	External Signal Description .....	162
10.3	Register Definition .....	162
10.3.1	Memory Map .....	162
10.3.2	Register Descriptions .....	163
10.4	Functional Description .....	164
10.4.1	ITU-T (CCITT) Recommendations and Expected CRC Results.....	165
10.4.2	Programming model extension for CF1Core .....	166
10.4.3	Transpose feature .....	167
10.5	Initialization Information .....	168

## Chapter 11 Digital to Analog Converter (S08DACV1)

11.1	Introduction .....	169
11.2	Module Configurations .....	169
11.2.1	DAC Clock Gating.....	169

Section Number	Title	Page
11.2.2	DAC/HSCMP Configuration .....	169
11.2.3	Reference Options.....	169
11.2.4	Features .....	171
11.2.5	Modes of Operation .....	171
11.2.6	Block Diagram.....	171
11.3	Memory Map/Register Definition .....	172
11.3.1	DAC Control Register (DACCTRL) .....	172
11.3.2	Functional Description.....	172
11.4	Resets .....	173
11.5	Clocks.....	173
11.6	Interrupts .....	173

## Chapter 12 Flextimer Module (S08FTMV2)

12.1	Introduction .....	175
12.2	Module Configurations .....	175
12.2.1	FTM1 and FTM2 Clocking.....	175
12.2.2	FTM External Clock .....	175
12.2.3	FTM External Clock Pin Repositioning .....	176
12.2.4	HSCMPx/FTM1 Input Capture .....	176
12.2.5	FTM Fault .....	176
12.2.6	FTM Input Synchronization.....	176
12.2.7	FTM/ADC Triggering.....	177
12.2.8	FTM/PDB Triggering .....	177
12.2.9	Features .....	179
12.2.10	Modes of Operation .....	180
12.2.11	Block Diagram.....	180
12.3	Signal Description .....	182
12.3.1	FTMxCLK — FTM External Clock .....	182
12.3.2	FTMxCHn — FTM Channel (n) I/O Pin(s).....	182
12.4	Memory Map and Register Definition .....	183
12.4.1	Module Memory Map .....	183
12.4.2	Register Descriptions .....	185
12.4.3	FTM Status and Control Register (FTMxSC) .....	185
12.4.4	FTM Counter Registers (FTMxCNTH:FTMxCNTL).....	186
12.4.5	FTM Counter Modulo Registers (FTMxMODH:FTMxMODL).....	187
12.4.6	FTM Channel (n) Status and Control Register (FTMxCnSC).....	188
12.4.7	FTM Channel Value Registers (FTMxCnVH:FTMxCnVL) .....	190
12.4.8	FTM Counter Initial Value Registers (FTMxCNTINH:FTMxCNTINL).....	191
12.4.9	FTM Capture and Compare Status Register (FTMxSTATUS).....	192
12.4.10	FTM Features Mode Selection Register (FTMxMODE) .....	193
12.4.11	FTM Synchronization Register (FTMxSYNC) .....	195

Section Number	Title	Page
12.4.12	FTM Initial State for Channels Output Register (FTMxOUTINIT).....	196
12.4.13	FTM Output Mask Register (FTMxOUTMASK) .....	196
12.4.14	FTM Function For Linked Channels Register (FTMxCOMBINEm) .....	197
12.4.15	FTM Deadtime Insertion Control Register (FTMxDEADTIME) .....	198
12.4.16	FTM External Trigger Register (FTMxEXTTRIG).....	201
12.4.17	FTM Channels Polarity Register (FTMxPOL) .....	202
12.4.18	FTM Fault Mode Status Register (FTMxFMS).....	202
12.4.19	FTM Input Capture Filter Control Register (FTMxFILTERm).....	203
12.4.20	FTM Fault Input Filter Control Register (FTMxFLTFILTER).....	204
12.4.21	FTM Fault Control Register (FTMxFLTCTRL).....	204
12.4.22	FTM Register for Future Use (FTMxRFU) .....	205
12.5	Functional Description .....	205
12.5.1	Clock Source.....	206
12.5.2	Prescaler.....	207
12.5.3	Counter.....	207
12.5.4	Input capture mode .....	212
12.5.5	Output compare mode.....	214
12.5.6	Edge-aligned PWM (EPWM) mode .....	216
12.5.7	Center-aligned PWM (CPWM) mode .....	217
12.5.8	Combine mode .....	220
12.5.9	Complementary mode.....	228
12.5.10	Load of the registers with write buffers .....	229
12.5.11	PWM synchronization .....	230
12.5.12	Deadtime insertion .....	241
12.5.13	Output Mask .....	242
12.5.14	Fault control mode .....	243
12.5.15	Polarity Control.....	246
12.5.16	Initialization .....	247
12.5.17	Features Priority.....	247
12.5.18	Match Trigger .....	247
12.5.19	Initialization Trigger .....	248
12.5.20	Capture Test Mode.....	250
12.5.21	TPM Emulation.....	251
12.6	Reset Overview .....	252
12.7	FTM Interrupts .....	252
12.7.1	Timer Overflow Interrupt.....	252
12.7.2	Channel (n) Interrupt .....	252
12.7.3	Fault Interrupt .....	252
13.1	Introduction .....	253

## Chapter 13 High Speed Analog Comparator 5-V (S08HSCMPV1)

<b>Section Number</b>	<b>Title</b>	<b>Page</b>
13.2	Module Configurations .....	253
13.2.1	HSCMP Clock Gating .....	253
13.2.2	Analog Supply Connections .....	253
13.2.3	HSCMP Analog Input Configuration .....	253
13.2.4	PDB2/HSCMP Windowing/Sampling Input Configuration .....	254
13.2.5	HSCMP Output Configuration .....	255
13.3	Features .....	258
13.4	Block Diagram .....	259
13.5	Pin Descriptions .....	260
13.5.1	External Pins .....	260
13.6	Functional Description .....	261
13.6.1	HSCMP Functional Modes .....	261
13.6.2	Hysteresis .....	270
13.6.3	Startup and Operation .....	271
13.6.4	Low Pass Filter .....	271
13.7	Interrupts .....	273
13.8	Memory Map & Register Definitions .....	273
13.8.1	Control Register 0 (HSCMPxCR0) .....	274
13.8.2	Control Register 1 (HSCMPxCR1) .....	274
13.8.3	Filter Period Register (HSCMPxFPR) .....	275
13.8.4	Status & Control Register (HSCMPxSCR) .....	276
13.8.5	Pin Control Register (HSCMPxPCR) .....	277

## Chapter 14 Internal Clock Source (S08ICSV3)

14.1	Introduction .....	279
14.2	Module Configuration .....	279
14.2.1	System Clock Distribution .....	279
14.2.2	External Oscillator .....	279
14.2.3	Stop3 Operation .....	279
14.2.4	Features .....	281
14.2.5	Block Diagram .....	281
14.2.6	Modes of Operation .....	282
14.3	External Signal Description .....	283
14.4	Register Definition .....	283
14.4.1	ICS Control Register 1 (ICSC1) .....	284
14.4.2	ICS Control Register 2 (ICSC2) .....	286
14.4.3	ICS Trim Register (ICSTRM) .....	286
14.4.4	ICS Status and Control (ICSSC) .....	287
14.5	Functional Description .....	289
14.5.1	Operational Modes .....	289
14.5.2	Mode Switching .....	291

<b>Section Number</b>	<b>Title</b>	<b>Page</b>
14.5.3	Bus Frequency Divider .....	292
14.5.4	Low Power Bit Usage .....	292
14.5.5	DCO Maximum Frequency with 32.768 kHz Oscillator .....	292
14.5.6	Internal Reference Clock .....	292
14.5.7	External Reference Clock .....	293
14.5.8	Fixed Frequency Clock .....	293
14.5.9	Local Clock .....	293
<b>Chapter 15 Inter-Integrated Circuit (S08IICV3)</b>		
15.1	Introduction .....	295
15.2	Module Configuration .....	295
15.2.1	IIC Clock Gating .....	295
15.2.2	Pin Repositioning .....	295
15.2.3	IIC Pin Filtering .....	296
15.2.4	Features .....	300
15.2.5	Modes of Operation .....	300
15.2.6	Block Diagram .....	301
15.3	External Signal Description .....	301
15.3.1	SCL — Serial Clock Line .....	301
15.3.2	SDA — Serial Data Line .....	301
15.4	Register Definition .....	302
15.4.1	Module Memory Map .....	302
15.4.2	IIC Address Register 1 (IICA1) .....	302
15.4.3	IIC Frequency Divider Register (IICF) .....	303
15.4.4	IIC Control Register (IICC1) .....	305
15.4.5	IIC Status Register (IICS) .....	306
15.4.6	IIC Data I/O Register (IICD) .....	307
15.4.7	IIC Control Register 2 (IICC2) .....	308
15.4.8	IIC SMBus Control and Status Register (IICSLMB) .....	309
15.4.9	IIC Address Register 2 (IICA2) .....	310
15.4.10	IIC SCL Low Time Out Register High (IICSLTH) .....	310
15.4.11	IIC SCL LowTime Out register Low (IICSLTL) .....	310
15.4.12	IIC Programmable Input Glitch Filter (IICFLT) .....	310
15.5	Functional Description .....	311
15.5.1	IIC Protocol .....	311
15.5.2	10-bit Address .....	315
15.5.3	Address Matching .....	316
15.5.4	System Management Bus Specification .....	316
15.6	Resets .....	318
15.7	Interrupts .....	318
15.7.1	Byte Transfer Interrupt .....	319

Section Number	Title	Page
15.7.2	Address Detect Interrupt .....	319
15.7.3	Arbitration Lost Interrupt.....	319
15.7.4	Timeouts Interrupt in SMBus .....	319
15.7.5	Programmable input glitch filter.....	320
15.8	Initialization/Application Information .....	320

## Chapter 16 Interrupt Priority Controller (S08IPCV1)

16.1	Introduction .....	323
16.1.1	Stop3 Operation .....	324
16.1.2	Features .....	325
16.1.3	Modes of Operation .....	325
16.1.4	Block Diagram.....	325
16.2	External Signal Description .....	326
16.2.1	INTIN[47:0] — Interrupt Source Interrupt Request Input .....	327
16.2.2	VFETCH — Vector Fetch Indicator from HCS08 CPU.....	327
16.2.3	IADB[5:0] — Address Bus Input from HCS08 CPU.....	327
16.2.4	INTOUT[47:0] — Interrupt Request to HCS08 CPU .....	327
16.3	Register Definition .....	327
16.3.1	IPC Status and Control Register (IPCSC) .....	327
16.3.2	Interrupt Priority Mask Pseudo Stack Register (IPMPS) .....	328
16.3.3	Interrupt Level Setting Registers (ILRS0–ILRS11) .....	329
16.4	Functional Description .....	330
16.4.1	Interrupt Priority Level Register .....	330
16.4.2	Interrupt Priority Level Comparator Set.....	330
16.4.3	Interrupt Priority Mask Update and Restore Mechanism .....	331
16.4.4	The Integration and Application of the IPC.....	331
16.5	Application Examples .....	331
16.6	Initialization/Application Information .....	333

## Chapter 17 Modulo Timer (S08MTIMV1)

17.1	Introduction .....	335
17.2	Module Configurations .....	335
17.2.1	MTIM Clock Gating .....	335
17.2.2	MTIM External Clock .....	335
17.2.3	MTIM External Clock Pin Repositioning.....	335
17.2.4	Features .....	338
17.2.5	Modes of Operation .....	338
17.3	External Signal Description .....	339
17.4	Memory Map and Register Definition .....	339
17.4.1	Memory Map (Register Summary).....	339

Section Number	Title	Page
17.4.2	Register Descriptions .....	339
17.5	Functional Description .....	343
17.5.1	MTIM Operation Example .....	344

## Chapter 18 Programmable Delay Block (S08PDBV1)

18.1	Introduction .....	345
18.2	Module Configurations .....	345
18.2.1	PDB Clock Gating .....	345
18.2.2	PDB Input Trigger Assignments.....	346
18.2.3	PDB Trigger Output Usage .....	346
18.2.4	Features .....	349
18.2.5	Modes of Operation .....	349
18.2.6	Block Diagram .....	349
18.3	Memory Map/Register Definitions .....	354
18.3.1	PDB Control Register 1 (PDBxCTRL1) .....	354
18.3.2	PDB Control Register 2 (PDBxCTRL2) .....	355
18.3.3	PDB Delay A & Delay B Registers .....	356
18.3.4	PDB Modulus Registers.....	357
18.3.5	PDB Counter Registers (PDBxCNTH & PDBxCNTL) .....	357
18.3.6	PDB Status and Control Register (PDBxSCR).....	357
18.4	Functional Description .....	358
18.4.1	Miscellaneous Concerns and SoC Integration .....	358
18.4.2	Impacts of Using the Prescaler on Timing Resolution .....	358
18.4.3	Resets .....	359
18.4.4	Clocks .....	359
18.4.5	Interrupts .....	359

## Chapter 19 Programmable Gain Amplifier (S08PGAV1)

19.1	Introduction .....	361
19.2	Module Configurations .....	361
19.2.1	PGA Clock Gating .....	361
19.2.2	PGA Clock Connections .....	361
19.2.3	Analog Supply Connections .....	361
19.2.4	PGA Inputs .....	361
19.2.5	PDB1/PGA Trigger Input .....	362
19.2.6	PGA Outputs .....	362
19.2.7	Default Reset Value for NUM_CLK_GS Bits.....	362
19.2.8	Features .....	364
19.3	Modes of Operation.....	364
19.3.1	PGA Power Down .....	364

19.3.2	PGA Startup .....	365
19.3.3	PGA Calibration .....	365
19.3.4	PGA Mission Mode .....	367
19.4	Memory Map/Register Definitions .....	368
19.4.1	Control Register 0 (PGACNTL0).....	368
19.4.2	Control Register 1 (PGACNTL1).....	370
19.4.3	Control Register 2 (PGACNTL2).....	370
19.4.4	Status Register (PGASTS).....	371
19.5	Functional Description .....	371
19.5.1	Transfer Function.....	371
19.5.2	Options for On-Chip Analog Conversions .....	372
19.5.3	PGA Prerequisites.....	372
19.5.4	Analog Block Diagram .....	372
19.5.5	PGA Clock Requirements.....	373
19.5.6	Effects on ADC Latency .....	377
19.5.7	ADC Triggers.....	378
19.5.8	Interrupts .....	378
19.5.9	Reset Considerations.....	378

## Chapter 20 Real-Time Counter (S08RTCV1)

20.1	Introduction .....	379
20.2	Module Configurations .....	379
20.2.1	RTC Clock Gating .....	379
20.2.2	RTC/PDB Trigger .....	379
20.2.3	RTC Clock Sources.....	379
20.2.4	RTC Modes of Operation.....	379
20.2.5	Features .....	381
20.2.6	Modes of Operation .....	381
20.2.7	Block Diagram .....	382
20.3	External Signal Description .....	382
20.4	Register Definition .....	382
20.4.1	RTC Status and Control Register (RTCSC).....	383
20.4.2	RTC Counter Register (RTCCNT).....	384
20.4.3	RTC Modulo Register (RTCMOD) .....	384
20.5	Functional Description .....	384
20.5.1	RTC Source Clock Switching .....	385
20.5.2	RTC Operation Example.....	386
20.6	Initialization/Application Information .....	387

Section Number	Title	Page
	<b>Chapter 21 Serial Communications Interface (S08SCIV4)</b>	
21.1	Introduction .....	389
21.2	Module Configurations .....	389
	21.2.1 SCI Clock Gating.....	389
	21.2.2 Features .....	391
	21.2.3 Modes of Operation .....	391
	21.2.4 Block Diagram.....	392
21.3	Register Definition .....	394
	21.3.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL) .....	394
	21.3.2 SCI Control Register 1 (SCIxC1) .....	395
	21.3.3 SCI Control Register 2 (SCIxC2) .....	396
	21.3.4 SCI Status Register 1 (SCIxS1) .....	397
	21.3.5 SCI Status Register 2 (SCIxS2) .....	399
	21.3.6 SCI Control Register 3 (SCIxC3) .....	400
	21.3.7 SCI Data Register (SCIxD).....	401
21.4	Functional Description .....	401
	21.4.1 Baud Rate Generation .....	401
	21.4.2 Transmitter Functional Description .....	402
	21.4.3 Receiver Functional Description .....	403
	21.4.4 Interrupts and Status Flags.....	405
	21.4.5 Additional SCI Functions .....	406
	<b>Chapter 22 Serial Peripheral Interface (S08SPIV4)</b>	
22.1	Introduction .....	409
22.2	Module Configurations .....	409
	22.2.1 SPI Port Configuration Information .....	409
	22.2.2 SPI Clock Gating .....	409
	22.2.3 Features .....	411
	22.2.4 Block Diagrams .....	411
	22.2.5 SPI Baud Rate Generation .....	413
22.3	External Signal Description .....	414
	22.3.1 SPSCK — SPI Serial Clock.....	414
	22.3.2 MOSI — Master Data Out, Slave Data In .....	414
	22.3.3 MISO — Master Data In, Slave Data Out .....	414
	22.3.4 SS — Slave Select .....	414
22.4	Modes of Operation.....	415
	22.4.1 SPI in Stop Modes .....	415
22.5	Register Definition .....	415
	22.5.1 SPI Control Register 1 (SPIC1) .....	415
	22.5.2 SPI Control Register 2 (SPIC2) .....	416

Section Number	Title	Page
22.5.3	SPI Baud Rate Register (SPIBR).....	417
22.5.4	SPI Status Register (SPIS).....	418
22.5.5	SPI Data Register (SPID) .....	419
22.6	Functional Description .....	420
22.6.1	General.....	420
22.6.2	Master Mode .....	420
22.6.3	Slave Mode .....	421
22.6.4	SPI Clock Formats .....	422
22.6.5	Special Features .....	424
22.6.6	SPI Interrupts .....	426
22.6.7	Mode Fault Detection .....	426

## Chapter 23 Development Support

23.1	Introduction .....	427
23.1.1	Forcing Active Background.....	427
23.1.2	Module Configuration.....	427
23.1.3	Features .....	428
23.2	Background Debug Controller (BDC) .....	428
23.2.1	BKGD Pin Description .....	429
23.2.2	Communication Details .....	429
23.2.3	BDC Commands .....	433
23.2.4	BDC Hardware Breakpoint.....	435
23.3	Register Definition .....	435
23.3.1	BDC Registers and Control Bits .....	436
23.3.2	System Background Debug Force Reset Register (SBDFR).....	438

## Chapter 24 Debug Module (S08DBGV3) (64K)

24.1	Introduction .....	441
24.1.1	Features .....	441
24.1.2	Modes of Operation .....	442
24.1.3	Block Diagram.....	442
24.2	Signal Description .....	442
24.3	Memory Map and Registers .....	443
24.3.1	Module Memory Map .....	443
24.3.2	Register Descriptions .....	444
24.4	Functional Description .....	455
24.4.1	Comparator .....	455
24.4.2	Breakpoints .....	455
24.4.3	Trigger Selection .....	456
24.4.4	Trigger Break Control (TBC) .....	456

<b>Section Number</b>	<b>Title</b>	<b>Page</b>
24.4.5	FIFO.....	460
24.4.6	Interrupt Priority .....	461
24.5	Resets .....	461
24.6	Interrupts .....	461
24.7	Electrical Specifications .....	462

<b>Section Number</b>	<b>Title</b>	<b>Page</b>
-----------------------	--------------	-------------

# Chapter 1

## Device Overview

The MC9S08MP16 and MC9S08MP12 are members of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types.

### 1.1 Devices in the MC9S08MP16 Series

Table 1-1 summarizes the feature set available in the MC9S08MP16 Series of MCUs.

**Table 1-1. MC9S08MP16 Series Features by MCU and Package**

Feature	MC9S08MP16		MC9S08MP12
FLASH size (bytes)	16384		12288
RAM size (bytes)	1024		512
Pin quantity	48	32	28
ADC channels	13	12	8
CRC	yes		
DAC	3		
DBG	yes		
FTM1 channels	2		
FTM2 channels	6		
HSCMP	3		
ICS	yes		
IIC	yes		no
MTIM	yes		
KBI Pins	24	15	14
PDB	2		
PGA	yes		no
Pin I/O <sup>1</sup>	40	25	22
RTC	yes		
SCI	yes		
SPI	yes		
XOSC	yes		

<sup>1</sup> Port I/O count does not include the output-only pins PTF0/BKGD/MS and PTF1/RESET.

### 1.2 Features

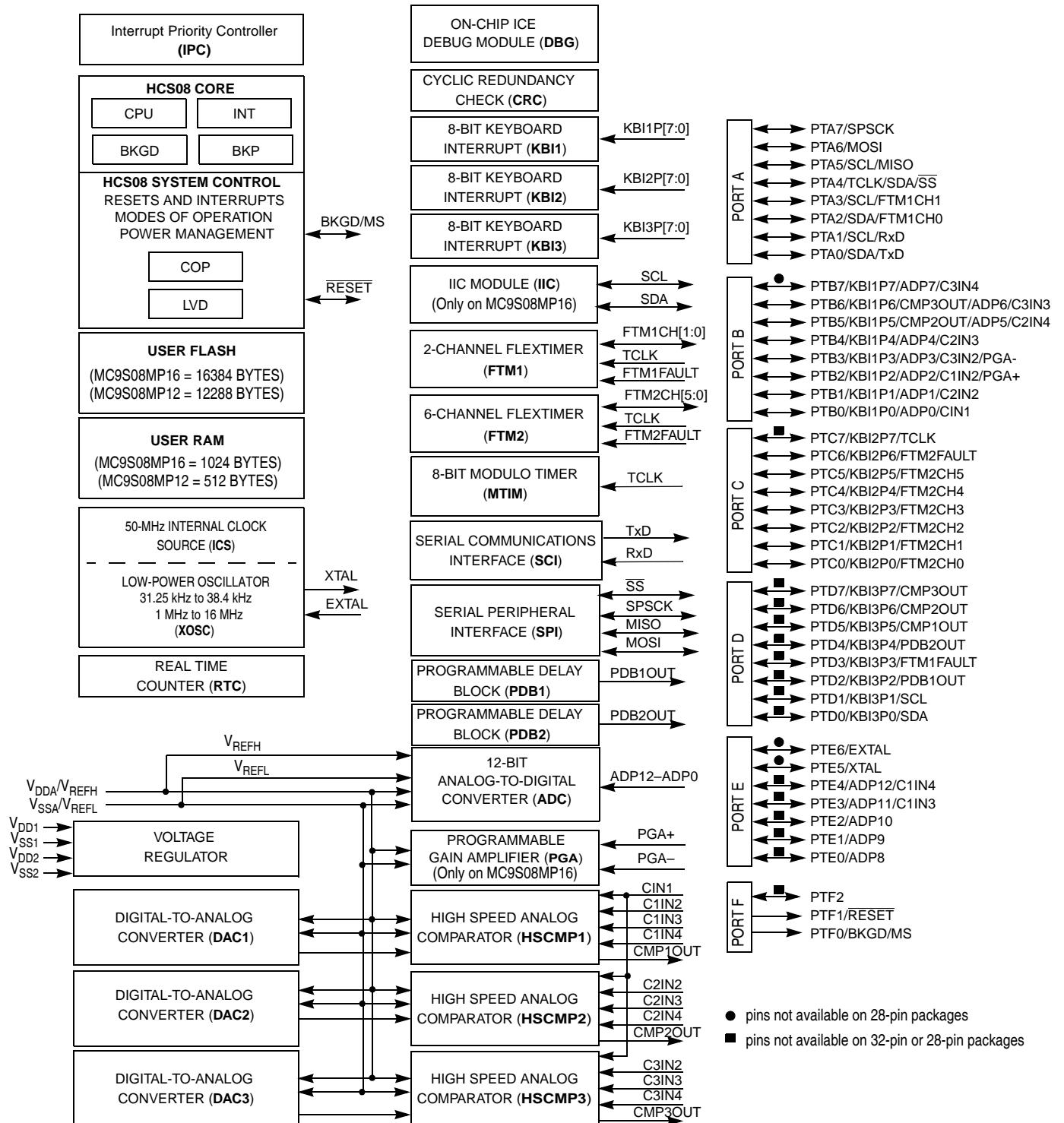
- 8-Bit S08 Central Processor Unit (CPU)
  - Up to 50-MHz CPU at 2.7V to 5.5V across temperature range of -40°C to 105°C
  - HC08 instruction set with an additional BGND instruction and additional addressing modes for LDHX and STHX

- Support for up to 48 interrupt/reset sources
- On-Chip Memory
  - Flash read/program/erase over full operating voltage and temperature
  - Random-access memory (RAM)
  - Security circuitry to prevent unauthorized access to RAM and FLASH contents
- Power-Saving Modes
  - Two low power stop modes; reduced power wait mode
  - Allows clock to remain enabled to RTC in all stop modes
- Clock Source Options
  - Oscillator (XOSC) supporting Crystal and ceramic resonator range of 31.25Khz to 38.4Khz (low range mode) or 1-16 Mhz (high range mode)
  - Internal Clock Source (ICS) — Containing a frequency-locked-loop (FLL) controlled by internal or external reference; precision trimming of internal reference allows 0.2% resolutions and 2% deviation over temperature and voltage; supports CPU frequencies up to 50MHz
- System Protection
  - Watchdog computer operating properly (COP) reset with option to run from dedicated 1-kHz internal clock source or bus clock
  - Low-voltage detection with reset or interrupt; selectable trip points
  - Illegal opcode detection with reset; Illegal address detection with reset
  - Flash block protection
- Development Support
  - Single-wire background debug interface
  - Breakpoint capability to allow single breakpoint setting during in-circuit debugging (plus three more breakpoints in the on-chip debug module)
  - On-chip, in-circuit emulation (ICE) debug module containing three comparators and nine trigger modes. Eight deep FIFO for storing change-of-flow address and event-only data. Debug module supports both tag and force breakpoints.
- Peripherals
  - **IPC** — Interrupt Priority Controller with 4 programmable interrupt priority levels
  - **ADC** — 13-channel, 12-bit resolution, 2.5  $\mu$ s conversion time, automatic compare function, 1.7 mV/ $^{\circ}$ C temperature sensor, internal bandgap reference channel; runs in stop3
  - **PGA** — Differential Programmable Gain Amplifier with programmable gain (x1, x2, x4, x8, x16, or x32)
  - **HSCMP** — Three fast analog comparators (HSCMP1, HSCMP2, and HSCMP3) with both positive and negative inputs; separately selectable interrupt on rising and falling comparator output; filtering; windowing; HSCMP1 and HSCMP2 outputs can be optionally routed to FTM1 module; runs in stop3
  - **DAC** — Three 5-bit digital to analog convertor used as a 32 tap voltage reference
  - **PDB** — Two programmable delay blocks
    - PDB1 synchronizes PWM with samples of ADC

- PDB2 synchronize PWM with comparing window of analog comparators
- PWM output synchronize with FTM PWM output.
- **SCI** — Full duplex non-return to zero (NRZ); LIN master extended break generation; LIN slave extended break detection; wake up on active edge
- **SPI** — Full-duplex or single-wire bidirectional; Double-buffered transmit and receive; Master or Slave mode; MSB-first or LSB-first shifting
- **IIC/SMBus** — Up to 400 kbps; Multi-master operation; Programmable slave address; Interrupt driven byte-by-byte data transfer; supports broadcast mode and 10-bit addressing; SMBus compatible
- **FTM** — two Flextimer modules with total of 8 channels
  - One 2-channel (FTM1) and one 6-channel (FTM2)
  - supports up to 50 MHz operation clock;
  - selectable input capture, output compare, edge- or center-aligned PWM; dead time insertion; fault inputs
  - Input edge interrupt capability on FTM1 inputs
- **MTIM** — 8-bit modulo counter with 8-bit prescaler and overflow interrupt
- **RTC** — (Real-time counter) 8-bit modulus counter with binary or decimal based prescaler; External clock source for precise time base, time-of-day, calendar or task scheduling functions; Free running on-chip low power oscillator (1 kHz) for cyclic wake-up without external components, runs in all MCU modes
- **CRC** — Cyclic redundancy check generator
- **KBI** — Three 8 channel keyboard interrupt module with software selectable polarity on edge or edge/level modes‘
- Input/Output
  - 40GPIOs, 2 output-only pins.
  - Hysteresis and configurable pull up device on all input pins; Configurable slew rate and drive strength on all output pins.
  - Sink/Source current up to 20mA
- Package Options
  - 48-LQFP, 32-LQFP, 28-SOIC

## 1.3 MCU Block Diagram

The block diagram in [Figure 1-1](#) shows the structure of the MC9S08MP16 Series MCU.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

$V_{DD2}$  pad is tied internally on 32-pin and 28-pin packages,

$V_{SS2}$  pad is tied internally on 28-pin packages

Figure 1-1. MC9S08MP16 Series Block Diagram

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

**Table 1-2. Module Versions**

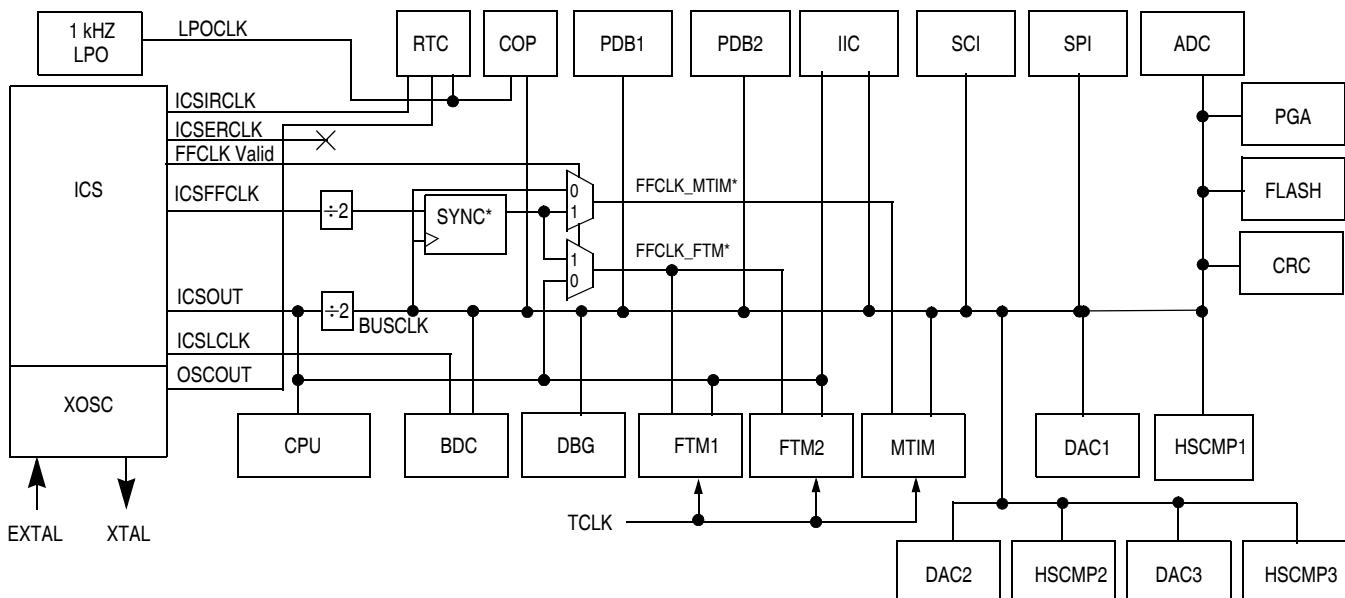
Module	Version
High Speed Analog Comparator (5V) (HSCMP)	1
Analog-to-Digital Converter (ADC)	1
Central Processor Unit (CPU)	5
Inter-Integrated Circuit (IIC)	3
Internal Clock Source (ICS)	3
Low Power Oscillator (XOSC)	1
Modulo Timer (MTIM)	1
On-Chip In-Circuit Emulator (DBG)	3
Real-Time Counter (RTC)	1
Serial Peripheral Interface (SPI)	4
Serial Communications Interface (SCI)	4
Flextimer (FTM)	2
Digital-to-Analog Converter (DAC)	1
Programmable Gain Amplifier (PGA)	1
Programmable Delay Block (PDB)	1
Cyclic Redundancy Check Generator (CRC)	3
Keyboard Interrupt (KBI)	2

## 1.4 System Clock Distribution

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

- ICSOUT — This clock source is used as the CPU clock (CPUCLK) and is divided by 2 to generate the peripheral bus clock (BUSCLK). 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 14, “Internal Clock Source \(S08ICSV3\),”](#) for details on configuring the ICSOUT clock.
- ICSLCLK — This clock source is derived from the digitally controlled oscillator, DCO, of the ICS when the ICS is configured to run off the internal or external reference clock. Development tools can select this internal self-coded source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICSEERCLK — This is the external reference clock. [Chapter 14, “Internal Clock Source \(S08ICSV3\),”](#) explains the ICSEERCLK in more detail. This clock source is not used on MC9S08MP16 Series MCUs.

- ICSIRCLK — This is the internal reference clock and can be selected as the real-time counter clock source. [Chapter 14, “Internal Clock Source \(S08ICSV3\)”](#) explains the ICSIRCLK in more detail. See [Chapter 20, “Real-Time Counter \(S08RTCV1\),”](#) for more information regarding the use of ICSIRCLK.
  - ICSFFCLK — This generates the fixed frequency clock (FFCLK) after being synchronized to the bus clock. FFCLK can be selected as a clock source for the FTM and MTIM modules (referred to as XCLK in the MTIM chapter). The ICSFFCLK frequency is determined by the ICS settings. See the “Fixed Frequency Clock” section in [Chapter 14, “Internal Clock Source \(S08ICSV3\).”](#) The fixed frequency clock (FFCLK) must not exceed one half of the bus clock frequency. In clock configurations where this requirement is not met, the MTIM is clocked at the bus frequency and the FTM is clocked at the ICSOUT frequency.
  - LPOCLK — This clock is generated from an internal low power oscillator completely independent of the ICS module. The LPOCLK can be selected as the clock source to the RTC or COP modules. To use the LPOCLK with these modules, see [Chapter 20, “Real-Time Counter \(S08RTCV1\),”](#) and [Section 5.4, “Computer Operating Properly \(COP\) Watchdog.”](#)
  - OSCOUT — This is the output of the XOSC module and can be selected as the RTC clock source.
  - TCLK — TCLK is the optional external clock source for the FTM and MTIM modules. The TCLK must be limited to 1/4 the frequency of the bus clock for synchronization. TCLK is referenced as EXTCLK in FTM chapters. If only FTM is using this optional clock, then TCLK must be limited to one-fourth the frequency of the ICSOUT clock.



\* The fixed frequency clock (FFCLK) is internally synchronized to the bus clock and must not exceed one half of the bus clock frequency. In clock configurations where this requirement is not met, the modules operate at maximum frequency (MTIM @ bus clock frequency and FTM @ 2x bus clock frequency).

ADC and PGA have min and max frequency requirements.

FLASH has frequency requirements for program and erase operation.

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





# **Chapter 2**

## **Pins and Connections**

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

### **2.1 Device Pin Assignment**

[Figure 2-1](#) - [Figure 2-3](#) shows the pin assignments for the MC9S08MP16 Series devices.

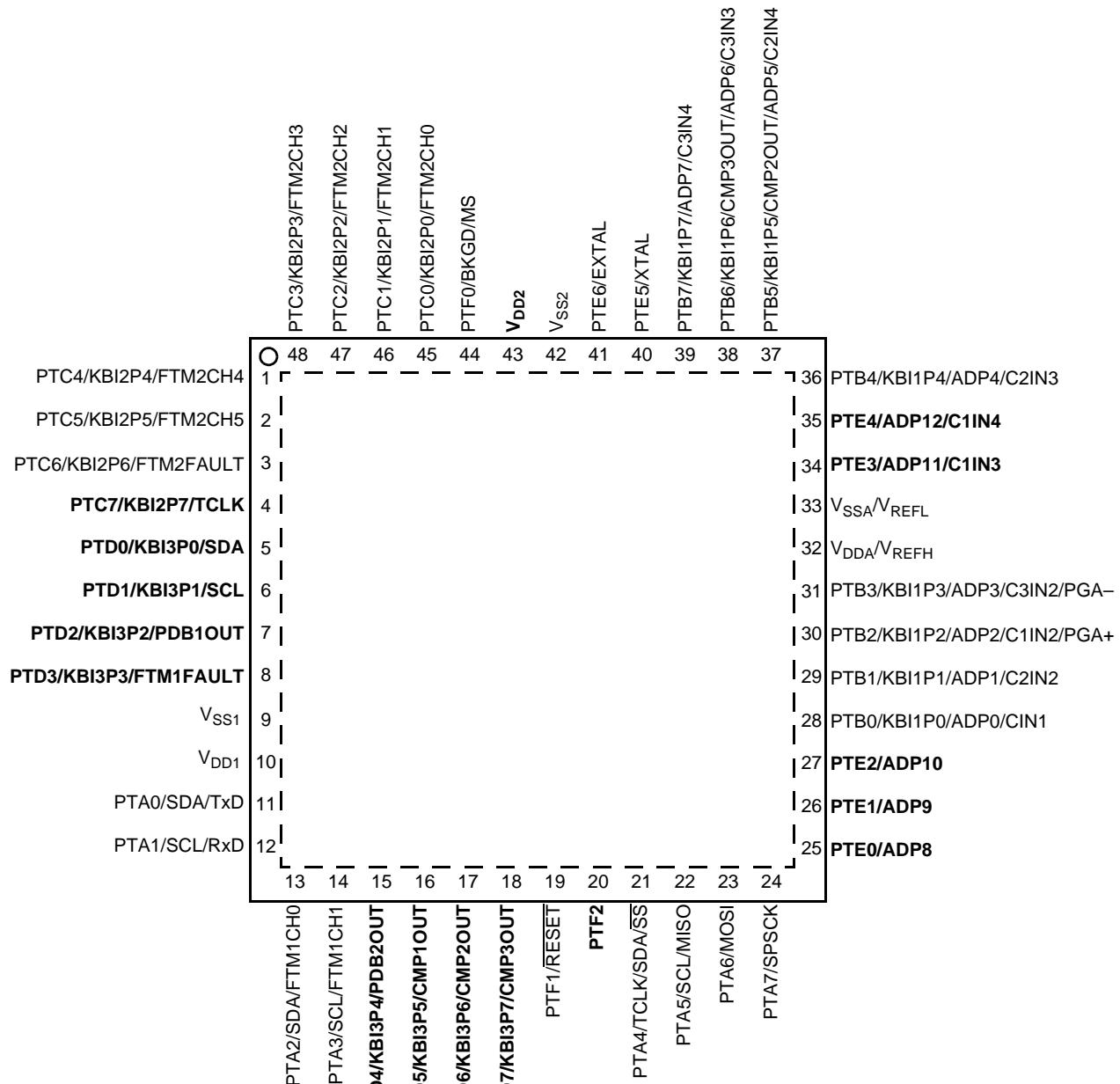
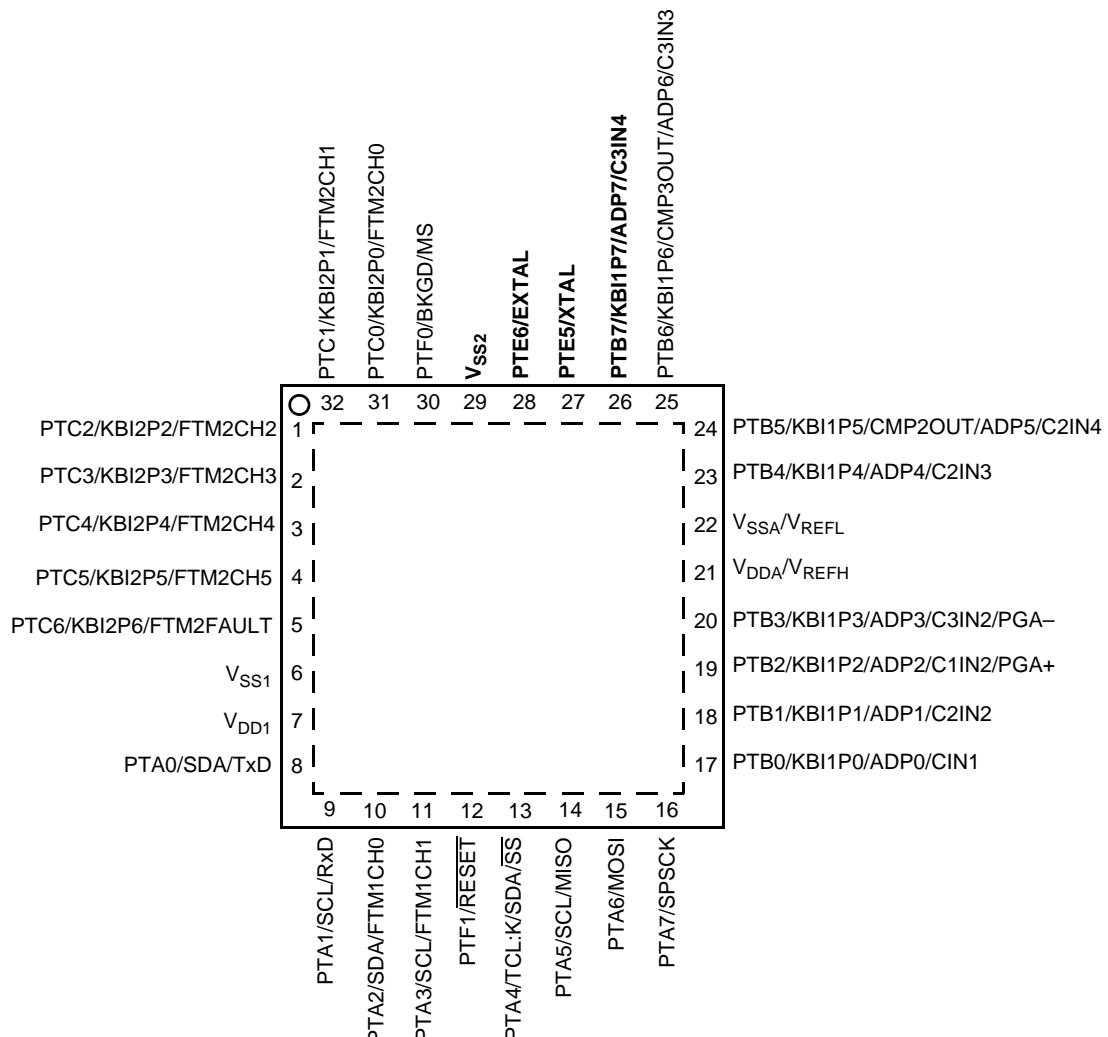


Figure 2-1. MC9S08MP16 Series in 48-Pin LQFP Package



Pins in **bold** are lost in the next lower pin count package.

Figure 2-2. MC9S08MP16 Series in 32-Pin LQFP Package

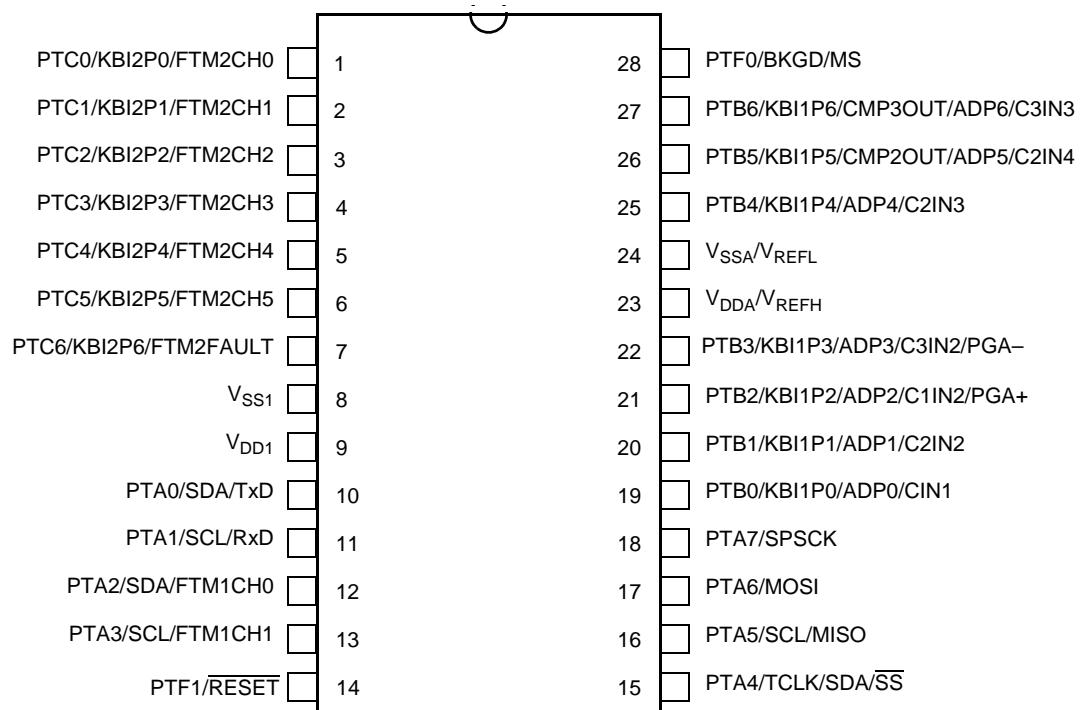


Figure 2-3. MC9S08MP16 Series in 28-Pin SOIC Package

## 2.2 Recommended System Connections

Figure 2-4 shows pin connections that are common to MC9S08MP16 Series application systems.

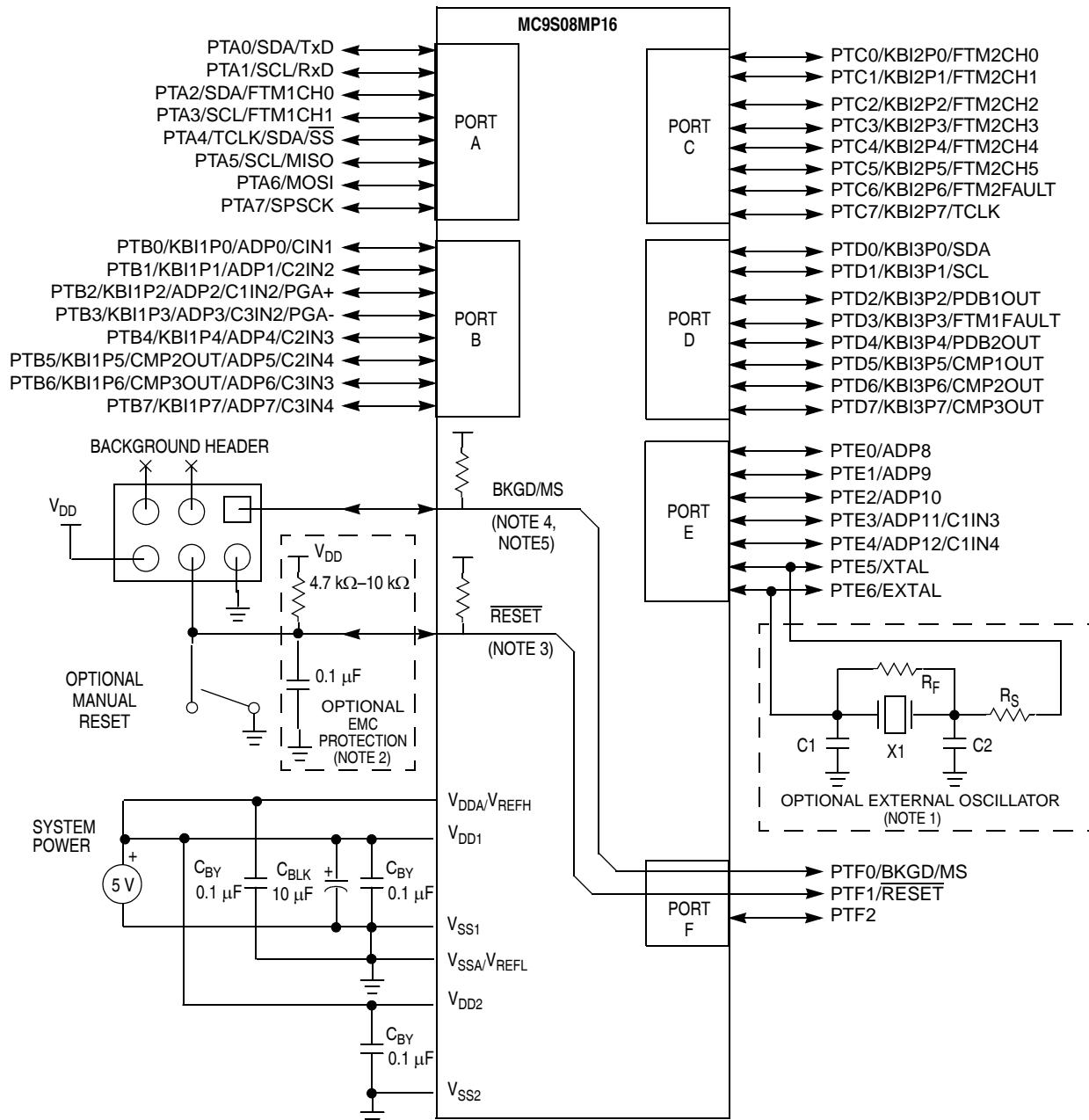


Figure 2-4. Basic System Connections

## 2.2.1 Power

$V_{DD1}$ ,  $V_{DD2}$ ,  $V_{SS1}$  and  $V_{SS2}$  are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides regulated lower-voltage source for the CPU and other internal circuitry of the MCU.

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

$V_{DDA}$  and  $V_{SSA}$  are the analog power supply pins for MCU. This voltage source supplies power to the ADC, PGA, HSCMPs, and DACs modules. A 0.1uF ceramic bypass capacitor should be located as near to the MCU power pins as practical to suppress high-frequency noise. The  $V_{REFH}$  and  $V_{REFL}$  pins are the voltage reference high and voltage reference low inputs, respectively for the ADC module. For this MCU,  $V_{DDA}$  shares the  $V_{REFH}$  pin. For this MCU,  $V_{SSA}$  shares the  $V_{REFL}$  pin.

## 2.2.2 Oscillator (XOSC)

Immediately after reset, the MCU uses an internally generated clock provided by the clock source generator (ICS) module. For more information on the ICS, see [Chapter 14, “Internal Clock Source \(S08ICSV3\).”](#)

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

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

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

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

## 2.2.3 RESET Pin

The external pin reset function is shared with an output-only port function on the PTF1/RESET. The reset function is enabled when RSTPE in SOPT1 is set. RSTPE is set following any reset of the MCU and must be cleared in order to use this pin as an output-only port pin. When enabled as the RESET pin (RSTPE = 1), an internal pullup device is automatically enabled.

**RESET** is an open-drain drive containing an internal pull-up device. Internal power-on reset and low-voltage reset circuitry typically make external reset circuitry unnecessary. This pin is normally connected to the standard 6-pin background debug connector so a development system can directly reset the MCU system. If desired, a manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

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

#### NOTE

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

The voltage measured on the internally pulled up **RESET** pin will not be pulled to  $V_{DD}$ . The internal gates connected to this pin are pulled to  $V_{DD}$ . If the **RESET** pin is required to drive to a  $V_{DD}$  level an external pullup should be used.

#### NOTE

In EMC-sensitive applications, an external RC filter is recommended on the **RESET**. See [Figure 2-4](#) for an example.

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

During a power-on-reset (POR) or background debug force reset (see [Section 5.8.2, “System Background Debug Force Reset Register \(SBDFR\)](#),” for more information), the PTF0/BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin (**BKGDP** = 1), an internal pullup device is automatically enabled, the output drive strength is configured for high drive, and the output slew rate is disabled.

The background debug communication function is enabled when **BKGDP** in SOPT1 is set. **BKGDP** is set following any reset of the MCU and must be cleared to use the PTF0/BKGD/MS pin’s alternative pin functions.

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

The BKGD pin is used primarily with background debug controller (BDC) communications, and features a custom protocol that uses 16 clock cycles of the target MCU’s BDC clock per bit time. The target MCU’s BDC clock can run as fast as the bus clock, so no significant capacitance should 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

cables and the absolute value of the internal pullup device play a minimal role in determining rise and fall times on the BKGD pin.

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

The MC9S08MP16 Series of MCUs support up to 40 general-purpose I/O pins and 2 output-only pins which are shared with on-chip peripheral functions (timers, serial I/O, ADC, HSCMP, etc.). The GPIO output-only pins (PTF0/BKGD/MS and PTF1/RESET) are bi-directional when configured as BKGD and RESET, respectively.

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

PTF1 is a special case I/O pin. When the PTF1/RESET pin is configured as PTF1, it is configured as open-drain and can only drive out low. When PTFD1 bit is set, the internal pullup is controlled via the PTFPE1 bit.

### NOTE

The voltage measured on the internally pulled up PTF1 pin will not be pulled to  $V_{DD}$ . If the PTF1 pin is required to drive to a  $V_{DD}$  level an external pullup should be used.

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

### NOTE

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

When using 32-pin devices, the user must either enable on-chip pullup devices or change the direction of non-bonded out PTF2, PTE4-PTE0, PTD7-PTD0, and PTC7 pins to outputs so the pins do not float.

When using 28-pin devices, the user must either enable on-chip pullup devices or change the direction of non-bonded out PTF2, PTE6-PTE0, PTD7-PTD0, PTC7 and PTB7 pins to outputs so the pins do not float.

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

Pin Number			<-- Lowest Priority --> Highest				
48 LQFP	32 LQFP	28 SOIC	Port Pin	Alt 1	Alt 2	Alt3	Alt4
1	3	5	PTC4	KBI2P4	FTM2CH4		
2	4	6	PTC5	KBI2P5	FTM2CH5		
3	5	7	PTC6	KBI2P6	FTM2FAULT		
4	—	—	PTC7	KBI2P7	TCLK <sup>1</sup>		
5	—	—	PTD0	KBI3P0	SDA <sup>5</sup>		
6	—	—	PTD1	KBI3P1	SCL <sup>5</sup>		
7	—	—	PTD2	KBI3P2	PDB1OUT		
8	—	—	PTD3	KBI3P3	FTM1FAULT		
9	6	8					V <sub>SS1</sub>
10	7	9					V <sub>DD1</sub>
11	8	10	PTA0	SDA <sup>5</sup>	TxD		
12	9	11	PTA1	SCL <sup>5</sup>	RxD		
13	10	12	PTA2	SDA <sup>5</sup>	FTM1CH0		
14	11	13	PTA3	SCL <sup>5</sup>	FTM1CH1		
15	—	—	PTD4	KBI3P4	PDB2OUT		
16	—	—	PTD5	KBI3P5	CMP1OUT		
17	—	—	PTD6	KBI3P6	CMP2OUT <sup>2</sup>		
18	—	—	PTD7	KBI3P7	CMP3OUT <sup>3</sup>		
19	12	14	PTF1	RESET <sup>4</sup>			
20	—	—	PTF2				
21	13	15	PTA4	TCLK <sup>1</sup>	SDA <sup>5</sup>	SS	
22	14	16	PTA5		SCL <sup>5</sup>	MISO	
23	15	17	PTA6			MOSI	
24	16	18	PTA7			SPSCK	
25	—	—	PTE0		ADP8		
26	—	—	PTE1		ADP9		
27	—	—	PTE2		ADP10		
28	17	19	PTB0	KBI1P0	ADP0 <sup>6</sup>	CIN1 <sup>6</sup>	
29	18	20	PTB1	KBI1P1	ADP1 <sup>6</sup>	C2IN2 <sup>6</sup>	
30	19	21	PTB2	KBI1P2	ADP2 <sup>6</sup>	C1IN2 <sup>6</sup>	PGA+ <sup>6</sup>
31	20	22	PTB3	KBI1P3	ADP3 <sup>6</sup>	C3IN2 <sup>6</sup>	PGA- <sup>6</sup>
32	21	23					V <sub>DDA</sub> /V <sub>REFH</sub>

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

Pin Number			Port Pin	<-- Lowest Priority	--> Highest Priority		
48 LQFP	32 LQFP	28 SOIC		Alt 1	Alt 2	Alt3	Alt4
33	22	24					$V_{SSA}/V_{REFL}$
34	—	—	PTE3		ADP11 <sup>6</sup>	C1IN3 <sup>6</sup>	
35	—	—	PTE4		ADP12 <sup>6</sup>	C1IN4 <sup>6</sup>	
36	23	25	PTB4	KBI1P4		ADP4 <sup>6</sup>	C2IN3 <sup>6</sup>
37	24	26	PTB5	KBI1P5	CMP2OUT <sup>2</sup>	ADP5 <sup>6</sup>	C2IN4 <sup>6</sup>
38	25	27	PTB6	KBI1P6	CMP3OUT <sup>3</sup>	ADP6 <sup>6</sup>	C3IN3 <sup>6</sup>
39	26	—	PTB7	KBI1P7		ADP7 <sup>6</sup>	C3IN4 <sup>6</sup>
40	27	—	PTE5	XTAL			
41	28	—	PTE6	EXTAL			
42	29	—					$V_{SS2}$
43	—	—					$V_{DD2}$
44	30	28	PTF0	BKGD	MS		
45	31	1	PTC0	KBI2P0	FTM2CH0		
46	32	2	PTC1	KBI2P1	FTM2CH1		
47	1	3	PTC2	KBI2P2	FTM2CH2		
48	2	4	PTC3	KBI2P3	FTM2CH3		

<sup>1</sup> TCLK pin can be repositioned using TCLKPS in SOPT2. Default reset location is PTC7.

<sup>2</sup> HSCMP2 output CMP2OUT can be repositioned using the CMP2OPS in the SOPT2 register. Default reset location is PTD6.

<sup>3</sup> HSCMP3 output CMP3OUT can be repositioned using the CMP3OPS in the SOPT2 register. Default reset location is PTD7.

<sup>4</sup> Pin is open drain with an internal pullup that is always enabled. Pin does not contain a clamp diode to  $V_{DD}$  and should not be driven above  $V_{DD}$ . The voltage measured on the internally pulled up  $\overline{RESET}$  will not be pulled to  $V_{DD}$ . The internal gates connected to this pin are pulled to  $V_{DD}$ .

<sup>5</sup> IIC pins SDA and SCL can be repositioned using IICPS in SOPT2. Default reset locations are PTD0 and PTD1.

<sup>6</sup> If ADC, HSCMP, or PGA is enabling a shared analog input pin, each will have access to the pin.

# Chapter 3

## Modes of Operation

### 3.1 Introduction

The operating modes of the MC9S08MP16 Series are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

### 3.2 Features

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

### 3.3 Run Mode

This is the normal operating mode for the MC9S08MP16 Series. This mode is selected upon the MCU exiting reset if the BKGD/MS pin is high. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE–0xFFFF after reset.

### 3.4 Active Background Mode

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

Active background mode is entered in any of the following ways:

- When the BKGD/MS pin is low during POR or immediately after issuing a background debug force reset (see [Section 5.8.2, “System Background Debug Force Reset Register \(SBDFR\)”](#))
- When a BACKGROUND command is received through the BKGD/MS pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

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

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD/MS pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:

- Memory access commands
- Memory-access-with-status commands
- BDC register access commands
- The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08MP16 Series is shipped from the Freescale Semiconductor factory, the FLASH program memory is erased by default unless specifically noted so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to the [Development Support](#) chapter.

## 3.5 Wait Mode

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

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

## 3.6 Stop Modes

One of two stop modes (stop2 or stop3) is entered upon execution of a STOP instruction when STOPE bit in system option 1 register (SOPT1) is set. In both stop modes, the bus and CPU clocks are halted. In stop3 the regulator is in standby. In stop2 the regulator is in partial powerdown. The ICS module can be configured to leave the reference clocks running. See [Chapter 14, “Internal Clock Source \(S08ICSV3\),”](#) for more information.

If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU will not enter either stop mode and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in the System Power Management Status and Control 2 Register (SPMSC2).

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

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

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

<sup>1</sup> ENBDM is located in the BDCSCR, which is only accessible through BDC commands, see [Section 23.3.1.1, “BDC Status and Control Register \(BDCSCR\)”](#).

<sup>2</sup> When in Stop3 mode with BDM enabled, The S<sub>IDD</sub> will be near R<sub>IDD</sub> levels because internal clocks are enabled.

### 3.6.1 Stop3 Mode

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

Stop3 can be exited by asserting RESET, or by an interrupt from one of the following sources: the real-time counter (RTC), LVD system, HSCMP1, HSCMP2, HSCMP3, ADC, SCI, KBI1, KBI2, or KBI3.

If stop3 is exited by means of the RESET pin, then the MCU is reset and operation will resume after taking the reset vector.. Exit by means of one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

#### NOTE

IPC must be configured to allow all enabled asynchronous modules to an interrupt priority level (ILRx) equal to or greater than the interrupt priority mask level (IPM). If not configured properly, system clocks may begin operation without CPU waking from stop.

#### 3.6.1.1 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. For configuring the LVD system for interrupt or reset, refer to [Section 5.6, “Low-Voltage Detect \(LVD\) System”](#). If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) the voltage regulator remains active during stop mode. If the user attempts to enter stop2 with the LVD enabled for stop, the MCU will instead enter stop3.

For the ADC to operate in stop mode, the LVD must be enabled when entering stop3.

#### 3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in [Chapter 23, “Development Support.”](#) If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible.

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

In addition, the ICS and XOSC continue operation in the clock configurations set prior to stop entry and the voltage regulator does not enter its low-power standby state but maintains full internal regulation.

### 3.6.2 Stop2 Mode

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

Exit from stop2 is performed by asserting the wake-up pin (using an analog input path from PTF1/ $\overline{\text{RESET}}$ ) on the MCU.

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

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

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

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

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

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

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

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

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

Peripheral	Mode	
	Stop2	Stop3
ADC	Off	Optionally On <sup>1</sup>
BDM	Off <sup>2</sup>	Optionally On
CPU	Off	Standby
CRC	Off	Standby
FLASH	Off	Standby
HSCMP1, HSCMP2, HSCMP3	Off	Optionally On <sup>3</sup>
ICS	Off	Optionally On <sup>4</sup>
IPC	Off	Standby
IIC	Off	Standby
LVD/LVW	Off <sup>5</sup>	Optionally On
MTIM	Off	Standby
Parallel Port Registers	Off	Standby
PDB1, PDB2	Off	Standby
PGA	Off	Off
DAC1, DAC2, DAC3	Off	Optionally On
RAM	Standby	Standby
RTC	Optionally On	Optionally On
SCI	Off	Standby
SPI	Off	Standby
FTM1, FTM2	Off	Standby
Voltage Regulator	Standby	Optionally On <sup>6</sup>
XOSC	Off	Optionally On <sup>7</sup>
I/O Pins	States Held	States Held

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

<sup>2</sup> If ENBDM is set when entering stop2, the MCU will actually enter stop3.

<sup>3</sup> HSCMPx blocks must not be configured for windowing or filtering

<sup>4</sup> IRCLKEN and IREFSTEN set in ICSC1, else in standby.

<sup>5</sup> If LVDSE is set when entering stop2, the MCU will actually enter stop3.

<sup>6</sup> Voltage regulator will be on if BDM is enabled or if LVD is enabled when entering stop3.

<sup>7</sup> ERCLKEN and EREFSTEN set in ICSC2, else in standby. For high frequency range (RANGE in ICSC2 set) requires the LVD to also be enabled in stop3.



# Chapter 4

## Memory

### 4.1 MC9S08MP16 Series Memory Map

As shown in [Figure 4-1](#), on-chip memory in the MC9S08MP16 Series MCU consists of RAM, FLASH program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x008F)
- High-page registers (0x1800 through 0x18BF)
- Nonvolatile registers (0xFFB0 through 0xFFFF)

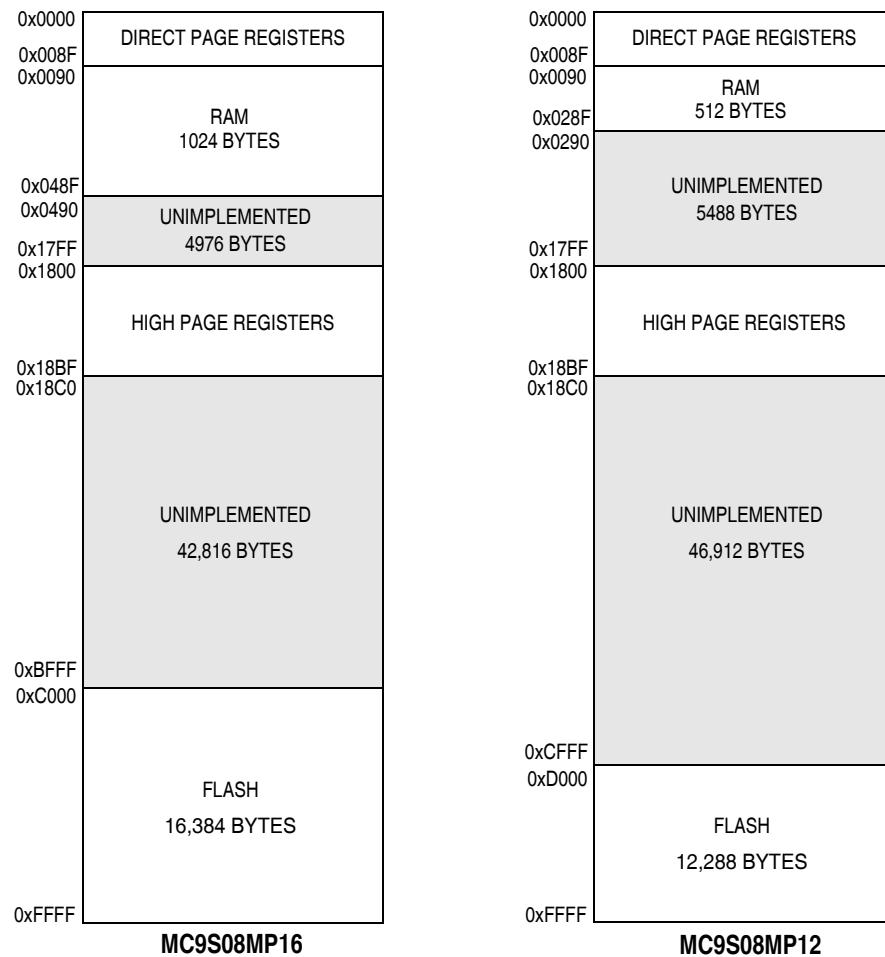


Figure 4-1. MC9S08MP16 Series Memory Maps

## 4.2 Reset and Interrupt Vector Assignments

**Table 4-1** shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale Semiconductor provided equate file for the MC9S08MP16 Series.

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

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1	Reserved	—
0xFFC2:0xFFC3	KBI3 Interrupt	Vkeyboard3
0xFFC4:0xFFC5	KBI2 Interrupt	Vkeyboard2
0xFFC6:0xFFC7	KBI1 Interrupt	Vkeyboard1
0xFFC8:0xFFC9	HSCMP3	Vhscmp3
0xFFCA:0xFFCB	HSCMP2	Vhscmp2
0xFFCC:0xFFCD	HSCMP1	Vhscmp1
0xFFCE:0xFFCF	RTC	Vrtc
0xFFD0:0xFFD1	IIC	Viic
0xFFD2:0xFFD3	SCI Transmit	Vscitx
0xFFD4:0xFFD5	SCI Receive	Vscirx
0xFFD6:0xFFD7	SCI Error	Vscierr
0xFFD8:0xFFD9	SPI	Vspi
0xFFDA:0xFFDB	ADC Conversion	Vadc
0xFFDC:0xFFDD	PDB2	Vpdb2
0xFFDE:0xFFDF	PDB1	Vpdb1
0FFE0:0FFE1	MTIM Overflow	Vmtim
0FFE2:0FFE3	FTM2 Overflow	Vftm2ovf
0FFE4:0FFE5	FTM2 Channel 5	Vftm2ch5
0FFE6:0FFE7	FTM2 Channel 4	Vftm2ch4
0FFE8:0FFE9	FTM2 Channel 3	Vftm2ch3
0FFEAE:0FFEBC	FTM2 Channel 2	Vftm2ch2
0FFEC:0FFED	FTM2 Channel 1	Vftm2ch1
0FFEE:0FFEF	FTM2 Channel 0	Vftm2cho
0FFF0:0FFF1	FTM1 Overflow	Vftm1ovf
0FFF2:0FFF3	FTM1 Channel 1	Vftm1ch1
0FFF4:0FFF5	FTM1 Channel 0	Vftm1ch0
0FFF6:0FFF7	FTM2 Fault	Vftm2fault
0FFF8:0FFF9	FTM1 Fault	Vftm1fault
0FFFA:0FFFB	Low Voltage Warning	Vlvw
0FFFC:0FFFD	SWI	Vswi
0FFF8:0FFFF	Reset	Vreset

## 4.3 Register Addresses and Bit Assignments

The registers in the MC9S08MP16 Series are divided into these groups:

- Direct-page registers are located in the first 144 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and RAM.
- The nonvolatile register area consists of reserved locations in flash memory at 0xFFAE–0xFFBF. Reserved flash locations include:
  - Two locations reserved for ICS trim values
  - NVPROT and NVOPT are loaded into working registers at reset
  - An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory

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

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

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

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	PTCD	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006	PTDD	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
0x0007	PTDDD	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
0x0008	PTED	0	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x0009	PTEDD	0	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x000A	PTFD	0	0	0	0	0	PTFD2	PTFD1	PTFD0
0x000B	PTFDD	0	0	0	0	0	PTFDD2	PTFDD1	PTFDD0
0x000C	KBI1SC	0	0	0	0	KBF	KBACK	BKIE	KBIMOD
0x000D	KBI1PE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x000E	KBI1ES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG
0x000F	Reserved	0	0	0	0	0	0	0	0
0x0010	ADCSC1	COCO	AIEN	ADCO					ADCH
0x0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	—	—	—	—
0x0012	ADCRH	0	0	0	0	ADR11	ADR10	ADR9	ADR8
0x0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	ADCVH	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8
0x0015	ADCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCCFG	ADLPC		ADIV	ADLSMP		MODE		ADICLK
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	APCTL2	—	—	—	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x0019	DAC1CTRL	DACEN	—	0					VOSEL
0x001A	DAC2CTRL	DACEN	—	0					VOSEL
0x001B	DAC3CTRL	DACEN	—	0					VOSEL
0x001C	KBI2SC	0	0	0	0	KBF	KBACK	BKIE	KBIMOD
0x001D	KBI2PE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x001E	KBI2ES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG
0x001F	Reserved	0	0	0	0	0	0	0	0
0x0020	IICA1	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x0021	IICF	MULT							ICR
0x0022	IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x0023	IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x0024	IICD								DATA
0x0025	IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x0026	IICSM	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF	0	0
0x0027	IICA2	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	IICSLTH	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
0x0029	IICSLTL	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0
0x002A	IICFLT	0	0	0	0	0	0	0	FLT
0x002B	Reserved	0	0	0	0	0	0	0	0
0x002C	KBI3SC	0	0	0	0	KBF	KBACK	BKIE	KBIMOD
0x002D	KBI3PE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x002E	KBI3ES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG
0x002F	Reserved	0	0	0	0	0	0	0	0
0x0030	FTM1SC	TOF	TOIE	CPWMS	CLKS		PS		
0x0031	FTM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0032	FTM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0033	FTM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0034	FTM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0035	FTM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0036	FTM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0037	FTM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0038	FTM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0039	FTM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x003A	FTM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x003B	Reserved	0	0	0	0	0	0	0	0
0x003C	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x003D	ICSC2	BDIV		RANGE	HGO	LP	EREFs	ERCLKEN	EREFSTEN
0x003E	ICSTRM	TRIM							
0x003F	ICSSC	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
0x0040	FTM1CNTINH	Bit 15	14	13	12	11	10	9	Bit 8
0x0041	FTM1CNTINL	Bit 7	6	5	4	3	2	1	Bit 0
0x0042	FTM1STATUS	0	0	0	0	0	0	CH1F	CH0F
0x0043	FTM1MODE	FAULTIE	FAULTM		CAPTTTEST	PWMSYNC	WPDIS	INIT	FTMEN
0x0044	FTM1SYNC	SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
0x0045	FTM1OUTINIT	0	0	0	0	0	0	CH1OI	CH0OI
0x0046	FTM1OUTMASK	0	0	0	0	0	0	CH1OM	CH0OM
0x0047	FTM1COMBINE0	0	FAULTEN	SYNCEN	DTEN	0	0	COMP	COMBINE
0x0048	FTM Reserved	0	0	0	0	0	0	0	0
0x0049	FTM Reserved	0	0	0	0	0	0	0	0
0x004A	FTM Reserved	0	0	0	0	0	0	0	0
0x004B	FTM1DEADTIME	DTPS		DTVAl					
0x004C	FTM1EXTTRIG	—	INITTRIGE N	—	—	—	—	—	—
0x004D	FTM1POL	0	0	0	0	0	0	POL1	POLO
0x004E	FTM1FMS	FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0							
0x004F	<b>FTM1FILTER0</b>	CH1FVAL				CH0FVAL										
0x0050	FTM Reserved	0	0	0	0	0	0	0	0							
0x0051	<b>FTM1FLTFILTER</b>	0	0	0	0	FFVAL										
0x0052	<b>FTM1FLTCTRL</b>	0	0	0	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN							
0x0053	Reserved	0	0	0	0	0	0	0	0							
0x0054	<b>MTIMSC</b>	TOF	TOIE	TRST	TSTP	0	0	0	0							
0x0055	<b>MTIMCLK</b>	0	0	CLKS		PS										
0x0056	<b>MTIMCNT</b>	CNT														
0x0057	<b>MTIMMOD</b>	MOD														
0x0058	<b>SPIC1</b>	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE							
0x0059	<b>SPIC2</b>	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0							
0x005A	<b>SPIBR</b>	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0							
0x005B	<b>SPIS</b>	SPRF	0	SPTEF	MODF	0	0	0	0							
0x005C	Reserved	0	0	0	0	0	0	0	0							
0x005D	<b>SPID</b>	Bit 7	6	5	4	3	2	1	Bit 0							
0x005E	<b>IPCSC</b>	IPCE	0	PSE	PSF	PULIPM	0	IPM								
0x005F	<b>IPMPS</b>	IPM3		IPM2		IPM1		IPM0								
0x0060	<b>ILRS0</b>	ILR3		ILR2		ILR1		ILR0								
0x0061	<b>ILRS1</b>	ILR7		ILR6		ILR5		ILR4								
0x0062	<b>ILRS2</b>	ILR11		ILR10		ILR9		ILR8								
0x0063	<b>ILRS3</b>	ILR15		ILR14		ILR13		ILR12								
0x0064	<b>ILRS4</b>	ILR19		ILR18		ILR17		ILR16								
0x0065	<b>ILRS5</b>	ILR23		ILR22		ILR21		ILR20								
0x0066	<b>ILRS6</b>	ILR27		ILR26		ILR25		ILR24								
0x0067	<b>ILRS7</b>	ILR31		ILR30		ILR29		ILR28								
0x0068	<b>SCIBDH</b>	LBKDI	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8							
0x0069	<b>SCIBDL</b>	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0							
0x006A	<b>SCIC1</b>	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT							
0x006B	<b>SCIC2</b>	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK							
0x006C	<b>SCIS1</b>	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF							
0x006D	<b>SCIS2</b>	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF							
0x006E	<b>SCIC3</b>	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE							
0x006F	<b>SCID</b>	Bit 7	6	5	4	3	2	1	Bit 0							
0x0070	<b>PDB1CTRL1</b>	LDMOD	BOS		ENB	AOS		ENA	LDOCK							
0x0071	<b>PDB1CTRL2</b>	PRESCALER			TRIGSEL			CONT	SWTRIG							
0x0072	<b>PDB1DLYAH</b>	DELAYA[15:8]														
0x0073	<b>PDB1DLYAL</b>	DELAYA[7:0]														
0x0074	<b>PDB1DLYBH</b>	DELAYB[15:8]														
0x0075	<b>PDB1DLYBL</b>	DELAYB[7:0]														
0x0076	<b>PDB1MODH</b>	MOD[15:8]														

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0						
0x0077	PDB1MODL								MOD[7:0]						
0x0078	PDB1COUNTH								COUNT[15:8]						
0x0079	PDB1COUNTL								COUNT[7:0]						
0x007A	PDB1SCR	PADEN	PDBEN	COF	COIE	DBF	DBIE	DAF	DAIE						
0x007B	Reserved	0	0	0	0	0	0	0	0						
0x007C	CRCH	Bit 15	14	13	12	11	10	9	Bit 8						
0x007D	CRCL	Bit 7	6	5	4	3	2	1	Bit 0						
0x007E	TRANSPOSE	Bit 7	6	5	4	3	2	1	Bit 0						
0x007F	Reserved	0	0	0	0	0	0	0	0						
0x0080	PDB2CTRL1	LDMOD	—	—	—	—	—	—	LDOCK						
0x0081	PDB2CTRL2	PRESCALER			TRIGSEL			CONT	SWTRIG						
0x0082	PDB2DLYAH	DELAYA[15:8]													
0x0083	PDB2DLYAL	DELAYA[7:0]													
0x0084	PDB2DLYBH	DELAYB[15:8]													
0x0085	PDB2DLYBL	DELAYB[7:0]													
0x0086	PDB2MODH	MOD[15:8]													
0x0087	PDB2MODL	MOD[7:0]													
0x0088	PDB2COUNTH	COUNT[15:8]													
0x0089	PDB2COUNTL	COUNT[7:0]													
0x008A	PDB2SCR	PADEN	PDBEN	COF	COIE	DBF	DBIE	DAF	DAIE						
0x008B	Reserved	0	0	0	0	0	0	0	0						
0x008C	RTCSC	RTIF	RTCLKS		RTIE	RTCPS									
0x008D	RTCCNT	RTCCNT													
0x008E	RTCMOD	RTCMOD													
0x008F	Reserved	0	0	0	0	0	0	0	0						

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

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPT		STOPE	SPIFE	ACIC2	ACIC1	BKGDP	RSTPE
0x1803	SOPT2	COPCLKS	COPW	FTM2T2S	IICPS		TCLKPS	CMP3OPS	CMP2OPS
0x1804 – 0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	Reserved	—	—	—	—	—	—	—	—
0x1809	SPMSC1	LVWF	LVWACK	LVWIE	LVDRE	LVDSE	LVDE	0	BGBE
0x180A	SPMSC2	0	0	LVDV	LVWV	PPDF	PPDACK	—	PPDC
0x180B	SOPT3	0	0	0	0	0	0	0	SYNCFTM
0x180C – 0x180D	Reserved	—	—	—	—	—	—	—	—
0x180E	SCGC1	FTM2	FTM1	MTIM	RTC	PDB	CMPDAC3	CMPDAC2	CMPDAC1
0x180F	SCGC2	DBG	FLS	KBI	ADC	SCI	SPI	IIC	PGA
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGCCH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGCCL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1817	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1818	DBGCAX	RWAEN	RWA	0	0	0	0	0	0
0x1819	DBGCBX	RWBEN	RWB	0	0	0	0	0	0
0x181A	DBGCCX	RWCEN	RWC	0	0	0	0	0	0
0x181B	Reserved	—	—	—	—	—	—	—	—
0x181C	DBGC	DBGEN	ARM	TAG	BRKEN	0	0	0	LOOP1
0x181D	DBGT	TRGSEL	BEGIN	0	0	TRG			
0x181E	DBGS	AF	BF	CF	0	0	0	0	ARMF
0x181F	DBGCNT	0	0	0	CNT				
0x1820	FCDIV	DIVLD	PRDIV8	DIV					
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC	
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS						FPDIS	
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1826	FCMD	FCMD							
0x1827– 0x183F	Reserved	—	—	—	—	—	—	—	—
0x1840	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1845	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1846	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1849	PTCSE	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x184A	PTCDS	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	PTDPE	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x184D	PTDSE	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x184E	PTDDS	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x184F	Reserved	—	—	—	—	—	—	—	—
0x1850	PTEPE	0	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x1851	PTESE	0	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x1852	PTEDS	0	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x1853	Reserved	—	—	—	—	—	—	—	—
0x1854	PTFPE	0	0	0	0	0	PTFPE2	PTFPE1	PTFPE0
0x1855	PTFSE	0	0	0	0	0	PTFSE2	PTFSE1	PTFSE0
0x1856	PTFDS	0	0	0	0	0	PTFDS2	PTFDS1	PTFDS0
0x1857– 0x185F	Reserved	—	—	—	—	—	—	—	—
0x1860	HSCMP1CR0	0	FILTER_CNT			PMC		MMC	
0x1861	HSCMP1CR1	SE	WE	0	PMODE	INV	CO	OPE	EN
0x1862	HSCMP1FPR	FILT_PER							
0x1863	HSCMP1SCR	0	0	SMLEB	IER	IEF	CPR	CPF	COUT
0x1864	HSCMP1PCR	INPPE4	INPPE3	INPPE2	INPPE1	INMPE4	INMPE3	INMPE2	INMPE1
0x1865– 0x1867	Reserved	—	—	—	—	—	—	—	—
0x1868	HSCMP2CR0	0	FILTER_CNT			PMC		MMC	
0x1869	HSCMP2CR1	SE	WE	0	PMODE	INV	CO	OPE	EN
0x186A	HSCMP2FPR	FILT_PER							
0x186B	HSCMP2SCR	0	0	SMLEB	IER	IEF	CPR	CPF	COUT
0x186C	HSCMP2PCR	INPPE4	INPPE3	INPPE2	INPPE1	INMPE4	INMPE3	INMPE2	INMPE1

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x186D–0x186F	Reserved	—	—	—	—	—	—	—	—
0x1870	HSCMP3CR0	0	FILTER_CNT			PMC		MMC	
0x1871	HSCMP3CR1	SE	WE	0	PMODE	INV	CO	OPE	EN
0x1872	HSCMP3FPR	FILT_PER							
0x1873	HSCMP3SCR	0	0	SMLEB	IER	IEF	CPR	CPF	COUT
0x1874	HSCMP3PCR	INPPE4	INPPE3	INPPE2	INPPE1	INMPE4	INMPE3	INMPE2	INMPE1
0x1875–0x1877	Reserved	—	—	—	—	—	—	—	—
0x1878	PGACNTL0	TM	GAINSEL				LP	EN	
0x1879	PGACNTL1	0	0	0	CALMODE		CPD		
0x187A	PGACNTL2	0	0	SWTRIG	NUM_CLK_GS			ADIV	
0x187B	PGASTS	0	0	0	0	0	RUNNING	STCOMP	
0x187C–0x187F	Reserved	—	—	—	—	—	—	—	—
0x1880	FTM2SC	TOF	TOIE	CPWMS	CLKS		PS		
0x1881	FTM2CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x1882	FTM2CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x1883	FTM2MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x1884	FTM2MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x1885	FTM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x1886	FTM2C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x1887	FTM2C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x1888	FTM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x1889	FTM2C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x188A	FTM2C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x188B	FTM2C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x188C	FTM2C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x188D	FTM2C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x188E	FTM2C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x188F	FTM2C3VH	Bit 15	14	13	12	11	10	9	Bit 8
0x1890	FTM2C3VL	Bit 7	6	5	4	3	2	1	Bit 0
0x1891	FTM2C4SC	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	0	0
0x1892	FTM2C4VH	Bit 15	14	13	12	11	10	9	Bit 8
0x1893	FTM2C4VL	Bit 7	6	5	4	3	2	1	Bit 0
0x1894	FTM2C5SC	CH5F	CH5IE	MS5B	MS5A	ELS5B	ELS5A	0	0
0x1895	FTM2C5VH	Bit 15	14	13	12	11	10	9	Bit 8
0x1896	FTM2C5VL	Bit 7	6	5	4	3	2	1	Bit 0
0x1897–0x189F	Reserved	—	—	—	—	—	—	—	—

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x18A0	FTM2CNTINH	Bit 15	14	13	12	11	10	9	Bit 8
0x18A1	FTM2CNTINL	Bit 7	6	5	4	3	2	1	Bit 0
0x18A2	FTM2STATUS	0	0	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
0x18A3	FTM2MODE	FAULTIE	FAULTM		CAPTTES T	PWMSYN C	WPDIS	INIT	FTMEN
0x18A4	FTM2SYNC	SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
0x18A5	FTM2OUTINIT	0	0	CH5OI	CH4OI	CH3OI	CH2OI	CH1OI	CH0OI
0x18A6	FTM2OUTMASK	0	0	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
0x18A7	FTM2COMBINE0	0	FAULTEN	SYNCEN	DTEN	0	0	COMP	COMBINE
0x18A8	FTM2COMBINE1	0	FAULTEN	SYNCEN	DTEN	0	0	COMP	COMBINE
0x18A9	FTM2COMBINE2	0	FAULTEN	SYNCEN	DTEN	0	0	COMP	COMBINE
0x18AA	FTM Reserved	0	0	0	0	0	0	0	0
0x18AB	FTM2DEADTIME	DTPS		DTVAL					
0x18AC	FTM2EXTTRIG	—	INITTRIG EN	—	—	—	—	—	—
0x18AD	FTM2POL	—	—	POL5	POL4	POL3	POL2	POL1	POL0
0x18AE	FTM2FMS	FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0
0x18AF	FTM2FILTER0	CH1FVAL				CH0FVAL			
0x18B0	FTM2FILTER1	CH3FVAL				CH2FVAL			
0x18B1	FTM2FLTFILTER	0	0	0	0	FFVAL			
0x18B2	FTM2FLTCtrl	0	0	0	FFLTR0E N	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
0x18B3– 0x18BF	Reserved	—	—	—	—	—	—	—	—

Reserved flash memory locations, shown in [Table 4-4](#), are used for storing values used by several registers. These registers include an 8-byte backdoor key, NVBACKKEY, which can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the reserved flash memory are transferred into corresponding FPROT and FOPT registers in the high-page registers area to control security and block protection options.

The factory ICS trim value is stored in the flash information row (IFR<sup>1</sup>) and will be loaded into the ICSTRM and ICSSC registers after any reset. The internal reference trim values stored in flash, TRIM and FTRIM, can be programmed by third party programmers and must be copied into the corresponding ICS registers by user code to override the factory trim.

#### NOTE

When the MCU is reset into active BDM, the trim value in the IFR will not be loaded. Instead, the ICSTRM register will reset to 0x80 and the FTRIM bit in the ICSSC register will be reset to 0.

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAE	Reserved for storage of FTRIM	0	0	0	0	0	0	0	FTRIM
TRIM									
8-Byte Comparison Key									
—									
0xFFB8 – 0xFFBC	Reserved	—	—	—	—	—	—	—	—
0xFFBD	NVPROT	FPS							FPDIS
0xFFBE	Reserved	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC	

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

1. IFR — Nonvolatile information memory that can be only accessed during production test. During production test, system initialization, configuration and test information is stored in the IFR. This information cannot be read or modified in normal user or background debug modes.

## 4.4 RAM

The MC9S08MP16 Series includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

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

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08MP16 Series, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor-provided equate file).

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

When security is enabled, the RAM is considered a secure memory resource and is not accessible through background debug mode (BDM) or through code executing from non-secure memory. See [Section 4.6, “Security”](#), for a detailed description of the security feature.

## 4.5 FLASH

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

## 4.5.1 Features

Features of the FLASH memory include:

- FLASH size
  - MC9S08MP16: 16,384 bytes (32 pages of 512 bytes each)
  - MC9S08MP12: 12,288 bytes (24 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

## 4.5.2 Program and Erase Times

Before any program or erase command can be accepted, the FLASH clock divider register (FCDIV) must be written to set the internal clock for the FLASH module to a frequency ( $f_{FCLK}$ ) between 150 kHz and 200 kHz (see [Section 4.7.2, “FLASH Clock Divider Register \(FCDIV\)”](#)). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ( $1/f_{FCLK}$ ) is used by the command processor to time program and erase pulses. An integer number of these timing pulses is used by the command processor to complete a program or erase command.

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

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

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 $\mu s$
Byte program (burst)	4	20 $\mu s$ <sup>1</sup>
Page erase	4000	20 ms
Mass erase	20,000	100 ms

<sup>1</sup> Excluding start/end overhead

### 4.5.3 Program and Erase Command Execution

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

1. Write a data value to an address in the FLASH array. The address and data information from this write is latched into the FLASH interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For page erase commands, the address may be any address in the 512-byte page of FLASH to be erased. For mass erase and blank check commands, the address can be any address in the FLASH memory. Whole pages of 512 bytes are the smallest block of FLASH that may be erased.

#### NOTE

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

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

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

A strictly monitored procedure must be obeyed or the command will not be accepted. This minimizes the possibility of any unintended changes to the FLASH memory contents. The command complete flag (FCCF) indicates when a command is complete. The command sequence must be completed by clearing FCBEF to launch the command. [Figure 4-2](#) is a flowchart for executing all of the commands except for burst programming. The FCDIV register must be initialized before using any FLASH commands. This must be done only once following a reset.

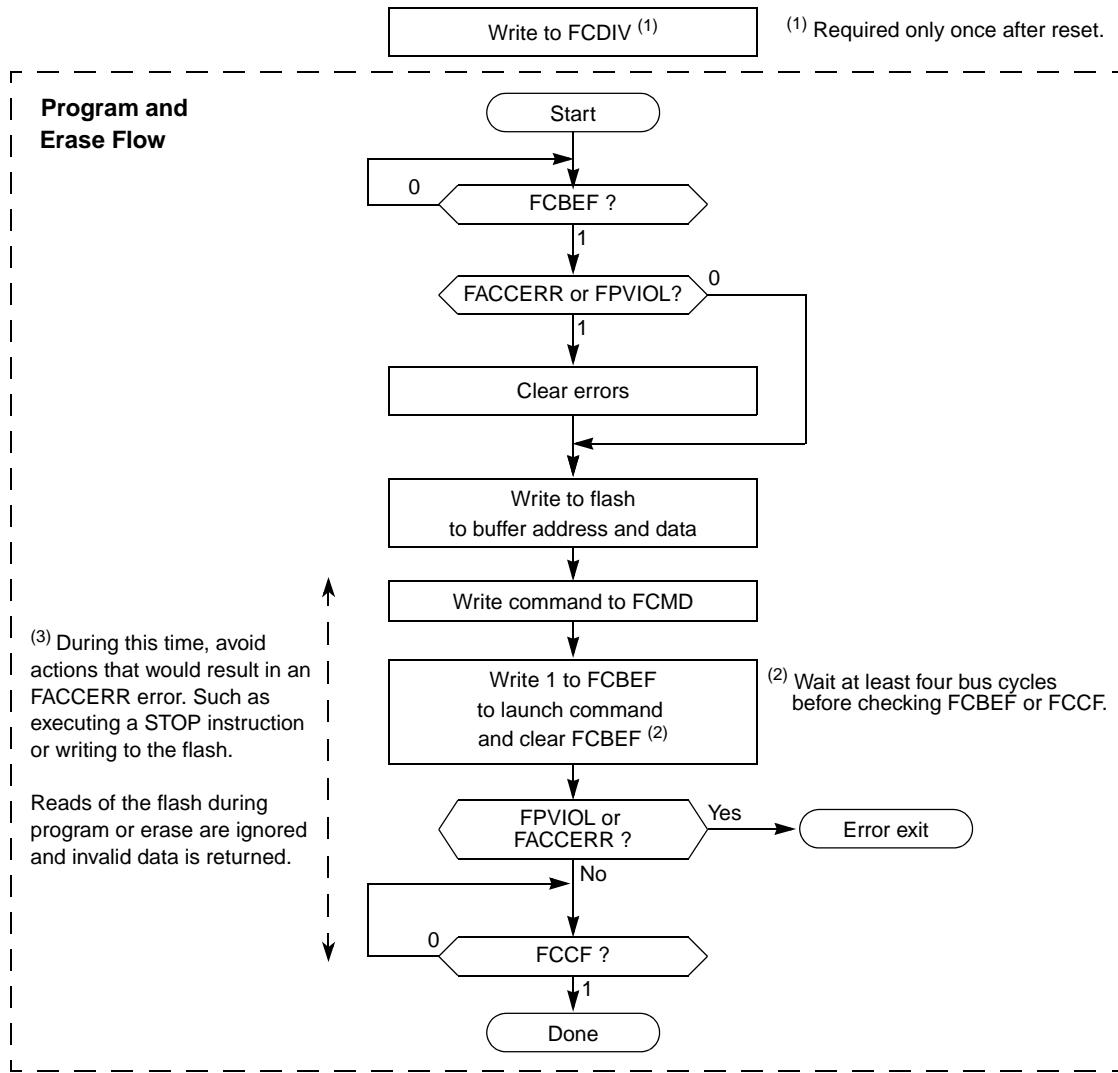


Figure 4-2. FLASH Program and Erase Flowchart

#### 4.5.4 Burst Program Execution

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

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

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

The first byte of a series of sequential bytes being programmed in burst mode will take the same amount of time to program as a byte programmed in standard mode. Subsequent bytes will program in the burst program time provided that the conditions above are met. In the case the next sequential address is the beginning of a new row, the program time for that byte will be the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump will be disabled and high voltage removed from the array.

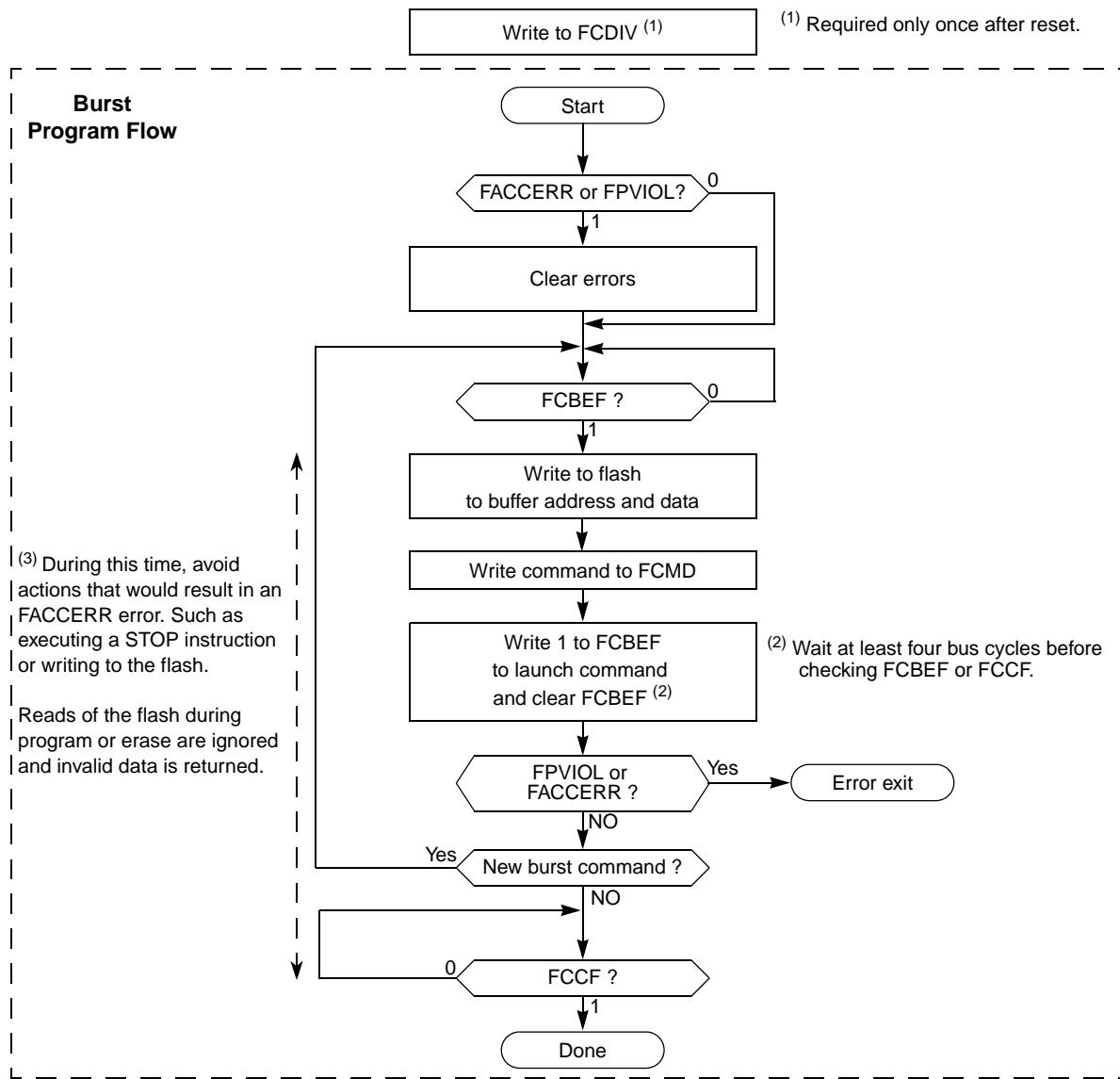


Figure 4-3. FLASH Burst Program Flowchart

### 4.5.5 Access Errors

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

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

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

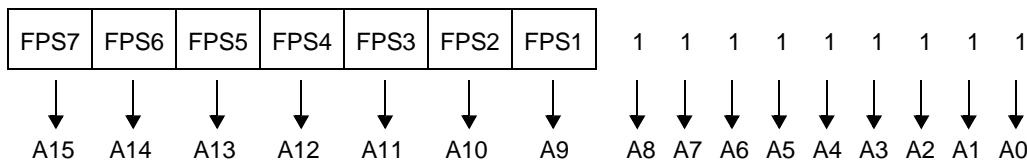
### 4.5.6 FLASH Block Protection

The block protection feature prevents the protected region of FLASH from program or erase changes. Block protection is controlled through the FLASH protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of FLASH, 0xFFFF. (See [Section 4.7.5, “FLASH Protection Register \(FPROT and NVPROT\)”](#)).

After exit from reset, FPROT is loaded with the contents of the NVPROT location, which is in the nonvolatile register block of the FLASH memory. FPROT cannot be changed directly from application software to prevent runaway programs from decreasing the block protection settings. Because NVPROT is within the last 512 bytes of FLASH, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written to any value through background debug commands, which allows a protected FLASH memory to be erased and reprogrammed.

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

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



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

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

#### 4.5.7 Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the FLASH memory must be block protected by programming the NVPROT register located at address 0xFFBD.

All of the interrupt vectors (memory locations 0xFFC0–0xFFFFD) are redirected, though the reset vector (0xFFFFE:FFFF) is not. The redirected vector read accesses a flash memory location offset from the original vector location by a negative value equal to the size of the protected region as defined by NVPROT. Modifications to the FPS bits in FPROT register to override the value loaded from NVPROT after reset do not affect the vector redirection logic.

For example, if 512 bytes of FLASH are protected, 0xFC is programmed in NVPROT and the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFFD) are redirected to the locations 0xFDC0–0xFDFD. With this configuration and redirection enabled, if interrupt vector 15 is serviced, the values in the locations 0xFDE0:FDE1 are used for the vector instead of the values in the locations 0FFE0:FFE1. This allows the user to reprogram the unprotected portion of the FLASH with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

### 4.6 Security

The MC9S08MP16 Series includes circuitry to prevent unauthorized access to the contents of FLASH and RAM memory. When security is engaged, FLASH and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location which can be done at the same time the FLASH memory is programmed. The 1:0 state disengages security and the other three combinations engage security. Notice the erased state (1:1) makes the MCU secure. During development, whenever the FLASH is erased, user code should immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

The on-chip debug module cannot be enabled while the MCU is secure. The separate background debug controller can still be used for background memory access commands of unsecured resources.

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

1. Writing 1 to KEYACC in the FCNFG register. This makes the FLASH module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a FLASH program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX should not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was just written matches the key stored in the FLASH locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

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

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in FLASH memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other FLASH memory location. The nonvolatile registers are in the same 512-byte block of FLASH as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

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

1. Disable any block protections by writing FPROT.
2. Mass erase FLASH if necessary.
3. Blank check FLASH. Provided FLASH is completely erased, security is disengaged until the next reset.

To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

## 4.7 FLASH Registers and Control Bits

The FLASH module has nine 8-bit registers in the high-page register space. Two locations (NVOPT, NVPROT) in the nonvolatile register space in FLASH memory are copied into corresponding high-page control registers (FOPT, FPROT) at reset. There is also an 8-byte comparison key in FLASH memory. Refer to [Table 4-3](#) and [Table 4-4](#) for the absolute address assignments for all FLASH registers. This section refers to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file is normally used to translate these names into the appropriate absolute addresses.

### 4.7.1 FLASH Clock Gating

The bus clock to the FLASH module can be gated on and off using the FLS bit in SCGC2. This bit is set after any reset, which enables continuous clocking to the FLASH module. To conserve power, the FLS bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

Clearing of the FLS bit does not affect normal code execution from the flash array. The FLS bit must be set when accessing the FLASH registers and during operations such as programming and erasing which are initiated via FLASH register accesses.

### 4.7.2 FLASH Clock Divider Register (FCDIV)

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



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

Table 4-6. FCDIV Register Field Descriptions

Field	Description
7 DIVLD	<b>Divisor Loaded Status Flag</b> — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit and the first write to this register causes this bit to become set regardless of the data written. 0 FCDIV has not been written since reset; erase and program operations disabled for FLASH. 1 FCDIV has been written since reset; erase and program operations enabled for FLASH.

**Table 4-6. FCDIV Register Field Descriptions (continued)**

Field	Description
6 PRDIV8	<b>Prescale (Divide) FLASH Clock by 8</b> 0 Clock input to the FLASH clock divider is the bus rate clock. 1 Clock input to the FLASH clock divider is the bus rate clock divided by 8.
5:0 DIV[5:0]	<b>Divisor for FLASH Clock Divider</b> — The FLASH clock divider divides the bus rate clock (or the bus rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV field plus one. The resulting frequency of the internal FLASH clock must fall within the range of 200 kHz to 150 kHz for proper FLASH operations. Program/Erase timing pulses are one cycle of this internal FLASH clock which corresponds to a range of 5 $\mu$ s to 6.7 $\mu$ s. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See <a href="#">Equation 4-1</a> and <a href="#">Equation 4-2</a> .

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

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

Table 4-7 shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

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

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

### 4.7.3 FLASH Options Register (FOPT and NVOPT)

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

7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	SEC01	SEC00
W							
Reset	This register is loaded from nonvolatile location NVOPT during reset.						
	= Unimplemented or Reserved						

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

**Table 4-8. FOPT Register Field Descriptions**

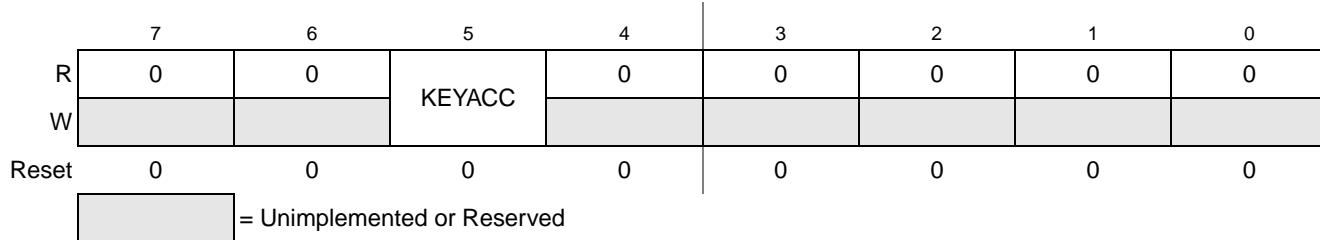
Field	Description
7 KEYEN	<b>Backdoor Key Mechanism Enable</b> — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to <a href="#">Section 4.6, "Security."</a> 0 No backdoor key access allowed. 1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset.
6 FNORED	<b>Vector Redirection Disable</b> — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	<b>Security State Code</b> — This 2-bit field determines the security state of the MCU as shown in <a href="#">Table 4-9</a> . When the MCU is secure, the contents of RAM and FLASH memory cannot be accessed by instructions from any unsecured source including the background debug interface. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of FLASH. For more detailed information about security, refer to <a href="#">Section 4.6, "Security."</a>

**Table 4-9. Security States<sup>1</sup>**

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

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

#### 4.7.4 FLASH Configuration Register (FCNFG)

**Figure 4-7. FLASH Configuration Register (FCNFG)****Table 4-10. FCNFG Register Field Descriptions**

Field	Description
5 KEYACC	<b>Enable Writing of Access Key</b> — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to <a href="#">Section 4.6, "Security."</a> 0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a FLASH programming or erase command. 1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.

## 4.7.5 FLASH Protection Register (FPROT and NVPROT)

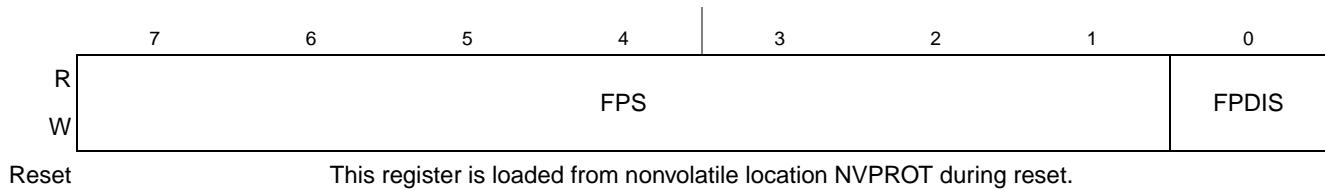
The FPROT register defines which FLASH sectors are protected against program and erase operations. FPROT also determines if FLASH protection is disabled. Mass erase is not possible if any of the sectors are protected.

During the reset sequence, the FPROT register is loaded from the nonvolatile location NVPROT. To change the data flash block protection on a temporary basis, the FPROT[FPS] bits can be written.

- When FPDIS is set, all FPROT bits are writeable.
- When FPDIS is cleared:
  - FPDIS cannot be set
  - While increasing the size of the protected region, the FPS bits are writeable.
  - Any attempted FPS write to decrease the size of the protected memory is ignored. This ignored write is not considered a protection violation and FPVIOL is not affected.

While in active background mode, background commands can be used to change the contents of FPROT to any value.

To change the protection that is loaded during the reset sequence, the sector containing NVPROT must be unprotected and erased. Then, reprogram NVPROT.



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

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

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

## 4.7.6 FLASH Status Register (FSTAT)

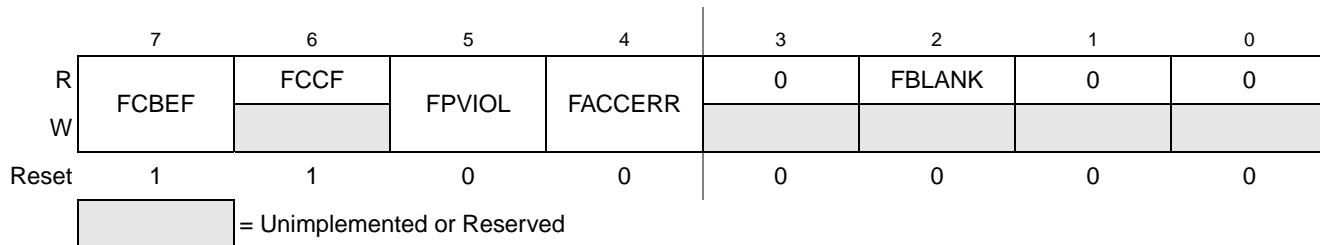


Figure 4-9. FLASH Status Register (FSTAT)

Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	<b>FLASH Command Buffer Empty Flag</b> — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a 1 to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command can be written to the command buffer.
6 FCCF	<b>FLASH Command Complete Flag</b> — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect. 0 Command in progress 1 All commands complete
5 FPVIOL	<b>Protection Violation Flag</b> — FPVIOL is set automatically when a command is written that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.
4 FACCERR	<b>Access Error Flag</b> — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see <a href="#">Section 4.5.5, “Access Errors.”</a> FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect. 0 No access error. 1 An access error has occurred.
2 FBLANK	<b>FLASH Verified as All Blank (erased) Flag</b> — FBLANK is set automatically at the conclusion of a blank check command if the entire FLASH array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the FLASH array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the FLASH array is completely erased (all 0xFF).

## 4.7.7 FLASH Command Register (FCMD)

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

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

**Figure 4-10. FLASH Command Register (FCMD)****Table 4-13. FLASH Commands**

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

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

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

# Chapter 5

## Resets, Interrupts, and General System Control

### 5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9S08MP16 Series. Some interrupt sources from peripheral modules are discussed in greater detail in other sections of this document. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog are not part of on-chip peripheral systems and have their own chapters.

### 5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- System reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vector for each module (reduces polling overhead) (see [Table 5-2](#))

### 5.3 MCU Reset

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

The MC9S08MP16 Series has the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD) - any address in memory map that is listed as unimplemented will produce an illegal address reset
- Background debug forced reset

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

## 5.4 Computer Operating Properly (COP) Watchdog

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

After any reset, the COPT bits becomes set in SOPT1 enabling the COP watchdog (see [Section 5.8.3, “System Options Register 1 \(SOPT1\)](#),” for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing both COPT bits.

The COP counter is reset by writing 0x0055 and 0x00AA (in this order) to the address of SRS during the selected timeout period. 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. As soon as the write sequence is done, the COP timeout period is restarted. If the program fails to do this during the time-out period, the MCU will reset. Also, if any value other than 0x0055 or 0x00AA is written to SRS, the MCU is immediately reset.

The COPCLKS bit in SOPT2 (see [Section 5.8.4, “System Options Register 2 \(SOPT2\)](#),” for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1-kHz clock source. With each clock source, there are three associated time-outs controlled by the COPT bits in SOPT1. [Table 5-1](#) summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1-kHz clock source and the longest time-out ( $2^{10}$  cycles).

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

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

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

<sup>2</sup> Values shown in milliseconds based on  $t_{LPO} = 1$  ms. See  $t_{LPO}$  in the data sheet for the tolerance of this value.

When the bus clock source is selected, windowed COP operation is available by setting COPW in the SOPT2 register. In this mode, writes to the SRS register to clear the COP timer must occur in the last 25%

of the selected timeout period. A premature write immediately resets the MCU. When the 1-kHz clock source is selected, windowed COP operation is not available.

The COP counter is initialized by the first writes to the SOPT1 and SOPT2 registers after any system reset. Subsequent writes to SOPT1 and SOPT2 have no effect on COP operation. Even if the application will use the reset default settings of COPT, COPCLKS, and COPW bits, the user should write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. This will prevent accidental changes if the application program gets lost. The write to SRS that services (clears) the COP counter should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

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

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

## 5.5 Interrupts

### NOTE

There is no IRQ module integrated on MC9S08MP16 Series devices.

Therefore the BIH and BIL instructions should not be used.

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

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

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

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

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is

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

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

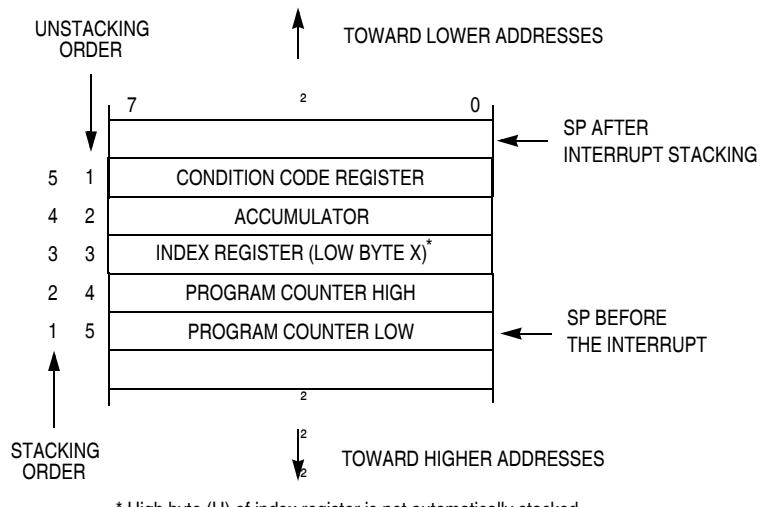
### NOTE

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

If more than one interrupt is pending when the I bit is cleared, the highest priority source is serviced first (see [Table 5-2](#)).

## 5.5.1 Interrupt Stack Frame

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



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

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

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

## 5.5.2 Interrupt Vectors, Sources, Priority Masks and Local Masks

Table 5-2 provides a summary of all interrupt sources. Higher-priority sources are located toward the bottom of the table. The high-order byte of the address for the interrupt service routine is located at the first address in the vector address column, and the low-order byte of the address for the interrupt service routine is located at the next highest address.

When an interrupt condition occurs, an associated flag bit becomes set.. If the associated local interrupt enable is 1, an interrupt request is sent to the IPC.. The IPC will compare the interrupt priority level of the pending interrupt versus the interrupt priority mask level. If the pending interrupt is equal to or greater than the mask level, it is sent to the CPU. For more information see [Chapter 16, “Interrupt Priority Controller \(S08IPCV1\)”](#).

Within the CPU, if the global interrupt mask (I bit in the CCR) is 0, the CPU will finish the current instruction; stack the PCL, PCH, X, A, and CCR CPU registers; set the I bit; and then fetch the interrupt vector for the highest priority pending interrupt. Processing then continues in the interrupt service routine.

**Table 5-2. Reset and Interrupt Vector Summary**

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lowest	31	0xFFC0/0xFFC1			Reserved		
	30	0xFFC2/0xFFC3	Vkeyboard3	KBI3	KBF	KBIE	Keyboard 3 pins
	29	0xFFC4/0xFFC5	Vkeyboard2	KBI2	KBF	KBIE	Keyboard 2 pins
	28	0xFFC6/0xFFC7	Vkeyboard1	KBI1	KBF	KBIE	Keyboard 1 pins
	27	0xFFC8/0xFFC9	Vhscmp3	HSCMP3	CFR, CFF	IER, IEF	Analog comparator 3
	26	0xFFCA/0xFFCB	Vhscmp2	HSCMP2	CFR, CFF	IER, IEF	Analog comparator 2
	25	0xFFCC/0xFFCD	Vhscmp1	HSCMP1	CFR, CFF	IER, IEF	Analog comparator 1
	24	0xFFCE/0xFFCF	Vrtc	RTC	RTIF	RTIE	Real-time counter
	23	0xFFD0/0xFFD1	Viic	IIC	IICIF	IICIE	IIC
	22	0xFFD2/0xFFD3	Vscitx	SCI	TDRE, TC	TIE, TCIE	SCI transmit
	21	0xFFD4/0xFFD5	Vscirx	SCI	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIIE, RXEDGIE	SCI receive
	20	0xFFD6/0xFFD7	Vscierr	SCI	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI error
	19	0xFFD8/0xFFD9	Vspi	SPI	SPRF, MODF, SPTEF	SPIE, SPIE, SPTIE	SPI
	18	0xFFDA/0xFFDB	Vadc	ADC	COCO	AIEN	ADC Conversion
	17	0xFFDC/0xFFDD	Vpdb2	PDB2	COF, DAF, DBF	COIE, DAIE, DBIE	PDB2
	16	0xFFDE/0xFFDF	Vpdb1	PDB1	COF, DAF, DBF	COIE, DAIE, DBIE	PDB1
	15	0xFFE0/0xFFE1	Vmtim	MTIM	TOF	TOIE	MTIM overflow
	14	0xFFE2/0xFFE3	Vftm2ovf	FTM2	TOF	TOIE	FTM2 overflow
	13	0xFFE4/0xFFE5	Vftm2ch5	FTM2	CH5F	CH5IE	FTM2 channel 5
	12	0xFFE6/0xFFE7	Vftm2ch4	FTM2	CH4F	CH4IE	FTM2 channel 4
	11	0xFFE8/0xFFE9	Vftm2ch3	FTM2	CH3F	CH3IE	FTM2 channel 3
	10	0xFFEA/0xFFEB	Vftm2ch2	FTM2	CH2F	CH2IE	FTM2 channel 2
	9	0xFFEC/0xFFED	Vftm2ch1	FTM2	CH1F	CH1IE	FTM2 channel 1
	8	0xFFEE/0xFFEF	Vftm2ch0	FTM2	CH0F	CH0IE	FTM2 channel 0
	7	0xFFFF/0xFFFF1	Vftm1ovf	FTM1	TOF	TOIE	FTM1 overflow
	6	0xFFF2/0xFFFF3	Vftm1ch1	FTM1	CH1F	CH1IE	FTM1 channel 1
	5	0xFFF4/0xFFFF5	Vftm1ch0	FTM1	CH0F	CH0IE	FTM1 channel 0
	4	0xFFF6/0xFFFF7	Vftm2fault	FTM2	FAULTF	FAULTIE	FTM2 fault
	3	0xFFF8/0xFFFF9	Vftm1fault	FTM1	FAULTF	FAULTIE	FTM1 fault
	2	0xFFFFA/0xFFFFB	Vlww	System control	LVWF	LVWIE	Low-voltage warning
	1	0xFFFFC/0xFFFFD	Vswi	Core	SWI Instruction	—	Software interrupt
	0	0xFFFFE/0xFFFFF	Vreset	System control	COP LVD RESET pin Illegal opcode Illegal address POR	COPE LVDRE RSTPE — — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address Power-on-reset

## 5.6 Low-Voltage Detect (LVD) System

The MC9S08MP16 Series includes a system to protect against low voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with trip voltages for warning and detection. The LVD circuit is enabled when LVDE in SPMSC1 is set to 1. The LVD is disabled upon entering any of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU cannot enter stop2, and the current consumption in stop3 with the LVD enabled will be higher.

### 5.6.1 Power-On Reset Operation

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

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

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

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

The LVD system has a low voltage warning flag (LVWF) to indicate to the user that the supply voltage is approaching, but remains above, the LVD voltage. The LVW also has an interrupt associated with it, enabled by setting the LVWIE bit. If enabled, an LVW interrupt request will occur when the LVWF bit becomes set.

## 5.7 Peripheral Clock Gating

The MC9S08MP16 Series includes a clock gating system to manage the bus clock sources to the individual peripherals. Using this system, the user can enable or disable the bus clock to each of the peripherals at the clock source, eliminating unnecessary clocks to peripherals which are not in use and thereby reducing the overall run and wait mode currents.

Out of reset, all peripheral clocks will be enabled. For lowest possible run or wait currents, user software should disable the clock source to any peripheral not in use. The actual clock will be enabled or disabled immediately following the write to the Clock Gating Control registers (SCGC1 and SCGC2). Any peripheral with a gated clock cannot be used unless its clock is enabled. Writing to the registers of a peripheral with a disabled clock has no effect.

#### NOTE

User software should disable the peripheral before disabling the clocks to the peripheral.

In stop modes, the bus clock is disabled for all gated peripherals, regardless of the settings in SCGC1 and SCGC2.

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

Several registers in the high-page register space are related to reset and interrupt systems.

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

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

### 5.8.1 System Reset Status Register (SRS)

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

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

<sup>1</sup> u = unaffected

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

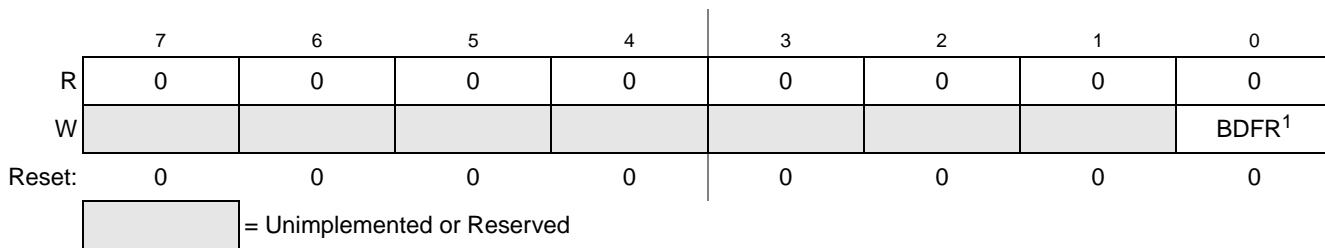
**Figure 5-2. System Reset Status (SRS)**

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

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

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

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



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

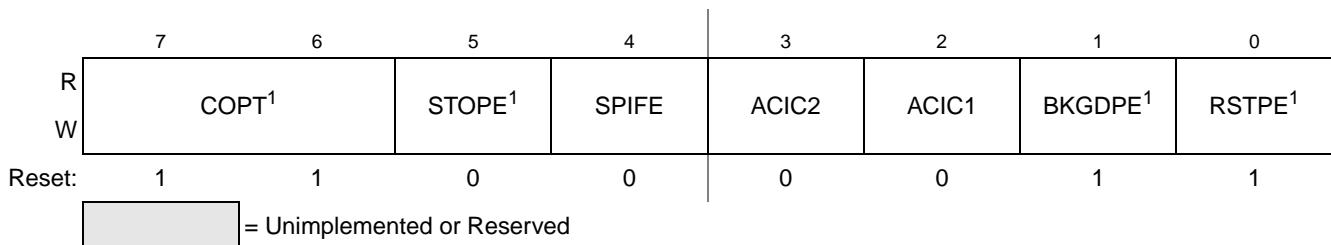
**Figure 5-3. System Background Debug Force Reset Register (SBDFR)**

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

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

### 5.8.3 System Options Register 1 (SOPT1)

This high page register contains bits to configure MCU specific features on the MC9S08MP16 Series devices. SOPT1 should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

**Figure 5-4. System Options Register 1 (SOPT1)**

<sup>1</sup> These bits can be written only one time after reset. Additional writes are ignored.

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

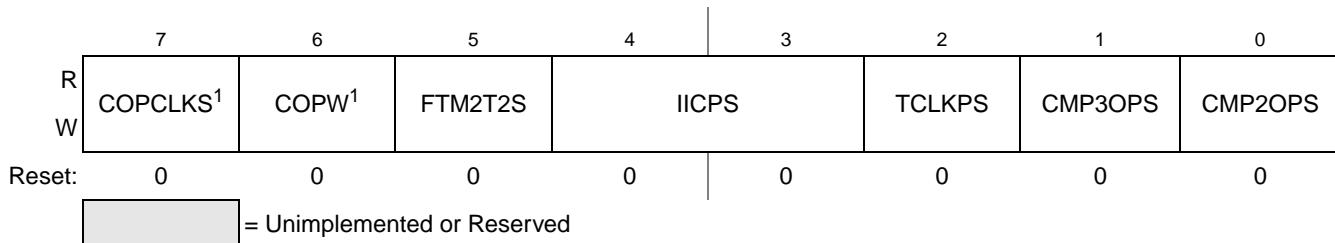
Field	Description
7:6 COPT[1:0]	<b>COP Watchdog Timeout</b> — These write-once bits select the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. See <a href="#">Table 5-1</a> .
5 STOPE	<b>Stop Mode Enable</b> — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
4 SPIFE	<b>SPI Ports Filter Enable</b> — This bit configures the input filters for SPI pins. 0 Disable input filter on SPI port pins to allow for higher SPI baud rate. 1 Enable input filter on SPI port pins to eliminate noise and restrict maximum SPI baud rate.
3 ACIC2	<b>Analog Comparator to Input Capture Enable</b> — This bit connects the output of HSCMP2 to FTM1 input channel 1. 0 HSCMP2 output not connected to FTM1 input channel 1. 1 HSCMP2 output connected to FTM1 input channel 1.
2 ACIC1	<b>Analog Comparator to Input Capture Enable</b> — This bit connects the output of HSCMP1 to FTM1 input channel 0. 0 HSCMP1 output not connected to FTM1 input channel 0. 1 HSCMP1 output connected to FTM1 input channel 0.

**Table 5-5. SOPT1 Register Field Descriptions (continued)**

Field	Description
1 BKGDP	<b>Background Debug Mode Pin Enable</b> — This write-once bit when set enables the PTF0/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 PTF0/BKGD/MS pin functions as PTF0 1 PTF0/BKGD/MS pin functions as BKGD/MS
0 RSTPE	<b>RESET Pin Enable</b> — This write-once bit when set enables the PTF1/RESET pin to function as RESET. When clear, the pin functions as an output only port function. This pin defaults to RESET function following any MCU reset. When RSTPE is set, an internal pullup device is enabled on RESET. 0 PTF1/RESET pin functions as PTF1. 1 PTF1/RESET pin functions as RESET.

## 5.8.4 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the MC9S08MP16 Series devices.

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

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

**Table 5-6. SOPT2 Register Field Descriptions**

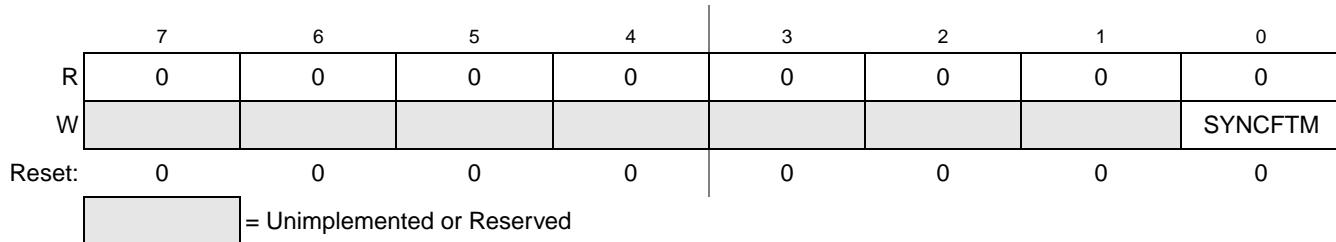
Field	Description
7 COPCLKS	<b>COP Watchdog Clock Select</b> — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1-kHz clock is source to COP. 1 Bus clock is source to COP.
6 COPW	<b>COP Window</b> — This write-once bit selects the COP operation mode. When set, the 0x55-0xAA write sequence to the SRS register must occur in the last 25% of the selected period. Any write to the SRS register during the first 75% of the selected period will reset the MCU. 0 Normal COP operation 1 Window COP operation (only if COPCLKS = 1)
5 FTM2T2S	<b>Flextimer 2 Trigger 2 Input Select</b> — This bit selects the trigger source for the Flextimer 2 Trigger 2 input to be either from HSCMP3 output or the Flextimer 1 output channel 0. 0 HSCMP3 output connected to FTM2 Trigger 2 input. 1 FTM1 channel 0 output connected to FTM2 Trigger 2 input.
4:3 IICPS[1:0]	<b>IIC Pin Select</b> — This bit selects the location of the SDA and SCL pins of the IIC module. 00SDA on PTD0, SCL on PTD1. 01SDA on PTA0, SCL on PTA1. 10SDA on PTA2, SCL on PTA3. 11SDA on PTA4, SCL on PTA5.

**Table 5-6. SOPT2 Register Field Descriptions (continued)**

Field	Description
2 TCLKPS	<b>TCLK Pin Select</b> — This bit selects the location of the TCLK pin used for the FTM1, FTM2, and MTIM modules. 0 TCLK on PTC7. 1 TCLK on PTA4.
1 CMP3OPS	<b>CMP3OUT Pin Select</b> — This bit selects the location of the CMP3OUT pin of the HSCMP3 module. 0 CMP3OUT on PTD7. 1 CMP3OUT on PTB6.
0 CMP2OPS	<b>CMP2OUT Pin Select</b> — This bit selects the location of the CMP2OUT pin of the HSCMP2 module. 0 CMP2OUT on PTD6. 1 CMP2OUT on PTB5.

## 5.8.5 System Options Register 3 (SOPT3)

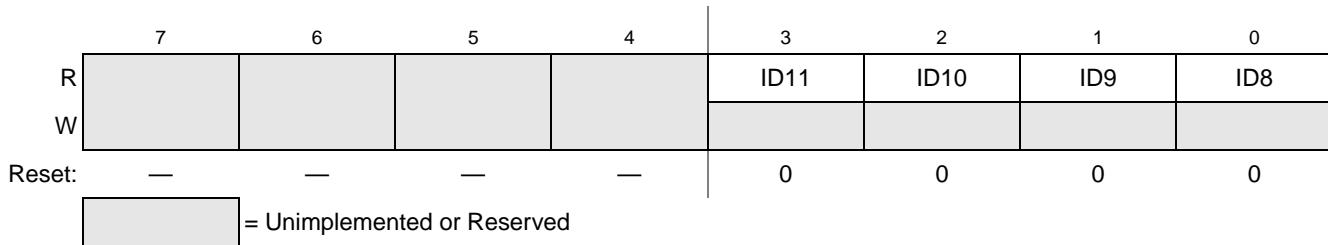
This high page register contains bits to configure MCU specific features on the MC9S08MP16 Series devices.

**Figure 5-6. System Options Register 3 (SOPT3)****Table 5-7. SOPT3 Register Field Descriptions**

Field	Description
0 SYNCFTM	<b>Synchronize FTM</b> — Writing a 1 to this bit generates a PWM synchronization trigger to the FTM modules. For more information, see the PWM Synchronization section in <a href="#">Chapter 12, “Flextimer Module (S08FTMV2)”</a> . 0 No trigger generated. 1 Generate a PWM synchronization trigger to the FTM modules.

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

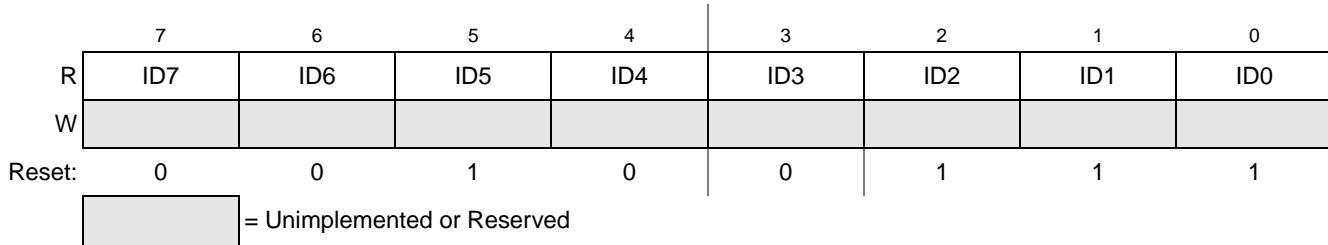
These high page read-only registers are included so host development systems can identify the HCS08 derivative. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.



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

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

Field	Description
7:4 Reserved	Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.
3:0 ID[11:8]	<b>Part Identification Number</b> — Each derivative in the HCS08 Family has a unique identification number. The MC9S08MP16 is hard coded to the value 0x027. See also ID bits in <a href="#">Table 5-9</a> .



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

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

Field	Description
7:0 ID[7:0]	<b>Part Identification Number</b> — Each derivative in the HCS08 Family has a unique identification number. The MC9S08MP16 is hard coded to the value 0x027. See also ID bits in <a href="#">Table 5-8</a> .

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

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

	7	6	5	4	3	2	1	0
R	LVWF <sup>1</sup>	0	LVWIE	LVDRE <sup>2</sup>	LVDSE	LVDE <sup>2</sup>	0	BGBE
W		LVWACK						
Reset:	0	0	0	1	1	1	0	0
	= Unimplemented or Reserved							

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

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

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

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

Field	Description
7 LVWF	<b>Low-Voltage Warning Flag</b> — The LVWF bit indicates the low voltage warning status. 0 Low voltage warning is not present. 1 Low voltage warning is present or was present.
6 LVWACK	<b>Low-Voltage Warning Acknowledge</b> — The LVWF bit indicates the low voltage warning status. Writing a 1 to LVWACK clears LVWF to a 0 if a low voltage warning is not present.
5 LVWIE	<b>Low-Voltage Warning Interrupt Enable</b> — This bit enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVWF = 1.
4 LVDRE	<b>Low-Voltage Detect Reset Enable</b> — This write-once bit enables LVD events to generate a hardware reset (provided LVDE = 1). 0 LVD events do not generate hardware resets. 1 Force an MCU reset when an enabled low-voltage detect event occurs.
3 LVDSE	<b>Low-Voltage Detect Stop Enable</b> — Provided LVDE = 1, this bit determines whether the low-voltage detect function operates when the MCU is in stop mode. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	<b>Low-Voltage Detect Enable</b> — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	<b>Bandgap Buffer Enable</b> — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

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

This register is used to report the status of the low voltage warning function, and to configure the stop mode behavior of the MCU.

	7	6	5	4	3	2	1	0
R	0	0	LVDV <sup>1</sup>	LVWV	PPDF	0	0	PPDC <sup>2</sup>
W						PPDACK		
Power-on Reset:	0	0	0	0	0	0	0	0
LVD Reset:	0	0	u	u	0	0	0	0
Any other Reset:	0	0	u	u	0	0	0	0
= Unimplemented or Reserved				u = Unaffected by reset				

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

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

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

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

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

**Table 5-12. LVD and LVW Trip Point Typical Values<sup>1</sup>**

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

<sup>1</sup> See Electrical Characteristics appendix for minimum and maximum values.

## 5.8.9 System Clock Gating Control 1 Register (SCGC1)

This high page register contains control bits to enable or disable the bus clock to the FTMx, MTIM, RTC, PDBx, HSCMPx, and DACx modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents. See [Section 5.7, “Peripheral Clock Gating,”](#) for more information.

### NOTE

User software should disable the peripheral before disabling the clocks to the peripheral.

	7	6	5	4	3	2	1	0
R W	FTM2	FTM1	MTIM	RTC	PDB	CMPDAC3	CMPDAC2	CMPDAC1
Reset:	1	1	1	1	1	1	1	1

Figure 5-11. System Clock Gating Control 1 Register (SCGC1)

Table 5-13. SCGC1 Register Field Descriptions

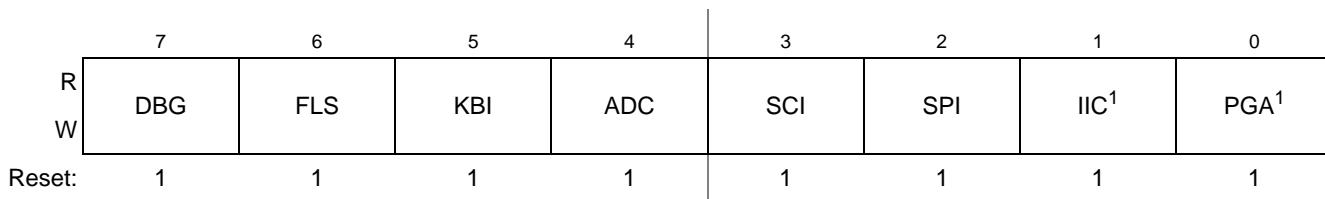
Field	Description
7 FTM2	<b>FTM2 Clock Gate Control</b> — This bit controls the clock gate to the FTM2 module. 0 Bus clock to the FTM2 module is disabled. 1 Bus clock to the FTM2 module is enabled.
6 FTM1	<b>FTM1 Clock Gate Control</b> — This bit controls the clock gate to the FTM1 module. 0 Bus clock to the FTM1 module is disabled. 1 Bus clock to the FTM1 module is enabled.
5 MTIM	<b>MTIM Clock Gate Control</b> — This bit controls the clock gate to the MTIM module. 0 Bus clock to the MTIM module is disabled. 1 Bus clock to the MTIM module is enabled.
4 RTC	<b>RTC Clock Gate Control</b> — This bit controls the bus clock gate to the RTC module. 0 Bus clock to the RTC module is disabled. 1 Bus clock to the RTC module is enabled.
3 PDB	<b>PDB1 and PDB2 Clock Gate Control</b> — This bit controls the clock gate to the PDB1 and PDB2 modules. 0 Bus clock to the PDB1 and PDB2 modules is disabled. 1 Bus clock to the PDB1 and PDB2 modules is enabled.
2 CMPDAC3	<b>HSCMP3 and DAC3 Clock Gate Control</b> — This bit controls the clock gate to the HSCMP3 and DAC3 modules. 0 Bus clock to the HSCMP3 and DAC3 modules is disabled. 1 Bus clock to the HSCMP3 and DAC3 modules is enabled.
1 CMPDAC2	<b>HSCMP2 and DAC2 Clock Gate Control</b> — This bit controls the clock gate to the HSCMP2 and DAC2 modules. 0 Bus clock to the HSCMP2 and DAC2 modules is disabled. 1 Bus clock to the HSCMP2 and DAC2 modules is enabled.
0 CMPDAC1	<b>HSCMP1 and DAC1 Clock Gate Control</b> — This bit controls the clock gate to the HSCMP1 and DAC1 modules. 0 Bus clock to the HSCMP1 and DAC1 modules is disabled. 1 Bus clock to the HSCMP1 and DAC1 modules is enabled.

## 5.8.10 System Clock Gating Control 2 Register (SCGC2)

This high page register contains control bits to enable or disable the bus clock to the DBG, FLS, KBI, ADC, SCI, SPI, IIC, and PGA modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents. See [Section 5.7, “Peripheral Clock Gating,”](#) for more information.

### NOTE

User software should disable the peripheral before disabling the clocks to the peripheral.



**Figure 5-12. System Clock Gating Control 2 Register (SCGC2)**

<sup>1</sup> These bits have no effect on MC9S08MP12 devices

**Table 5-14. SCGC2 Register Field Descriptions**

Field	Description
7 DBG	<b>DBG Clock Gate Control</b> — This bit controls the bus clock gate to the DBG module. 0 Bus clock to the DBG module is disabled. 1 Bus clock to the DBG module is enabled.
6 FLS	<b>Flash Clock Gate Control</b> — This bit controls the bus clock gate to the FLS module. This bit does not affect normal code execution from the flash array. 0 Bus clock to the flash module only active during array accesses. 1 Bus clock to the flash module is enabled.
5 KBI	<b>KBI1, KBI2, and KBI3 Clock Gate Control</b> — This bit controls the clock gate to all three KBI modules. 0 Bus clock to the KBI modules is disabled. 1 Bus clock to the KBI modules is enabled.
4 ADC	<b>ADC Clock Gate Control</b> — This bit controls the clock gate to the ADC module. 0 Bus clock to the ADC module is disabled. 1 Bus clock to the ADC module is enabled.
3 SCI	<b>SCI Clock Gate Control</b> — This bit controls the clock gate to the SCI module. 0 Bus clock to the SCI module is disabled. 1 Bus clock to the SCI module is enabled.
2 SPI	<b>SPI Clock Gate Control</b> — This bit controls the clock gate to the SPI module. 0 Bus clock to the SPI module is disabled. 1 Bus clock to the SPI module is enabled.
1 IIC	<b>IIC Clock Gate Control</b> — This bit controls the bus clock gate to the IIC module. 0 Bus clock to the IIC module is disabled. 1 Bus clock to the IIC module is enabled.
0 PGA	<b>PGA Clock Gate Control</b> — This bit controls the clock gate to the PGA module. 0 Bus clock to the PGA module is disabled. 1 Bus clock to the PGA module is enabled.



# Chapter 6

## Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9S08MP16 has six parallel I/O ports which include a total of 40 I/O pins and 2 output only pins. See [Chapter 2, “Pins and Connections,”](#) for more information about pin assignments and external hardware considerations of these pins.

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

After reset, the shared peripheral functions are disabled and the pins are configured as inputs ( $\text{PTxDDn} = 0$ ). The pin control functions for each pin are configured as follows: slew rate control disabled ( $\text{PTxSEN} = 0$ ), low drive strength selected ( $\text{PTxDsN} = 0$ ), and internal pull-ups disabled ( $\text{PTxPEn} = 0$ ).

### NOTE

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

### 6.1 Port Data and Data Direction

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

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

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

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

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

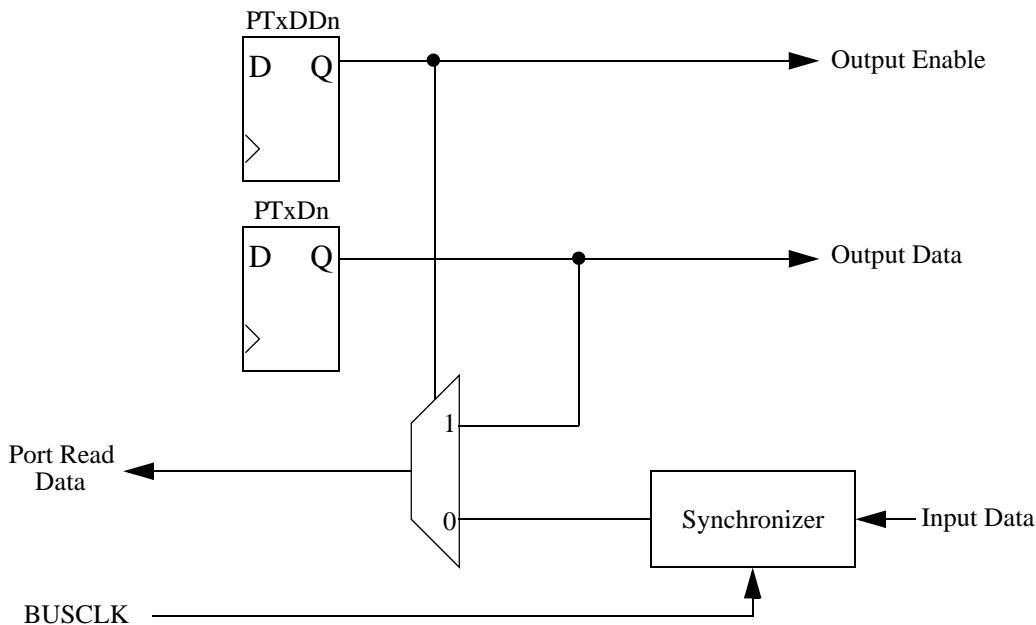


Figure 6-1. Parallel I/O Block Diagram

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

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

### 6.2.1 Port Internal Pull-up Enable

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

### 6.2.2 Port Slew Rate Enable

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

### 6.2.3 Port Drive Strength Select

An output pin can be configured for high output drive strength by setting the corresponding bit in the drive strength select register (PTxDSn). When high drive is selected, a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the MCU are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive

a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this, the EMC emissions may be affected by enabling pins as high drive.

## 6.3 Pin Behavior in Stop Modes

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

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

## 6.4 Parallel I/O and Pin Control Registers

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

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

### 6.4.1 Port A Registers

Port A is controlled by the registers listed below.

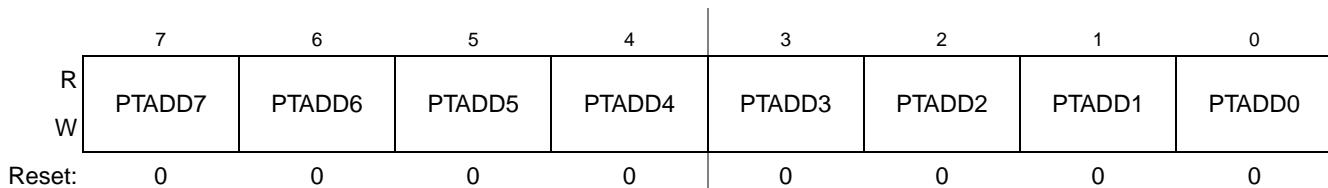
#### 6.4.1.1 Port A Data Register (PTAD)

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

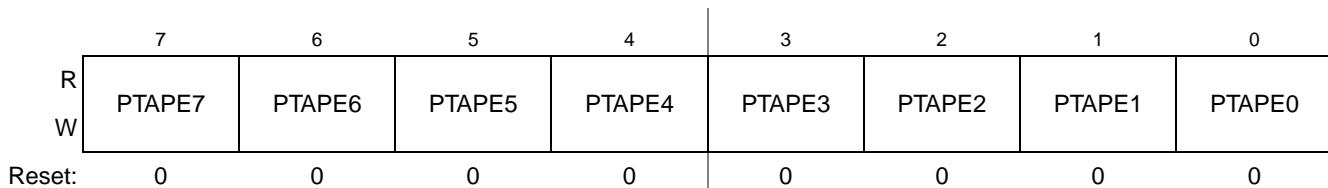
Figure 6-2. Port A Data Register (PTAD)

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

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

**6.4.1.2 Port A Data Direction Register (PTADD)****Figure 6-3. Port A Data Direction Register (PTADD)****Table 6-2. PTADD Register Field Descriptions**

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

**6.4.1.3 Port A Pull Enable Register (PTAPE)****Figure 6-4. Internal Pull Enable for Port A Register (PTAPE)****Table 6-3. PTAPE Register Field Descriptions**

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

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

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

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

Table 6-4. PTASE Register Field Descriptions

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

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

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

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

Table 6-5. PTADS Register Field Descriptions

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

### 6.4.2 Port B Registers

Port B is controlled by the registers listed below.

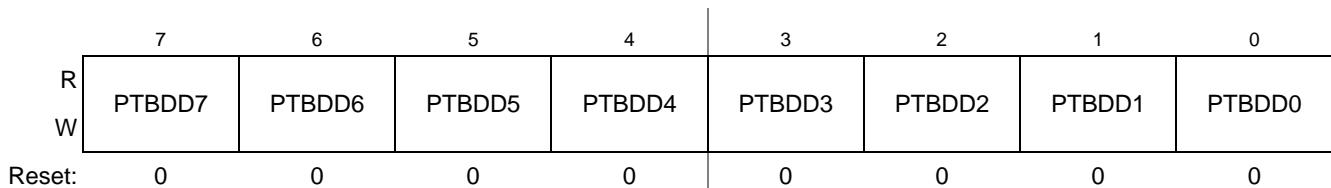
#### 6.4.2.1 Port B Data Register (PTBD)

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

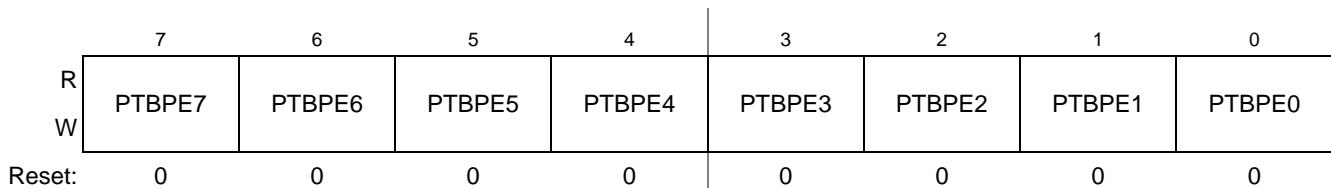
Figure 6-7. Port B Data Register (PTBD)

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

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

**6.4.2.2 Port B Data Direction Register (PTBDD)****Figure 6-8. Port B Data Direction Register (PTBDD)****Table 6-7. PTBDD Register Field Descriptions**

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

**6.4.2.3 Port B Pull Enable Register (PTBPE)****Figure 6-9. Internal Pull Enable for Port B Register (PTBPE)****Table 6-8. PTBPE Register Field Descriptions**

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

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

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

Figure 6-10. Slew Rate Enable for Port B Register (PTBSE)

Table 6-9. PTBSE Register Field Descriptions

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

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

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

Figure 6-11. Drive Strength Selection for Port B Register (PTBDS)

Table 6-10. PTBDS Register Field Descriptions

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

### 6.4.3 Port C Registers

Port C is controlled by the registers listed below.

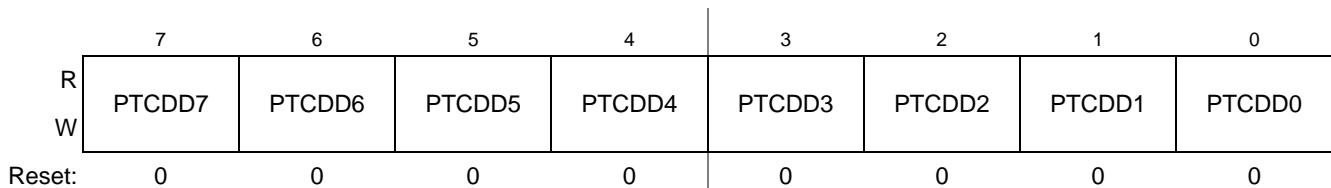
#### 6.4.3.1 Port C Data Register (PTCD)

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

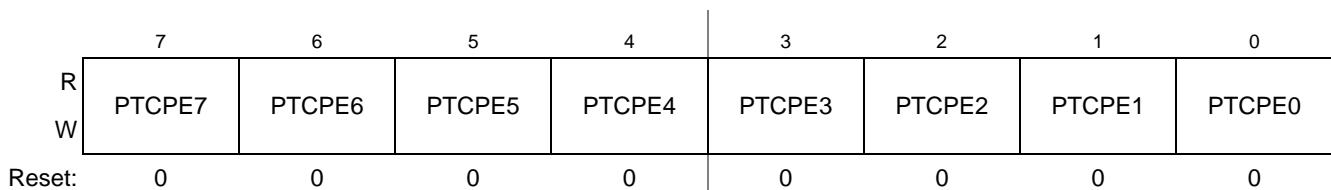
Figure 6-12. Port C Data Register (PTCD)

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

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

**6.4.3.2 Port C Data Direction Register (PTCDD)****Figure 6-13. Port C Data Direction Register (PTCDD)****Table 6-12. PTCDD Register Field Descriptions**

Field	Description
7:0 PTCDD[7:0]	<b>Data Direction for Port C Bits</b> — These read/write bits control the direction of port C pins and what is read for 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.

**6.4.3.3 Port C Pull Enable Register (PTCPE)****Figure 6-14. Internal Pull Enable for Port C Register (PTCPE)****Table 6-13. PTCPE Register Field Descriptions**

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

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

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

Figure 6-15. Slew Rate Enable for Port C Register (PTCSE)

Table 6-14. PTCSE Register Field Descriptions

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

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

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

Figure 6-16. Drive Strength Selection for Port C Register (PTCDS)

Table 6-15. PTCDS Register Field Descriptions

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

## 6.4.4 Port D Registers

Port D is controlled by the registers listed below.

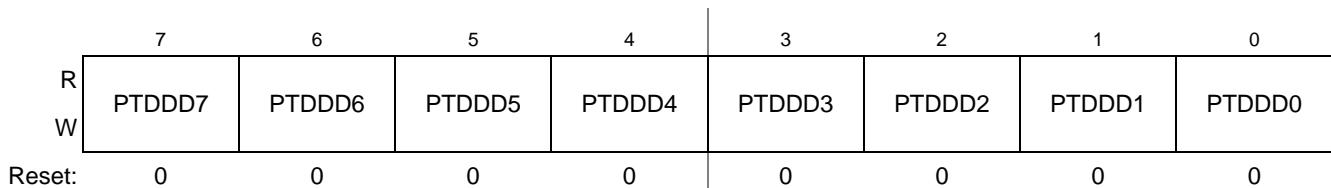
### 6.4.4.1 Port D Data Register (PTDD)

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

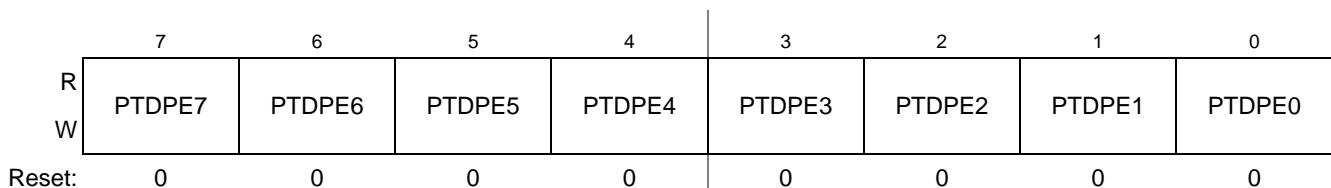
Figure 6-17. Port D Data Register (PTDD)

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

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

**6.4.4.2 Port D Data Direction Register (PTDDD)****Figure 6-18. Port D Data Direction Register (PTDDD)****Table 6-17. PTDDD Register Field Descriptions**

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

**6.4.4.3 Port D Pull Enable Register (PTDPE)****Figure 6-19. Internal Pull Enable for Port D Register (PTDPE)****Table 6-18. PTDPE Register Field Descriptions**

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

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

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

Figure 6-20. Slew Rate Enable for Port D Register (PTDSE)

Table 6-19. PTDSE Register Field Descriptions

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

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

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

Figure 6-21. Drive Strength Selection for Port D Register (PTDDS)

Table 6-20. PTDDS Register Field Descriptions

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

#### 6.4.5 Port E Registers

Port E is controlled by the registers listed below.

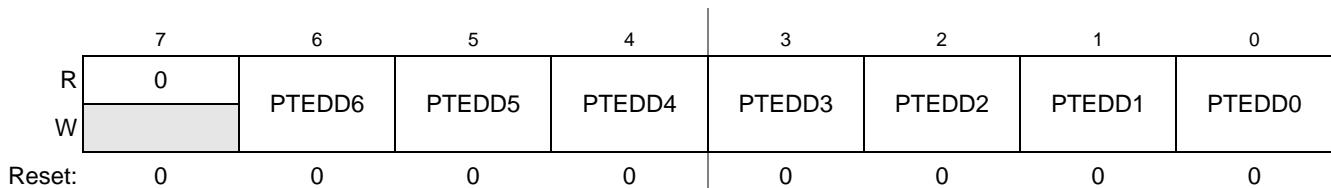
##### 6.4.5.1 Port E Data Register (PTED)

	7	6	5	4		3	2	1	0
R W	0	PTED6	PTED5	PTED4		PTED3	PTED2	PTED1	PTED0
Reset:	0	0	0	0		0	0	0	0

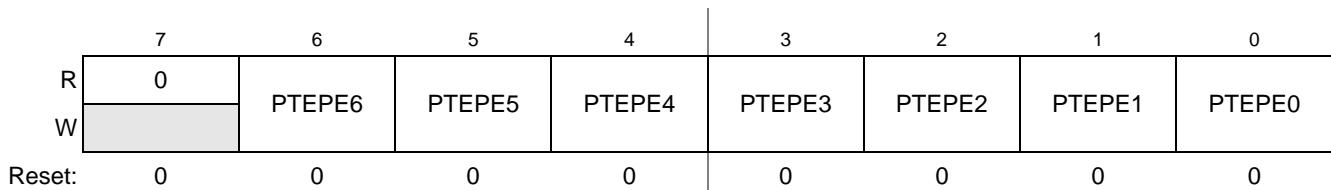
Figure 6-22. Port E Data Register (PTED)

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

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

**6.4.5.2 Port E Data Direction Register (PTEDD)****Figure 6-23. Port E Data Direction Register (PTEDD)****Table 6-22. PTEDD Register Field Descriptions**

Field	Description
6:0 PTEDD[6:0]	<b>Data Direction for Port E Bits</b> — These read/write bits control the direction of port E pins and what is read for PTED reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port E bit n and PTED reads return the contents of PTEDn.

**6.4.5.3 Port E Pull Enable Register (PTEPE)****Figure 6-24. Internal Pull Enable for Port E Register (PTEPE)****Table 6-23. PTEPE Register Field Descriptions**

Field	Description
6:0 PTEPE[6:0]	<b>Internal Pull Enable for Port E Bits</b> — Each of these control bits determines if the internal pull-up device is enabled for the associated PTE pin. For port E pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port E bit n. 1 Internal pull-up device enabled for port E bit n.

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

	7	6	5	4	3	2	1	0
R	0	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-25. Slew Rate Enable for Port E Register (PTESE)

Table 6-24. PTESE Register Field Descriptions

Field	Description
6:0 PTESE[6:0]	<b>Output Slew Rate Enable for Port E Bits</b> — Each of these control bits determines if the output slew rate control is enabled for the associated PTE pin. For port E pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port E bit n. 1 Output slew rate control enabled for port E bit n.

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

	7	6	5	4	3	2	1	0
R	0	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-26. Drive Strength Selection for Port E Register (PTEDS)

Table 6-25. PTEDS Register Field Descriptions

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

#### 6.4.6 Port F Registers

Port F is controlled by the registers listed below.

### 6.4.6.1 Port F Data Register (PTFD)

	7	6	5	4		3	2	1	0
R	0	0	0	0	0	0	PTFD2	PTFD1 <sup>1</sup>	PTFD0 <sup>2</sup>
W						0	0	0	0
Reset:	0	0	0	0	0	0	0	0	0

Figure 6-27. Port F Data Register (PTFD)

<sup>1</sup> Reads of bit PTFD1 always return the contents of PTFD1, regardless of the value stored in bit PTFDD1.

<sup>2</sup> Reads of bit PTFD0 always return the contents of PTFD0, regardless of the value stored in bit PTFDD0.

Table 6-26. PTFD Register Field Descriptions

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

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

	7	6	5	4		3	2	1	0
R	0	0	0	0	0	0	PTFDD2	PTFDD1 <sup>1</sup>	PTFDD0 <sup>2</sup>
W						0	0	0	0
Reset:	0	0	0	0	0	0	0	0	0

Figure 6-28. Port F Data Direction Register (PTFDD)

<sup>1</sup> Writing to PTFDD1 has no effect on the output-only PTF1 pin.

<sup>2</sup> Writing to PTFDD0 has no effect on the output-only PTF0 pin.

Table 6-27. PTFDD Register Field Descriptions

Field	Description
2:0 PTFDD[2:0]	<b>Data Direction for Port F Bits</b> — These read/write bits control the direction of port F pins and what is read for PTFD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port F bit n and PTFD reads return the contents of PTFDn.

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	PTFPE2	PTFPE1 <sup>1</sup>	PTFPE0 <sup>2</sup>
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-29. Internal Pull Enable for Port F Register (PTFPE)

<sup>1</sup> Setting PTFPE1 enables the pullup on PTF1 when PTFD1 is set and PTF1 is controlling the PTF1/RESET pin.

<sup>2</sup> Writing to PTFPE0 has no effect on the output only PTF0 pin.

Table 6-28. PTFPE Register Field Descriptions

Field	Description							
2:0 PTFPE[2:0]	<b>Internal Pull Enable for Port F Bits</b> — Each of these control bits determines if the internal pull-up device is enabled for the associated PTF pin. 0 Internal pull-up device disabled for port F bit n. 1 Internal pull-up device enabled for port F bit n.							

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	PTFSE2	PTFSE1 <sup>1</sup>	PTFSE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-30. Slew Rate Enable for Port F Register (PTFSE)

<sup>1</sup> Writing to PTFSE1 has no effect on the PTF1 pin.

Table 6-29. PTFSE Register Field Descriptions

Field	Description							
2:0 PTFSE[2:0]	<b>Output Slew Rate Enable for Port F Bits</b> — Each of these control bits determines if the output slew rate control is enabled for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port F bit n. 1 Output slew rate control enabled for port F bit n.							

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

	7	6	5	4		3	2	1	0
R	0	0	0	0	0	PTFDS2	PTFDS1 <sup>1</sup>	PTFDS0	
W					0	0	0	0	0
Reset:	0	0	0	0		0	0	0	0

Figure 6-31. Drive Strength Selection for Port F Register (PTFDS)

<sup>1</sup> Writing to PTFDS1 has no effect on the PTF1 pin.

Table 6-30. PTFDS Register Field Descriptions

Field	Description
2:0 PTFDS[2:0]	<b>Output Drive Strength Selection for Port F Bits</b> — Each of these control bits selects between low and high output drive for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port F bit n. 1 High output drive strength selected for port F bit n.

# Chapter 7

## Keyboard Interrupt (S08KBIv2)

### 7.1 Introduction

The keyboard interrupt (KBI) module provides up to eight independently enabled external interrupt sources. Some MCUs may have more than one KBI, so references to modules and register names include a placeholder character to identify which KBI module (use x as placeholder) is being referenced. In cases where only one KBI is available on a particular device, the x placeholder is omitted. Each KBI is associated with multiple pins, so a placeholder character is used to identify which KBI pin (use n as placeholder) is being referenced.

The MC9S08MP16 Series has three KBI modules, referred to as KBI1, KBI2, and KBI3. Each KBI module has up to eight interrupt sources.

#### 7.1.1 KBI Clock Gating

The bus clock to the KBI1, KBI2, and KBI3 modules can be gated on and off using the KBI bit in SCGC2. This bit is set after any reset, which enables the bus clock to these modules. To conserve power, this bit can be cleared to disable the clock to these modules when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

## 7.1.2 Features

The KBI features include:

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

## 7.1.3 Modes of Operation

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

### 7.1.3.1 KBI in Wait Mode

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

### 7.1.3.2 KBI in Stop Modes

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

During stop2 mode, the KBI is disabled. Upon wake-up from stop2 mode, the KBI module will be in the reset state.

### 7.1.3.3 KBI in Active Background Mode

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

## 7.1.4 Block Diagram

The block diagram for the keyboard interrupt module is shown [Figure 7-1](#).

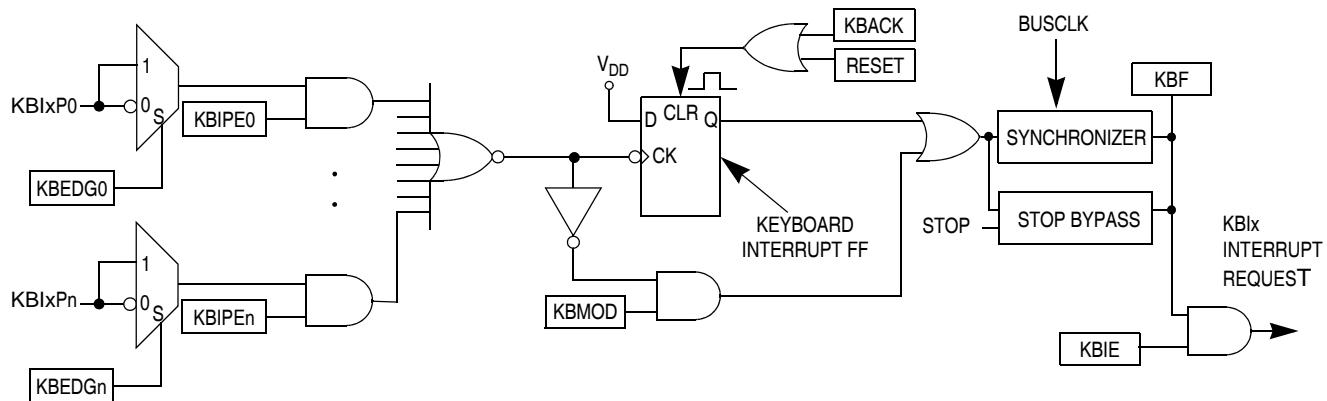


Figure 7-1. KBI Block Diagram

## 7.2 External Signal Description

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

The port association for the KBI1, KBI2, and KBI3 modules is listed in [Table 7-1 – Table 7-3](#).

Table 7-1. KBI1 Pin Mapping

Port pin	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
KBI1 pin	KBI1P7	KBI1P6	KBI1P5	KBI1P4	KBI1P3	KBI1P2	KBI1P1	KBI1P0

Table 7-2. KBI2 Pin Mapping

Port pin	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
KBI2 pin	KBI2P7	KBI2P6	KBI2P5	KBI2P4	KBI2P3	KBI2P2	KBI2P1	KBI2P0

Table 7-3. KBI3 Pin Mapping

Port pin	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
KBI3 pin	KBI3P7	KBI3P6	KBI3P5	KBI3P4	KBI3P3	KBI3P2	KBI3P1	KBI3P0

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

### 7.3.1 KBI Interrupt Status and Control Register (KBIxSC)

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

Figure 7-2. KBI Interrupt Status and Control Register (KBIxSC)

Table 7-4. KBIxSC Register Field Descriptions

Field	Description
3 KBF	<b>KBI Interrupt Flag</b> — KBF indicates when a KBI interrupt is detected. Writes have no effect on KBF. 0 No KBI interrupt detected. 1 KBI interrupt detected.
2 KBACK	<b>KBI Interrupt Acknowledge</b> — Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	<b>KBI Interrupt Enable</b> — KBIE determines whether a KBI interrupt is requested. 0 KBI interrupt request not enabled. 1 KBI interrupt request enabled.
0 KBIMOD	<b>KBI Detection Mode</b> — KBIMOD (along with the KBIES bits) controls the detection mode of the KBI interrupt pins. 0 KBI pins detect edges only. 1 KBI pins detect both edges and levels.

### 7.3.2 KBI Interrupt Pin Select Register (KBIxPE)

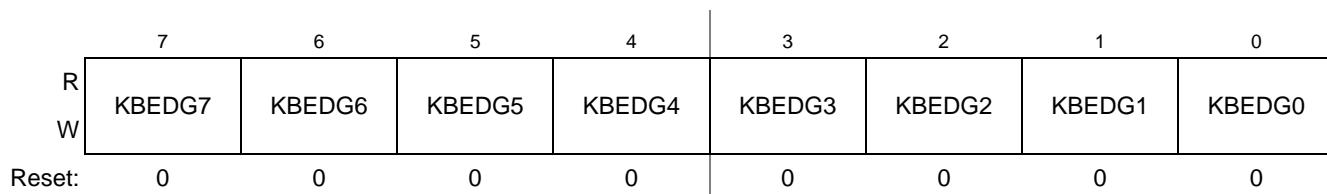
	7	6	5	4	3	2	1	0
R	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 7-3. KBI Interrupt Pin Select Register (KBIxPE)

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

Field	Description
7:0 KBIPE[7:0]	<b>KBI Interrupt Pin Selects</b> — Each of the KBIPEn bits enable the corresponding KBI interrupt pin. 0 Pin not enabled as interrupt. 1 Pin enabled as interrupt.

### 7.3.3 KBI Interrupt Edge Select Register (KBIxES)

**Figure 7-4. KBI Edge Select Register (KBIxES)****Table 7-6. KBIxES Register Field Descriptions**

Field	Description
7:0 KBEDG[7:0]	<b>KBI Edge Selects</b> — Each of the KBEDGn bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if the associated PTxPEN bit is set. 0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation. 1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation.

## 7.4 Functional Description

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

Writing to the KBIPEn bits in the keyboard interrupt pin enable register (KBIxPE) independently enables or disables each port pin. Each port can be configured as edge sensitive or edge and level sensitive based on the KBIMOD bit in the keyboard interrupt status and control register (KBIxSC). Edge sensitivity can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the KBEDGn bits in the keyboard interrupt edge select register (KBIxES).

### 7.4.1 Edge Only Sensitivity

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

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.

### 7.4.2 Edge and Level Sensitivity

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

### 7.4.3 Pull-up/Pull-down Resistors

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

### 7.4.4 Keyboard Interrupt Initialization

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

1. Mask interrupts by clearing KBIE in KBIXSC.
2. Select the pin polarity by setting the appropriate KBEDGn bits in KBIXES.
3. If using internal pull-up/pull-down device, configure the associated port pull enable bits in PTxPE.
4. Enable the interrupt pins by setting the appropriate KBIPEn bits in KBIXPE.
5. Write to KBACK in KBIXSC to clear any false interrupts.
6. Set KBIE in KBIXSC to enable interrupts.

# Chapter 8

## Central Processor Unit (S08CPUV5)

### 8.1 Introduction

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

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

#### 8.1.1 Features

Features of the HCS08 CPU include:

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

## 8.2 Programmer's Model and CPU Registers

Figure 8-1 shows the five CPU registers. CPU registers are not part of the memory map.

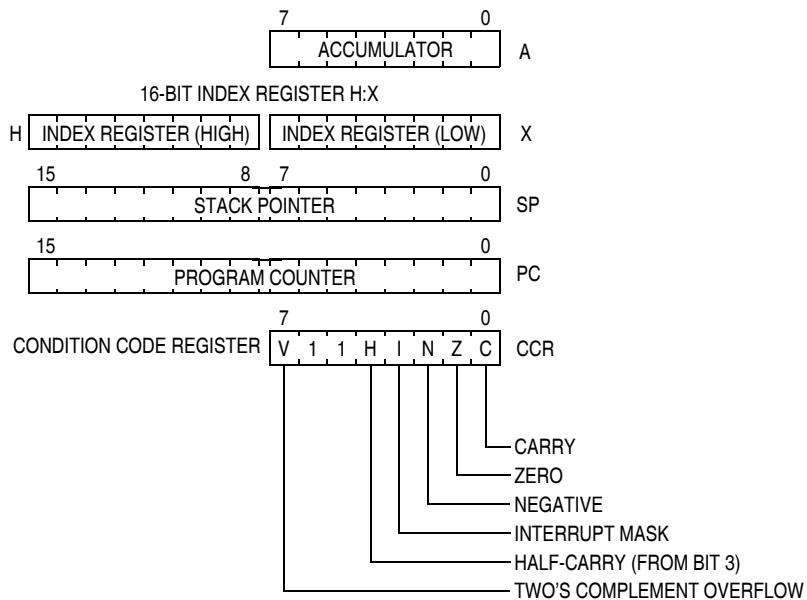


Figure 8-1. CPU Registers

### 8.2.1 Accumulator (A)

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

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

### 8.2.2 Index Register (H:X)

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

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

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

### 8.2.3 Stack Pointer (SP)

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

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

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

### 8.2.4 Program Counter (PC)

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

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

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

### 8.2.5 Condition Code Register (CCR)

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

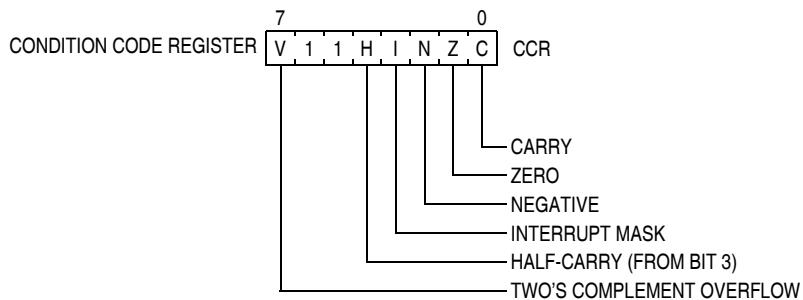


Figure 8-2. Condition Code Register

Table 8-1. CCR Register Field Descriptions

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

## 8.3 Addressing Modes

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

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

### 8.3.1 Inherent Addressing Mode (INH)

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

### 8.3.2 Relative Addressing Mode (REL)

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

### 8.3.3 Immediate Addressing Mode (IMM)

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

### 8.3.4 Direct Addressing Mode (DIR)

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

### 8.3.5 Extended Addressing Mode (EXT)

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

### 8.3.6 Indexed Addressing Mode

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 8.4 Special Operations

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

### 8.4.1 Reset Sequence

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

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

### 8.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

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

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

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

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

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

### 8.4.3 Wait Mode Operation

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

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

### 8.4.4 Stop Mode Operation

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

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

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

## 8.4.5 BGND Instruction

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

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

## 8.5 HCS08 Instruction Set Summary

Table 8-2 provides a summary of the HCS08 instruction set in all possible addressing modes. The table shows operand construction, execution time in internal bus clock cycles, and cycle-by-cycle details for each addressing mode variation of each instruction.

**Table 8-2. Instruction Set Summary (Sheet 1 of 9)**

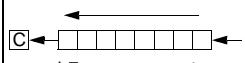
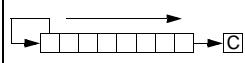
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V	I
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry A ← (A) + (M) + (C)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 ee ff E9 ff F9 9E D9 ee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

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

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

Table 8-2. Instruction Set Summary (Sheet 3 of 9)

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

Table 8-2. Instruction Set Summary (Sheet 4 of 9)

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

Table 8-2. Instruction Set Summary (Sheet 5 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V	I
INC opr8a INCA INCX INC oprx8,X INC ,X INC oprx8,SP	Increment M ← (M) + \$01 A ← (A) + \$01 X ← (X) + \$01 M ← (M) + \$01 M ← (M) + \$01 M ← (M) + \$01	DIR INH INH IX1 IX SP1	3C dd 4C 5C 6C ff 7C 9E 6C ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	P 1 1 -	- P P -
JMP opr8a JMP opr16a JMP oprx16,X JMP oprx8,X JMP ,X	Jump PC ← Jump Address	DIR EXT IX2 IX1 IX	BC dd CC hh ll DC ee ff EC ff FC	3 4 4 3 3	ppp pppp pppp ppp ppp	- 1 1 -	-----
JSR opr8a JSR opr16a JSR oprx16,X JSR oprx8,X JSR ,X	Jump to Subroutine PC ← (PC) + n (n = 1, 2, or 3) Push (PCL); SP ← (SP) - \$0001 Push (PCH); SP ← (SP) - \$0001 PC ← Unconditional Address	DIR EXT IX2 IX1 IX	BD dd CD hh ll DD ee ff ED ff FD	5 6 6 5 5	ssppp pssppp pssppp ssppp ssppp	- 1 1 -	-----
LDA #opr8i LDA opr8a LDA opr16a LDA oprx16,X LDA oprx8,X LDA ,X LDA oprx16,SP LDA oprx8,SP	Load Accumulator from Memory A ← (M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 ii B6 dd C6 hh ll D6 ee ff E6 ff F6 9E D6 ee ff 9E E6 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 -	- P P -
LDHX #opr16i LDHX opr8a LDHX opr16a LDHX ,X LDHX oprx16,X LDHX oprx8,X LDHX oprx8,SP	Load Index Register (H:X) H:X ← (M:M + \$0001)	IMM DIR EXT IX IX2 IX1 SP1	45 jj kk 55 dd 32 hh ll 9E AE 9E BE ee ff 9E CE ff 9E FE ff	3 4 5 5 6 5 5	ppp rrpp prrpp prrfp pprrpp prrpp prrpp	0 1 1 -	- P P -
LDX #opr8i LDX opr8a LDX opr16a LDX oprx16,X LDX oprx8,X LDX ,X LDX oprx16,SP LDX oprx8,SP	Load X (Index Register Low) from Memory X ← (M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE ii BE dd CE hh ll DE ee ff EE ff FE 9E DE ee ff 9E EE ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 -	- P P -
LSL opr8a LSLA LSLX LSL oprx8,X LSL ,X LSL oprx8,SP	Logical Shift Left 	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	P 1 1 -	- P P P
LSR opr8a LSRA LSRX LSR oprx8,X LSR ,X LSR oprx8,SP	Logical Shift Right 	DIR INH INH IX1 IX SP1	34 dd 44 54 64 ff 74 9E 64 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	P 1 1 -	- 0 P P

Table 8-2. Instruction Set Summary (Sheet 6 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V	I
MOV opr8a,opr8a MOV opr8a,X+ MOV #opr8i,opr8a MOV ,X+,opr8a	Move (M) <sub>destination</sub> ← (M) <sub>source</sub> In IX+/DIR and DIR/IX+ Modes, H:X ← (H:X) + \$0001	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E dd dd 5E dd 6E ii dd 7E dd	5 5 4 5	rwpwpp rfwpp pwpp rfwpp	0 1 1 -	- P P -
MUL	Unsigned multiply X:A ← (X) × (A)	INH	42	5	fffffp	- 1 1 0	- - - 0
NEG opr8a NEGA NEGX NEG oprx8,X NEG ,X NEG oprx8,SP	Negate (Two's Complement) M ← - (M) = \$00 - (M) A ← - (A) = \$00 - (A) X ← - (X) = \$00 - (X) M ← - (M) = \$00 - (M) M ← - (M) = \$00 - (M) M ← - (M) = \$00 - (M)	DIR INH INH IX1 IX SP1	30 dd 40 50 60 ff 70 9E 60 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	P 1 1 -	- P P P
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	- 1 1 -	- - - -
NSA	Nibble Swap Accumulator A ← (A[3:0]:A[7:4])	INH	62	1	p	- 1 1 -	- - - -
ORA #opr8i ORA opr8a ORA opr16a ORA oprx16,X ORA oprx8,X ORA ,X ORA oprx16,SP ORA oprx8,SP	Inclusive OR Accumulator and Memory A ← (A)   (M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA ii BA dd CA hh ll DA ee ff EA ff FA 9E DA ee ff 9E EA ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 -	- P P -
PSHA	Push Accumulator onto Stack Push (A); SP ← (SP) - \$0001	INH	87	2	sp	- 1 1 -	- - - -
PSHH	Push H (Index Register High) onto Stack Push (H); SP ← (SP) - \$0001	INH	8B	2	sp	- 1 1 -	- - - -
PSHX	Push X (Index Register Low) onto Stack Push (X); SP ← (SP) - \$0001	INH	89	2	sp	- 1 1 -	- - - -
PULA	Pull Accumulator from Stack SP ← (SP + \$0001); Pull (A)	INH	86	3	ufp	- 1 1 -	- - - -
PULH	Pull H (Index Register High) from Stack SP ← (SP + \$0001); Pull (H)	INH	8A	3	ufp	- 1 1 -	- - - -
PULX	Pull X (Index Register Low) from Stack SP ← (SP + \$0001); Pull (X)	INH	88	3	ufp	- 1 1 -	- - - -
ROL opr8a ROL A ROLX ROL oprx8,X ROL ,X ROL oprx8,SP	Rotate Left through Carry 	DIR INH INH IX1 IX SP1	39 dd 49 59 69 ff 79 9E 69 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	P 1 1 -	- P P P
ROR opr8a RORA RORX ROR oprx8,X ROR ,X ROR oprx8,SP	Rotate Right through Carry 	DIR INH INH IX1 IX SP1	36 dd 46 56 66 ff 76 9E 66 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	P 1 1 -	- P P P

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

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

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

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

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

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

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

*n* Any label or expression that evaluates to a single integer in the range 0-7.

*opr8i* Any label or expression that evaluates to an 8-bit immediate value.

*opr16i* Any label or expression that evaluates to a 16-bit immediate value.

*opr8a* Any label or expression that evaluates to an 8-bit direct-page address (\$00xx).

*opr16a* Any label or expression that evaluates to a 16-bit address.

*opr8x* Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.

*opr16x* Any label or expression that evaluates to a 16-bit value, used for indexed addressing.

*rel* Any label or expression that refers to an address that is within –128 to +127 locations from the start of the next instruction.

#### Operation Symbols:

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

#### Addressing Modes:

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

#### Cycle-by-Cycle Codes:

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

#### CCR Bits:

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

#### CCR Effects:

p	Set or cleared
–	Not affected
U	Undefined

Table 8-3. Opcode Map (Sheet 1 of 2)

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

INH Inherent  
 IMM Immediate  
 DIR Direct  
 EXT Extended  
 DD DIR to DIR  
 IX+D IX+ to DIR

REL Indexed  
 IX1 Indexed, No Offset  
 IX2 Indexed, 8-Bit Offset  
 IX3 Indexed, 16-Bit Offset  
 IMD IMM to DIR  
 DIX+ DIR to IX+

SP1 Stack Pointer, 8-Bit Offset  
 SP2 Stack Pointer, 16-Bit Offset  
 IX+ Indexed, No Offset with Post Increment  
 IX1+ Indexed, 1-Byte Offset with Post Increment

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

Table 8-3. Opcode Map (Sheet 2 of 2)

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

INH Inherent  
 IMM Immediate  
 DIR Direct  
 EXT Extended  
 DD DIR to DIR  
 IX-D IX+ to DIR  
 IX+ DIR to IX+

REL Relative  
 IX Indexed, No Offset  
 IX1 Indexed, 8-Bit Offset  
 IX2 Indexed, 16-Bit Offset  
 IMD IMM to DIR  
 DIX+ DIR to IX+

SP1 Stack Pointer, 8-Bit Offset  
 SP2 Stack Pointer, 16-Bit Offset  
 IX+ Indexed, No Offset with Post Increment  
 IX1+ Indexed, 1-Byte Offset with Post Increment

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

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

9E60 6 NEG 3 SP1	HCS08 Cycles Instruction Mnemonic Addressing Mode
------------------------	---------------------------------------------------------

# Chapter 9

## Analog-to-Digital Converter (S08ADC12V1)

### 9.1 Introduction

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

Figure 9-2 shows the MC9S08MP16 with the ADC module highlighted.

### 9.2 Module Configurations

This section provides device-specific information for configuring the ADC on MC9S08MP16 Series.

#### 9.2.1 ADC Clock Gating

The bus clock to the ADC can be gated on and off using the ADC bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ADC bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 9.2.2 Analog Supply and Voltage Reference Connections

The VDDAD power inputs and the VREFH reference input for the ADC are supplied by the V<sub>DDA</sub>/V<sub>REFH</sub> pin on the MC9S08MP16 Series devices.

The VSSAD power inputs and the VREFL reference input for the ADC are supplied by the V<sub>SSA</sub>/V<sub>REFL</sub> pin on the MC9S08MP16 Series devices.

#### 9.2.3 Configurations for Stop Modes

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

The ADC, if enabled, must be configured to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements.

#### 9.2.4 Channel Assignments

The ADC channel assignments for the MC9S08MP16 Series devices are shown in [Table 9-1](#). Reserved channels convert to an unknown value.

The ADC on MC9S08MP16 contains two analog pin enable registers (APCTL1 and APCTL2; APCTL3 is not supported on this device). The bits in these registers (APCTL1[ADPC7–0] and

APCTL2[ADPC12–8]) control the ADC channels that have an associated pin on the device. ADC channels inputs from internal sources do not have an associated bit in the APCTL registers.

**Table 9-1. ADC Channel Assignment**

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	AD0	PTB0/ADP0	ADPC0	10000	AD16	$V_{REFL}$	—
00001	AD1	PTB1/ADP1	ADPC1	10001	AD17	Reserved	—
00010	AD2	PTB2/ADP2/PGA+	ADPC2	10010	AD18	Reserved	—
00011	AD3	PTB3/ADP3/PGA-	ADPC3	10011	AD19	Reserved	—
00100	AD4	PTB4/ADP4	ADPC4	10100	AD20	Reserved	—
00101	AD5	PTB5/ADP5	ADPC5	10101	AD21	Reserved	—
00110	AD6	PTB6/ADP6	ADPC6	10110	AD22	Reserved	—
00111	AD7	PTB7/ADP7	ADPC7	10111	AD23	Reserved	—
01000	AD8	PTE0/ADP8	ADPC8	11000	AD24	Reserved	—
01001	AD9	PTE1/ADP9	ADPC9	11001	AD25	Reserved	—
01010	AD10	PTE2/ADP10	ADPC10	11010	AD26	Temperature Sensor <sup>1</sup>	—
01011	AD11	PTE3/ADP11	ADPC11	11011	AD27	Internal Bandgap	—
01100	AD12	PTE4/ADP12	ADPC12	11100	—	Reserved	—
01101	AD13	PGA	—	11101	$V_{REFH}$ <sup>2</sup>	$V_{REFH}$	—
01110	AD14	Reserved	—	11110	$V_{REFL}$ <sup>3</sup>	$V_{REFL}$	—
01111	AD15	Reserved	—	11111	Module Disabled	None	—

<sup>1</sup> For information, see [Section 9.2.7, “Temperature Sensor”](#).

<sup>2</sup>  $V_{REFH}$  is tied to  $V_{DDA}$  internally

<sup>3</sup>  $V_{REFL}$  is tied to  $V_{SSA}$  internally

## NOTE

Selecting the internal bandgap channel requires BGBE=1 in SPMSC1. See [Section 5.8.7, “System Power Management Status and Control 1 Register \(SPMSC1\).”](#) For value of bandgap voltage reference see the data sheet.

### 9.2.5 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. The alternate clock, ALTCLK, input for the MC9S08MP16 Series MCU devices is not implemented.

### 9.2.6 Hardware Trigger

In run mode and wait mode, the ADC hardware trigger, ADHWT, is initiated by either a PGA trigger output or the PDB1 Trigger B output. When the PGA is disabled, the PGA is bypassed and the PDB Trigger A output triggers the ADC directly. [Figure 9-1](#) shows the hardware trigger configuration for MC9S08MP16 Series devices. For more details on PGA and PDB triggering see [Chapter 19, “Programmable Gain Amplifier \(S08PGAV1\)”](#) and [Chapter 18, “Programmable Delay Block \(S08PDBV1\)”](#).

In stop mode, the RTC overflow bypasses the PDB1 and PGA logic and is used as the ADC hardware trigger, ADHWT.

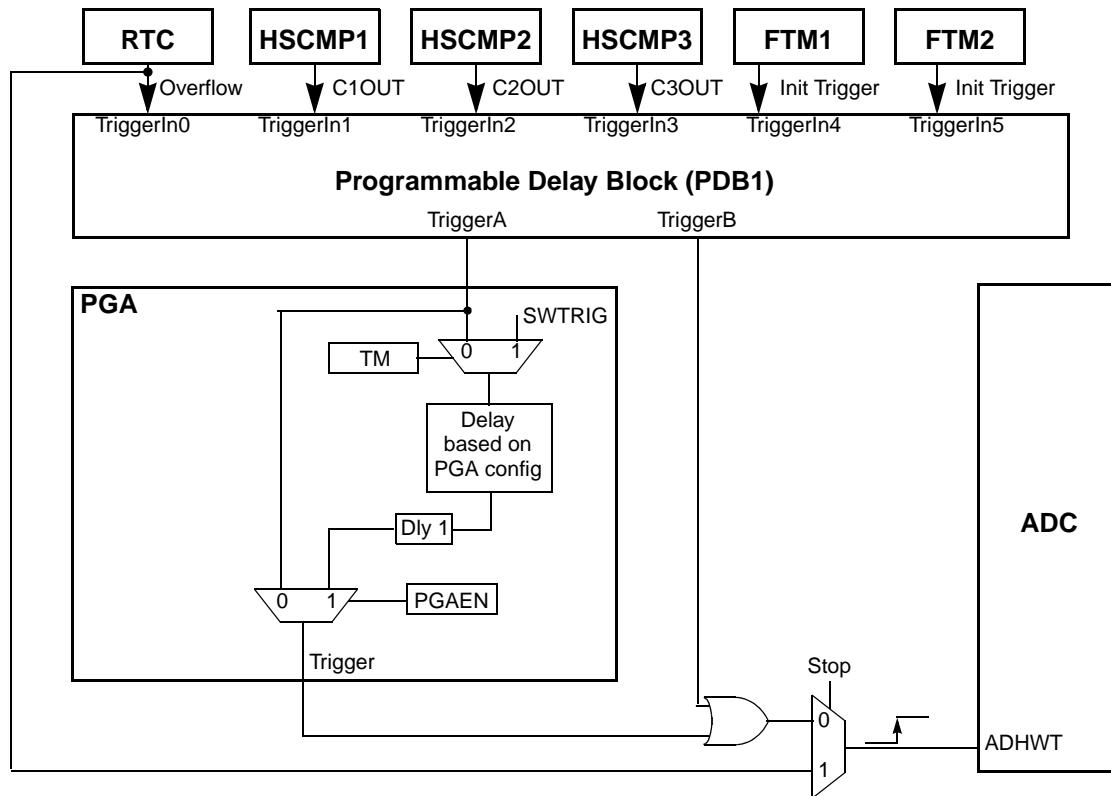


Figure 9-1. ADC Hardware Trigger Block Diagram

### 9.2.7 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. [Equation 9-1](#) provides an approximate transfer function of the temperature sensor for  $V_{DDA} = 5.0V$ , Temp =  $25^{\circ}\text{C}$ , using the ADC at  $f_{ADCK} = 1.0\text{MHz}$  and configured for long sample.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \quad \text{Eqn. 9-1}$$

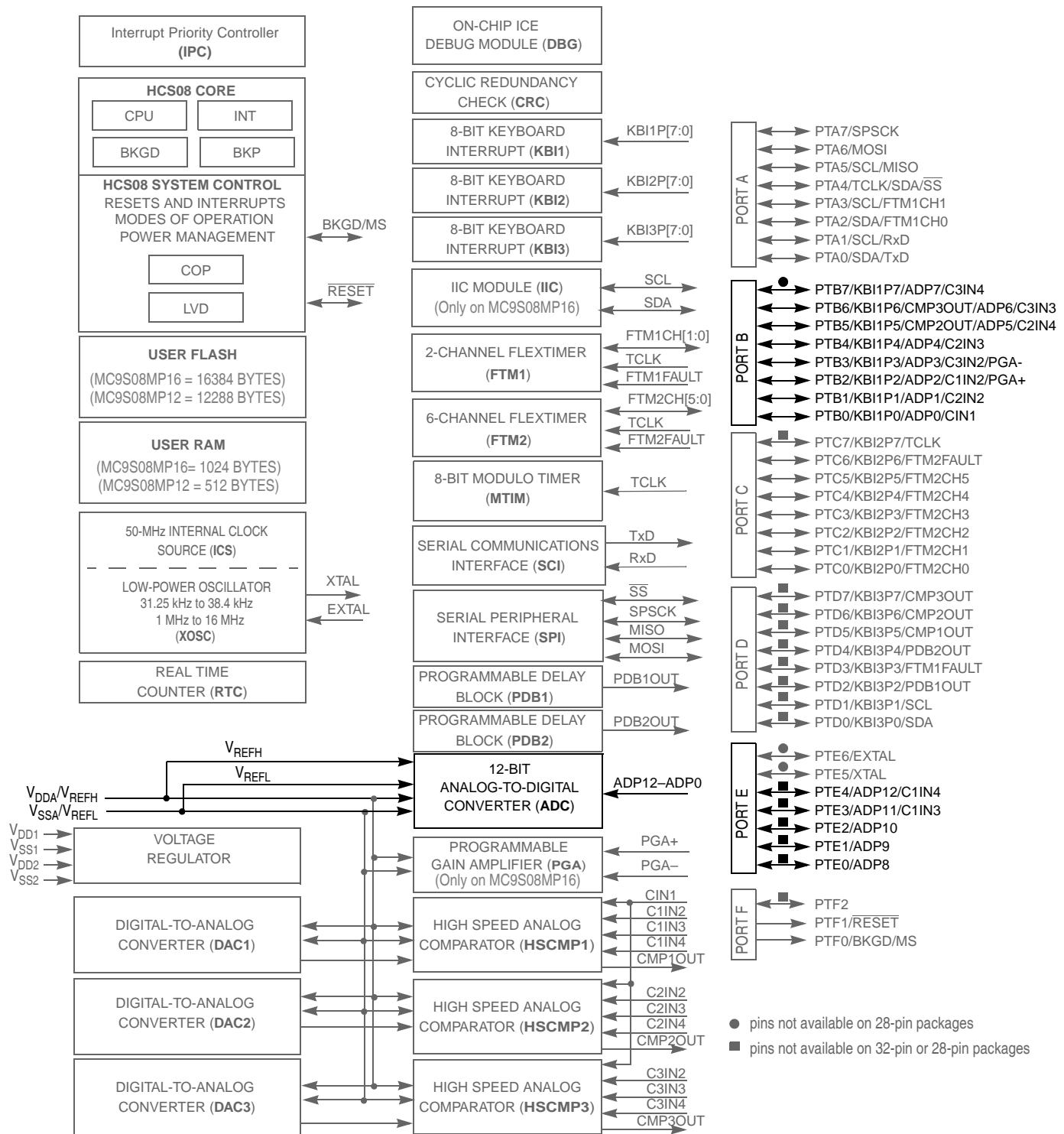
where:

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

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 9-1](#). If  $V_{TEMP}$  is less than  $V_{TEMP25}$  the hot slope value is applied in [Equation 9-1](#).

For more information on using the temperature sensor, consult AN3031.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

$V_{DD2}$  pad is tied internally on 32-pin and 28-pin packages,

$V_{SS2}$  pad is tied internally on 28-pin packages

**Figure 9-2. MC9S08MP16 Series Block Diagram Highlighting ADC Block and Pins**

## 0.0.1 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bit resolution
- Up to 28 analog inputs<sup>1</sup>
- Output formatted in 12-, 10-, or 8-bit right-justified unsigned format
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in wait or stop3 modes for lower noise operation
- Asynchronous clock source for lower noise operation
- Selectable asynchronous hardware conversion trigger
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value
- Temperature sensor

## 0.0.2 ADC Module Block Diagram

Figure 0-1 provides a block diagram of the ADC module.

---

1. Number of analog inputs varies according to the device and may be from external or internal sources. Refer to the introduction section to this chapter for AD0–AD27 channel input assignments.

## Analog-to-Digital Converter (S08ADC12V1)

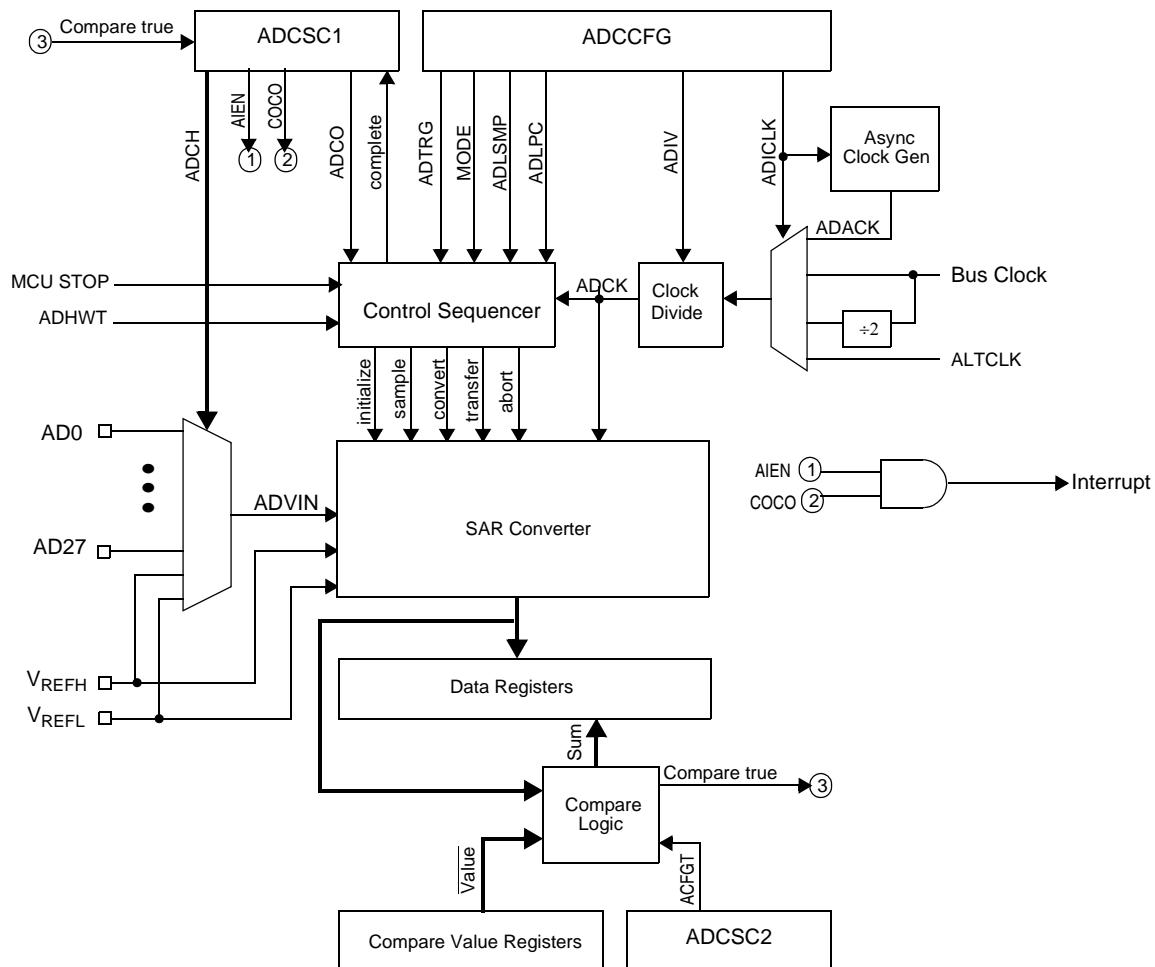


Figure 0-1. ADC Block Diagram

## 0.1 External Signal Description

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

Table 0-1. Signal Properties

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

### 0.1.1 Analog Power ( $V_{DDA}$ )

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

### 0.1.2 Analog Ground ( $V_{SSA}$ )

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

### 0.1.3 Voltage Reference High ( $V_{REFH}$ )

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

### 0.1.4 Voltage Reference Low ( $V_{REFL}$ )

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

### 0.1.5 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.

## 0.2 Register Definition

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

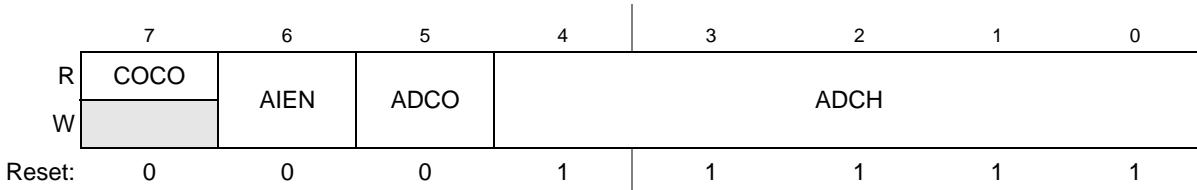
- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin control registers, APCTLx<sup>1</sup>

---

1. Number of APCTLx registers depends on the number of external analog inputs available on the device. Please refer to the introduction of this module for external analog input assignments.

## 0.2.1 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 0-2. Status and Control Register (ADCSC1)**

**Table 0-2. ADCSC1 Field Descriptions**

Field	Description
7 COCO	Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared when ADCSC1 is written or when ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	Interrupt Enable AIEN enables 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 enables 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 that selects one of the input channels. The input channels are detailed in <a href="#">Table 0-3</a> . The successive approximation converter subsystem is turned off when the channel select bits are all set. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

**Table 0-3. Input Channel Select**

ADCH	Input Select
00000–01111	AD0–15
10000–11011	AD16–27
11100	Reserved
11101	V <sub>REFH</sub>

**Table 0-3. Input Channel Select (continued) (continued)**

ADCH	Input Select
11110	$V_{REFL}$
11111	Module disabled

## 0.2.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register controls the compare function, conversion trigger, and conversion active of the ADC module.

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R <sup>1</sup>	R <sup>1</sup>
W								
Reset:	0	0	0	0	0	0	0	0

**Figure 0-3. Status and Control Register 2 (ADCSC2)**

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

**Table 0-4. ADCSC2 Register Field Descriptions**

Field	Description
7 ADACT	Conversion Active. Indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	Conversion Trigger Select. Selects the type of trigger used for initiating a conversion. Two types of triggers are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	Compare Function Enable. Enables the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	Compare Function Greater Than Enable. Configures the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare value 1 Compare triggers when input is greater than or equal to compare value

### 0.2.3 Data Result High Register (ADCRH)

In 12-bit operation, ADCRH contains the upper four bits of 12-bit conversion data. In 10-bit operation, ADCRH contains the upper two bits of 10-bit conversion data. In 12-bit and 10-bit mode, ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. When configured for 10-bit mode, ADR[11:10] are cleared. When configured for 8-bit mode, ADR[11:8] are cleared.

When automatic compare is not enabled, the value stored in ADCRH are the upper bits of the conversion result. When automatic compare is enabled, the conversion result is manipulated as described in [Section 0.3.5, “Automatic Compare Function”](#) prior to storage in ADCRH:ADCRL registers.

In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion data into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, the intermediate conversion data is lost. In 8-bit mode, there is no interlocking with ADCRL. If the MODE bits are changed, any data in ADCRH becomes invalid.

	7	6	5	4		3	2	1	0
R	0	0	0	0	ADR11	ADR10	ADR9	ADR8	
W									
Reset:	0	0	0	0	0	0	0	0	0

Figure 0-4. Data Result High Register (ADCRH)

### 0.2.4 Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of a 12-bit or 10-bit conversion data, and all eight bits of 8-bit conversion data. ADCRL is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met.

When automatic compare is not enabled, the value stored in ADCRL is the lower eight bits of the conversion result. When automatic compare is enabled, the conversion result is manipulated as described in [Section 0.3.5, “Automatic Compare Function”](#) prior to storage in ADCRH:ADCRL registers.

In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion data into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, the intermediate conversion data is lost. In 8-bit mode, there is no interlocking with ADCRH. If the MODE bits are changed, any data in ADCRL becomes invalid.

	7	6	5	4		3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	
W									
Reset:	0	0	0	0	0	0	0	0	0

Figure 0-5. Data Result Low Register (ADCRL)

## 0.2.5 Compare Value High Register (ADCCVH)

In 12-bit mode, the ADCCVH register holds the upper four bits of the 12-bit compare value. When the compare function is enabled, these bits are compared to the upper four bits of the result following a conversion in 12-bit mode.

	7	6	5	4		3	2	1	0
R	0	0	0	0		ADCV11	ADCV10	ADCV9	ADCV8
W						0	0	0	0
Reset:	0	0	0	0		0	0	0	0

Figure 0-6. Compare Value High Register (ADCCVH)

In 10-bit mode, the ADCCVH register holds the upper two bits of the 10-bit compare value (ADCV[9:8]). These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.

In 8-bit mode, ADCCVH is not used during compare.

## 0.2.6 Compare Value Low Register (ADCCVL)

This register holds the lower eight bits of the 12-bit or 10-bit compare value or all eight bits of the 8-bit compare value. When the compare function is enabled, bits ADCV[7:0] are compared to the lower eight bits of the result following a conversion in 12-bit, 10-bit or 8-bit mode.

	7	6	5	4		3	2	1	0
R	ADCV7	ADCV6	ADCV5	ADCV4		ADCV3	ADCV2	ADCV1	ADCV0
W						0	0	0	0
Reset:	0	0	0	0		0	0	0	0

Figure 0-7. Compare Value Low Register(ADCCVL)

## 0.2.7 Configuration Register (ADCCFG)

ADCCFG selects the mode of operation, clock source, clock divide, and configures for low power and long sample time.

	7	6	5	4		3	2	1	0
R	ADLPC	ADIV	ADLSMP		MODE				ADICLK
W						0	0	0	0
Reset:	0	0	0	0		0	0	0	0

Figure 0-8. Configuration Register (ADCCFG)

**Table 0-5. ADCCFG Register Field Descriptions**

Field	Description
7 ADLPC	Low-Power Configuration. ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: The power is reduced at the expense of maximum clock speed.
6:5 ADIV	Clock Divide Select. ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. <a href="#">Table 0-6</a> 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 select between 12-, 10-, or 8-bit operation. See <a href="#">Table 0-7</a> .
1:0 ADICLK	Input Clock Select. ADICLK bits select the input clock source to generate the internal clock ADCK. See <a href="#">Table 0-8</a> .

**Table 0-6. Clock Divide Select**

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock $\div 2$
10	4	Input clock $\div 4$
11	8	Input clock $\div 8$

**Table 0-7. Conversion Modes**

MODE	Mode Description
00	8-bit conversion (N=8)
01	12-bit conversion (N=12)
10	10-bit conversion (N=10)
11	Reserved

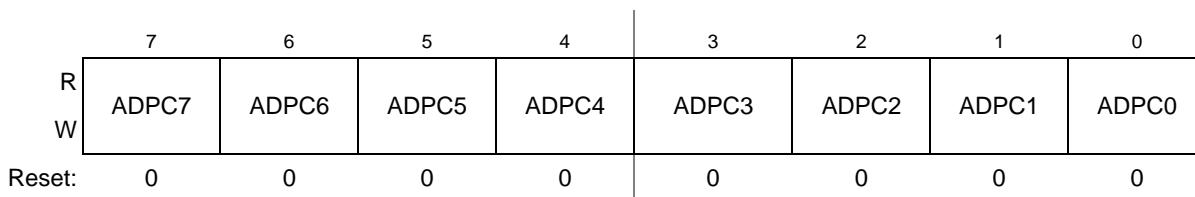
**Table 0-8. Input Clock Select**

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

## 0.2.8 Pin Control 1 Register (APCTL1)

The pin control registers disable the digital interface to the associated MCU pins used as analog inputs to reduce digital noise and improve conversion accuracy. APCTL1 controls the pins associated with channels 0–7 of the ADC module.

Some MCUs may not use all bits implemented in this register. Bits in this register that do not have associated external analog inputs have no control function. Consult the ADC channel assignment in the module introduction.

**Figure 0-9. Pin Control 1 Register (APCTL1)****Table 0-9. APCTL1 Register Field Descriptions**

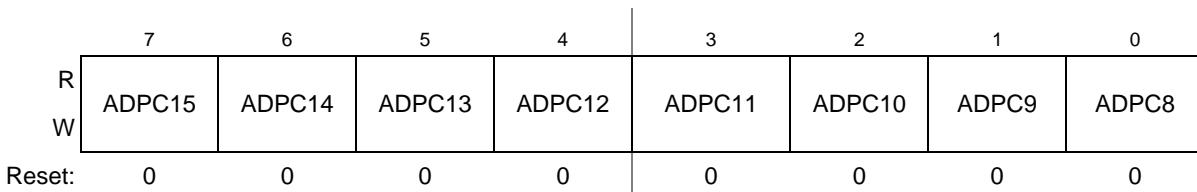
Field	Description
7 ADPC7	ADC Pin Control 7. ADPC7 controls the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	ADC Pin Control 6. ADPC6 controls the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	ADC Pin Control 5. ADPC5 controls 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 controls 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 controls 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 controls the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled

**Table 0-9. APCTL1 Register Field Descriptions (continued)**

Field	Description
1 ADPC1	ADC Pin Control 1. ADPC1 controls 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 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

## 0.2.9 Pin Control 2 Register (APCTL2)

The pin control registers disable the digital interface to the associated MCU pins used as analog inputs to reduce digital noise and improve conversion accuracy. APCTL2 controls channels 8–15 of the ADC module. This register is not implemented on MCUs that do not have associated external analog inputs. Consult the ADC channel assignment in the module introduction for information on availability of this register.

**Figure 0-10. Pin Control 2 Register (APCTL2)****Table 0-10. APCTL2 Register Field Descriptions**

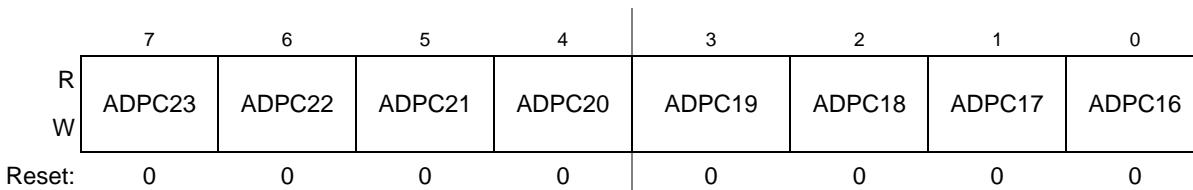
Field	Description
7 ADPC15	ADC Pin Control 15. ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	ADC Pin Control 14. ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	ADC Pin Control 13. ADPC13 controls 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 controls 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 controls 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 controls the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled

**Table 0-10. APCTL2 Register Field Descriptions (continued)**

Field	Description
1 ADPC9	ADC Pin Control 9. ADPC9 controls 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 controls the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

## 0.2.10 Pin Control 3 Register (APCTL3)

The pin control registers disable the digital interface to the associated MCU pins used as analog inputs to reduce digital noise and improve conversion accuracy. APCTL3 controls channels 16–23 of the ADC module. This register is not implemented on MCUs that do not have associated external analog inputs. Consult the ADC channel assignment in the module introduction for information on availability of this register.

**Figure 0-11. Pin Control 3 Register (APCTL3)****Table 0-11. APCTL3 Register Field Descriptions**

Field	Description
7 ADPC23	ADC Pin Control 23. ADPC23 controls 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 controls 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 controls 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 controls 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 controls 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 controls the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled

**Table 0-11. APCTL3 Register Field Descriptions (continued)**

Field	Description
1 ADPC17	ADC Pin Control 17. ADPC17 controls 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 controls the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

## 0.3 Functional Description

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

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. In 12-bit and 10-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 12-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

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

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

### 0.3.1 Clock Select and Divide Control

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

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

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

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

### 0.3.2 Input Select and Pin Control

The pin control registers (APCTLx) disable the digital interface to I/O 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.

### 0.3.3 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.

### 0.3.4 Conversion Control

Conversions can be performed in 12-bit mode, 10-bit mode, or 8-bit mode as determined by the MODE bits. Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

#### 0.3.4.1 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.

### 0.3.4.2 Completing Conversions

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

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 12-bit or 10-bit MODE (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

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

### 0.3.4.3 Aborting Conversions

Any conversion in progress is aborted when:

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

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

### 0.3.4.4 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).

### 0.3.4.5 Sample Time and Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock ( $f_{ADCK}$ ). After the module becomes active, sampling of the input begins. ADLSMP selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the

digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

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

The maximum total conversion time for different conditions is summarized in [Table 0-12](#).

**Table 0-12. 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 or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 $\mu$ s + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 $\mu$ s + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 $\mu$ s + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 $\mu$ s + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

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

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

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

#### NOTE

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

### 0.3.5 Automatic Compare Function

The compare function is enabled by the ACFE bit. The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the compare value (ADCCVH and ADCCVL) is subtracted from the conversion result. When comparing to an upper limit (ACFGT = 1), if the conversion 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. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

The subtract operation of two positive values (the conversion result less the compare value) results in a signed value that is 1-bit wider than the bit-width of the two terms. The final value transferred to the ADCRH and ADCRL registers is the result of the subtraction operation, excluding the sign bit. The value of the sign bit can be derived based on ACFGT control setting. When ACFGT=1, the sign bit of any value stored in ADCRH and ADCRL is always 0, indicating a positive result for the subtract operation. When ACFGT = 1, the sign bit of any result is always 1, indicating a negative result for the subtract operation.

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.

#### NOTE

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

An example of compare operation eases understanding of the compare feature. If the ADC is configured for 10-bit operation, ACFGT=0, and ADCCVH:ADCCVL= 0x200, then a conversion result of 0x080 causes the compare condition to be met and the COCO bit is set. A value of 0x280 is stored in ADCRH:ADCRL. This is signed data without the sign bit and must be combined with a derived sign bit to have meaning. The value stored in ADCRH:ADCRL is calculated as follows.

The value to interpret from the data is (Result – Compare Value) = (0x080 – 0x200) = -0x180. A standard method for handling subtraction is to convert the second term to its 2's complement, and then add the two terms. First calculate the 2's complement of 0x200 by complementing each bit and adding 1. Note that prior to complementing, a sign bit of 0 is added so that the 10-bit compare value becomes a 11-bit signed value that is always positive.

$$\begin{array}{r}
 \%101\ 1111\ 1111 & \text{=> 1's complement of 0x200 compare value} \\
 + & \%1 \\
 \hline
 \%110\ 0000\ 0000 & \text{=> 2's complement of 0x200 compare value}
 \end{array}$$

Then the conversion result of 0x080 is added to 2's complement of 0x200:

$$\begin{array}{r}
 \%000\ 1000\ 0000 \\
 + \%110\ 0000\ 0000 \\
 \hline
 \%110\ 1000\ 0000 & \text{=> Subtraction result is -0x180 in signed 11-bit data}
 \end{array}$$

The subtraction result is an 11-bit signed value. The lower 10 bits (0x280) are stored in ADCRH:ADCRL. The sign bit is known to be 1 (negative) because the ACFG<sub>T</sub>=0, the COCO bit was set, and conversion data was updated in ADCRH:ADCRL.

A simpler way to use the data stored in ADCRH:ADCRL is to apply the following rules. When comparing for upper limit (ACFG<sub>T</sub>=1), the value in ADCRH:ADCRL is a positive value and does not need to be manipulated. This value is the difference between the conversion result and the compare value. When comparing for lower limit (ACFG<sub>T</sub>=0), ADCRH:ADCRL is a negative value without the sign bit. If the value from these registers is complemented and then a value of 1 is added, then the calculated value is the unsigned (i.e., absolute) difference between the conversion result and the compare value. In the previous example, 0x280 is stored in ADCRH:ADCRL. The following example shows how the absolute value of the difference is calculated.

```
%01 0111 1111 <= Complement of 10-bit value stored in ADCRH:ADCRL
+
+           %1
-----
%01 1000 0000<= Unsigned value 0x180 is the absolute value of (Result - Compare Value)
```

### 0.3.6 MCU Wait Mode Operation

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

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

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

### 0.3.7 MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

#### 0.3.7.1 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.

### 0.3.7.2 Stop3 Mode With ADACK Enabled

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

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

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

#### NOTE

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the data transfer blocking mechanism (discussed in [Section 0.3.4.2, “Completing Conversions”](#)) is cleared when entering stop3 and continuing ADC conversions.

### 0.3.8 MCU Stop2 Mode Operation

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

## 0.4 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to [Table 0-6](#), [Table 0-7](#), and [Table 0-8](#) 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.

### 0.4.1 ADC Module Initialization Example

#### 0.4.1.1 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.

#### 0.4.1.2 Pseudo-Code Example

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

##### **ADCCFG = 0x98 (%10011000)**

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

##### **ADCSC2 = 0x00 (%00000000)**

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

##### **ADCSC1 = 0x41 (%01000001)**

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

##### **ADCRH/L = 0xxx**

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

##### **ADCCVH/L = 0xxx**

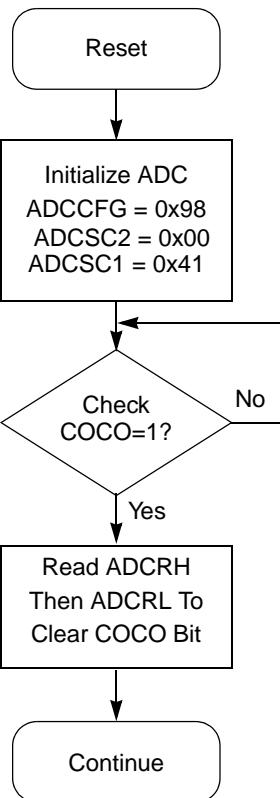
Holds compare value when compare function enabled

##### **APCTL1=0x02**

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

##### **APCTL2=0x00**

All other AD pins remain general purpose I/O pins

**Figure 0-12. Initialization Flowchart for Example**

## 0.5 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.

### 0.5.1 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.

#### 0.5.1.1 Analog Supply Pins

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

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

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

### 0.5.1.2 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  $V_{DDA}$  on some devices. The low reference is  $V_{REFL}$ , which may be shared on the same pin as  $V_{SSA}$  on some devices.

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

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

### 0.5.1.3 Analog Input Pins

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

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

For proper conversion, the input voltage must fall between  $V_{REFH}$  and  $V_{REFL}$ . If the input is equal to or exceeds  $V_{REFH}$ , the converter circuit converts the signal to 0xFFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than  $V_{REFL}$ , the converter circuit converts it to 0x000. Input voltages between  $V_{REFH}$  and  $V_{REFL}$  are straight-line linear conversions. There is a brief current associated with  $V_{REFL}$  when the sampling

capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

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

## 0.5.2 Sources of Error

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

### 0.5.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately  $7\text{k}\Omega$  and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source ( $R_{AS}$ ) is kept below 2 k $\Omega$ .

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

### 0.5.2.2 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  $V_{DDA} / (2^N * I_{LEAK})$  for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

### 0.5.2.3 Noise-Induced Errors

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

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

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V<sub>DD</sub> 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  $\mu$ F capacitor (C<sub>AS</sub>) on the selected input channel to V<sub>REFL</sub> or V<sub>SSA</sub> (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

#### 0.5.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1 \text{ lsb} = (V_{\text{REFH}} - V_{\text{REFL}}) / 2^N \quad \text{Eqn. 0-1}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be  $\pm 1/2$  lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is -1 lsb to 0 lsb and the code width of each step is 1 lsb.

#### 0.5.2.5 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<sub>ZS</sub>) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E<sub>FS</sub>) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.

- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

### 0.5.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

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

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around  $\pm 1/2$  lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 0.5.2.3](#) reduces this error.

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

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





# Chapter 10

## Cyclic Redundancy Check Generator (S08CRCV3)

### 10.1 Introduction

This section describes the Cyclic Redundancy Check (CRC) generator module which uses the 16-bit CRC-CCITT polynomial,  $x^{16} + x^{12} + x^5 + 1$ , to generate a CRC code for error detection. The 16-bit code is calculated for 8-bits of data at a time, and provides a simple check for all accessible memory locations, whether they be in FLASH or RAM.

#### 10.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using 16-bit shift register
- CRC16-CCITT compliancy with  $x^{16} + x^{12} + x^5 + 1$  polynomial
- Error detection for all single, double, odd, and most multi-bit errors
- Programmable initial seed value
- High-speed CRC calculation
- 8-bit transpose register to flip bit order



## **10.1.2 Features**

Features of the CRC module include:

- Hardware CRC generator circuit using 16-bit shift register
- CRC16-CCITT compliancy with  $x^{16} + x^{12} + x^5 + 1$  polynomial
- Error detection for all single, double, odd, and most multi-bit errors
- Programmable initial seed value
- High-speed CRC calculation
- Optional feature to transpose input data and CRC result via transpose register, required on applications where bytes are in LSb (Least Significant bit) format.

## **10.1.3 Modes of Operation**

This section defines the CRC operation in run, wait, and stop modes.

- Run Mode - This is the basic mode of operation.
- Wait Mode - The CRC module is operational.
- Stop 1 and 2 Modes- The CRC module is not functional in these modes and will be put into its reset state upon recovery from stop.
- Stop 3 Mode - In this mode, the CRC module will go into a low power standby state. Any CRC calculations in progress will stop and resume after the CPU goes into run mode.

## 10.1.4 Block Diagram

Figure 10-1 provides a block diagram of the CRC module.

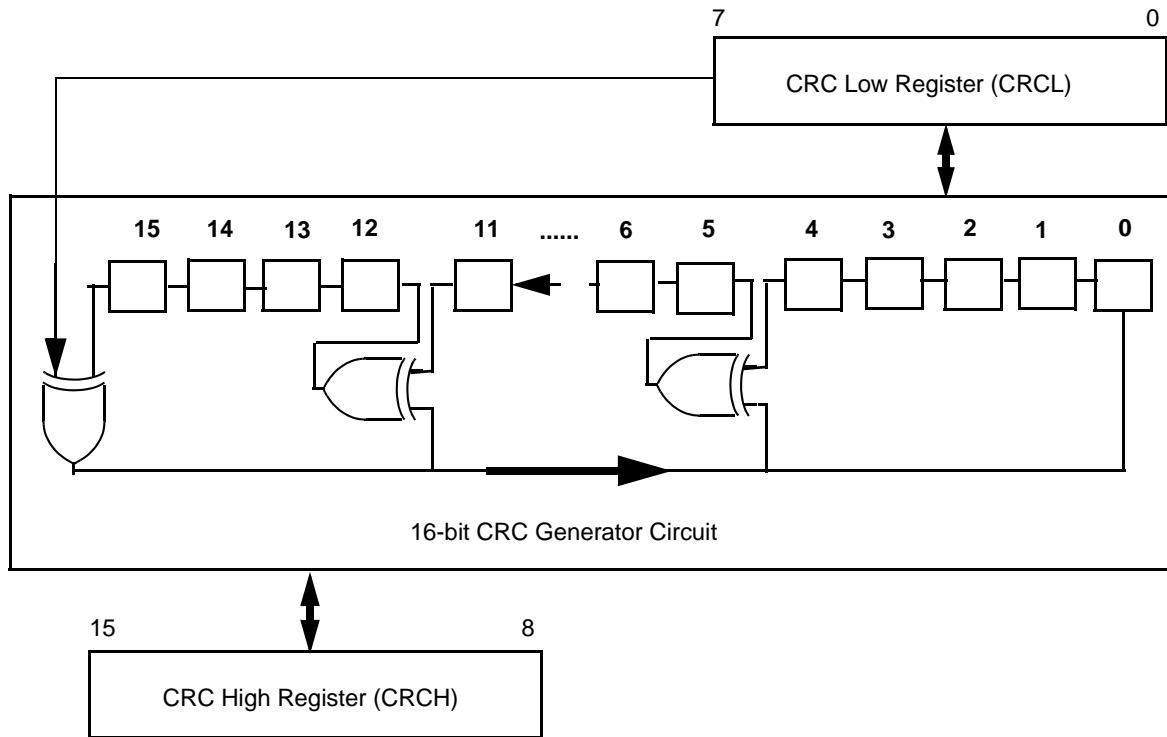


Figure 10-1. Cyclic Redundancy Check (CRC) Module Block Diagram

## 10.2 External Signal Description

There are no CRC signals that connect off chip.

## 10.3 Register Definition

### 10.3.1 Memory Map

Table 10-1. CRC Register Summary

Name		7	6	5	4	3	2	1	0
CRCH (offset=0)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
CRCL (offset=1)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
TRANSPOSE (offset=2)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								

## NOTE

The transpose feature (offset=2) is optional. If this feature is required by the MCU, parameter TRANSPOSE\_FEATURE should be 1'b1. Otherwise, it should be 1'b0 (default).

## NOTE

Offsets 4,5,6 and 7 are also mapped onto the CRCL register. This is an *alias* only used on CF1Core (version 1 of ColdFire core) and should be ignored for HCS08 cores. See [Section 10.4.2, “Programming model extension for CF1Core”](#) for more details.

### 10.3.2 Register Descriptions

The CRC module includes:

- A 16-bit CRC result and seed register (CRCH:CRCL)
- An 8-bit transpose register to convert from LSb to MSb format (or vice-versa) when required by the application

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all CRC registers. This section refers to registers only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

#### 10.3.2.1 CRC High Register (CRCH)

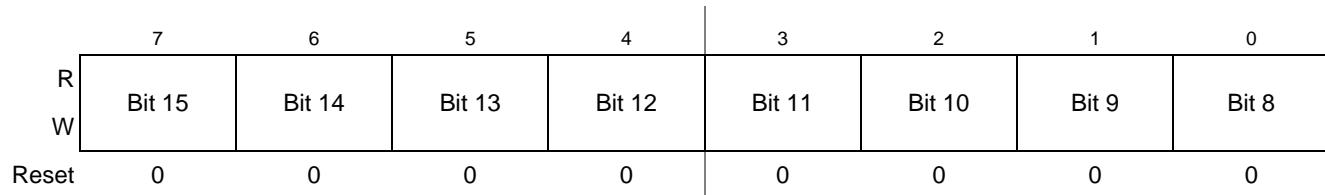


Figure 10-2. CRC High Register (CRCH)

Table 10-2. Register Field Descriptions

Field	Description
7:0 CRCH	<b>CRCH</b> -- This is the high byte of the 16-bit CRC register. A write to CRCH will load the high byte of the initial 16-bit seed value directly into bits 15-8 of the shift register in the CRC generator. The CRC generator will then expect the low byte of the seed value to be written to CRCL and loaded directly into bits 7-0 of the shift register. Once both seed bytes written to CRCH:CRCL have been loaded into the CRC generator, and a byte of data has been written to CRCL, the shift register will begin shifting. A read of CRCH will read bits 15-8 of the current CRC calculation result directly out of the shift register in the CRC generator.

### 10.3.2.2 CRC Low Register (CRCL)

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

Figure 10-3. CRC High Register (CRCH)

Table 10-3. Register Field Descriptions

Field	Description
7:0 CRCL	<b>CRCL</b> -- This is the low byte of the 16-bit CRC register. Normally, a write to CRCL will cause the CRC generator to begin clocking through the 16-bit CRC generator. As a special case, if a write to CRCH has occurred previously, a subsequent write to CRCL will load the value in the register as the low byte of a 16-bit seed value directly into bits 7-0 of the shift register in the CRC generator. A read of CRCL will read bits 7-0 of the current CRC calculation result directly out of the shift register in the CRC generator.

### 10.3.2.3 Transpose Register (TRANSPOSE)

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

Figure 10-4. CRC High Register (CRCH)

Table 10-4. Register Field Descriptions

Field	Description
7:0 TRANSPOSE OSE	<b>TRANSPOSE</b> -- This register is used to transpose data, converting from LSb to MSb (or vice-versa). The byte to be transposed should be first written to TRANSPOSE and then subsequent reads from TRANSPOSE will return the transposed value of the last written byte (bit 7 becomes bit 0, bit 6 becomes bit 1, and so forth).

## 10.4 Functional Description

To enable the CRC function, a write to the CRCH register will trigger the first half of the seed mechanism which will place the CRCH value directly into bits 15-8 of the CRC generator shift register. The CRC generator will then expect a write to CRCL to complete the seed mechanism.

As soon as the CRCL register is written to, its value will be loaded directly into bits 7-0 of the shift register, and the second half of the seed mechanism will be complete. This value in CRCH:CRCL will be the initial seed value in the CRC generator.

Now the first byte of the data on which the CRC calculation will be applied should be written to CRCL. This write after the completion of the seed mechanism will trigger the CRC module to begin the CRC checking process. The CRC generator will shift the bits in the CRCL register (MSB first) into the shift register of the generator. After all 8 bits have been shifted into the CRC generator (**in the next bus cycle**

after the data write to CRCL), the result of the shifting, or the value currently in the shift register, can be read directly from CRCH:CRCL, and the next data byte to include in the CRC calculation can be written to the CRCL register.

This next byte will then also be shifted through the CRC generator's 16-bit shift register, and after the shifting has been completed, the result of this second calculation can be read directly from CRCH:CRCL.

After each byte has finished shifting, a new CRC result will appear in CRCH:CRCL, and an additional byte may be written to the CRCL register to be included within the CRC16-CCITT calculation. A new CRC result will appear in CRCH:CRCL each time 8-bits have been shifted into the shift register.

To start a new CRC calculation, write to CRCH, and the seed mechanism for a new CRC calculation will begin again.

#### 10.4.1 ITU-T (CCITT) Recommendations and Expected CRC Results

The CRC polynomial  $0x1021 (x^{16} + x^{12} + x^5 + 1)$  is popularly known as *CRC-CCITT* since it was initially proposed by the ITU-T (formerly CCITT) committee.

Although the ITU-T recommendations are very clear about the polynomial to be used, 0x1021, they accept variations in the way the polynomial is implemented:

- ITU-T V.41 implements the same circuit shown in [Figure 10-1](#), but it recommends a SEED = 0x0000.
- ITU-T T.30 and ITU-T X.25 implement the same circuit shown in [Figure 10-1](#), but they recommend the final CRC result to be negated (one-complement operation). Also, they recommend a SEED = 0xFFFF.

Moreover, it is common to find circuits in literature slightly different from the one suggested by the recommendations above, but also known as CRC-CCITT circuits (many variations require the message to be augmented with zeros).

The circuit implemented in the CRC module is exactly the one suggested by the ITU-T V.41 recommendation, with an added flexibility of a programmable SEED. As in ITU-T V.41, no augmentation is needed and the CRC result is not negated. [Table 10-5](#) shows some expected results that can be used as a reference. Notice that these are ASCII string messages. For example, message “123456789” is encoded with bytes 0x31 to 0x39 (see ASCII table).

**Table 10-5. Expected CRC results**

ASCII String Message	SEED (initial CRC value)	CRC result
“A”	0x0000	<b>0x58e5</b>
“A”	0xffff	<b>0xb915</b>
“A”	0x1d0f <sup>1</sup>	<b>0x9479</b>
“123456789”	0x0000	<b>0x31c3</b>
“123456789”	0xffff	<b>0x29b1</b>

“123456789”	0x1d0f <sup>1</sup>	<b>0xe5cc</b>
A string of 256 upper case “A” characters with no line breaks	0x0000	<b>0xabe3</b>
A string of 256 upper case “A” characters with no line breaks	0xfffff	<b>0xea0b</b>
A string of 256 upper case “A” characters with no line breaks	0x1d0f <sup>1</sup>	<b>0xe938</b>

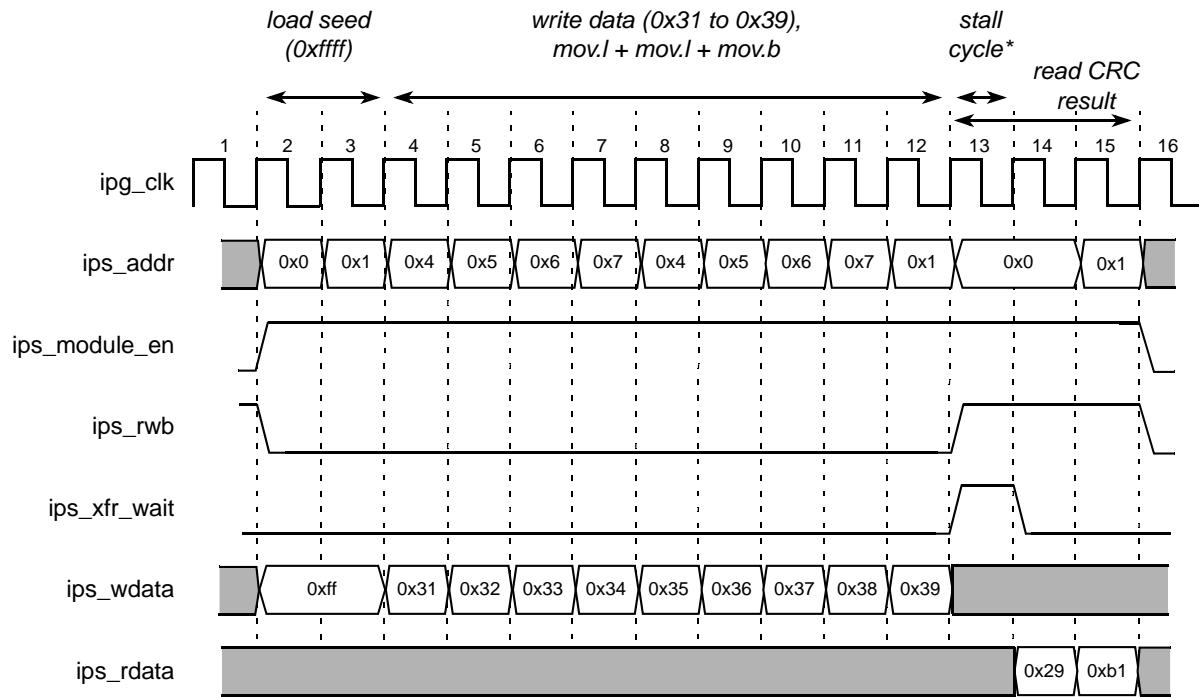
<sup>1</sup> One common variation of CRC-CCITT require the message to be augmented with zeros and a SEED=0xFFFF. The CRC module will give the same results of this alternative implementation when SEED=0x1D0F and no message augmentation.

#### 10.4.2 Programming model extension for CF1Core

The CRC module extends its original programming model to allow faster CRC calculations on CF1 cores. Memory offsets 0x4, 0x5, 0x6 and 0x7 are mapped (aliased) onto the CRCL register, in a way that the CF1Core can execute 32-bit store instructions to write data to the CRC module. The code should use a single *mov.l* store instruction to send four bytes beginning at address 0x4, which will be decomposed by the platform into four sequential/consecutive byte writes to offsets 0x4-0x7 (all aliased to the CRCL position).

In addition, reads from 0x4-0x7 return the contents of the CRCL register. Reads from 0x2-0x3 (unused/reserved) return 0x00. Writes to 0x2-0x3 have no effect.

Due to internal CF1Core characteristics, this approach provides a greater data transfer rate than the original programming model used on HCS08 (single byte writes to CRCL position). [Figure 10-5](#) illustrates a message calculation on CF1Core.



\* On cycle 13, there is a read-after-write hazard, since calculation of 0x39 data is underway. *ips\_xfr\_wait* is asserted to signalize a stall cycle (the IPS master should wait until cycle 14 to read the CRC result).

**Figure 10-5. CRC calculation of ASCII message “123456789” (0x31 to 0x39) on CF1Core**

### 10.4.3 Transpose feature

The CRC module provides an optional feature to transpose data (invert bit order). This feature is specially useful on applications where the LSb format is used, since the CRC CCITT expects data in the MSb format. In that case, before writing new data bytes to CRCL, these bytes should be transposed as follows:

1. Write data byte to TRANSPOSE register
2. Read data from TRANSPOSE register (subsequent reads will result in the transposed value of the last written data)
3. Write transposed byte to CRCL.

After all data is fed into CRC, the CRCH:CRCL result is available in the MSb format. Then, these two bytes should also be transposed: the values read from CRCH:CRCL should be written/read to/from the TRANSPOSE register.

Although the transpose feature was initially designed to address LSb applications interfacing with the CRC module, it is important to notice that this feature is not necessarily tied to CRC applications. Since it was designed as an independent register, any application should be able to transpose data by writing/reading to/from the TRANSPOSE register (e.g.: Big endian / Little endian conversion in USB).

## 10.5 Initialization Information

To initialize the CRC Module and initiate a CRC16-CCITT calculation, follow this procedure:

1. Write high byte of initial seed value to CRCH.
2. Write low byte of initial seed value to CRCL.
3. Write first byte of data on which CRC is to be calculated to CRCL (an alternative option is offered for CF1Cores. See [Section 10.4.2, “Programming model extension for CF1Core”](#)).
4. In the next bus cycle after step 3, if desired, the CRC result from the first byte can be read from CRCH:CRCL.
5. Repeat steps 3-4 until the end of all data to be checked.

# Chapter 11

## Digital to Analog Converter (S08DACV1)

### 11.1 Introduction

The 5-bit digital to analog converter (DAC) is a 32-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed.

Some MCUs may have more than one DAC, so references to modules and register names include a placeholder character to identify which DAC module (use x as placeholder) is being referenced. In cases where only one DAC is available on a particular device, the x placeholder is omitted.

The MC9S08MP16 Series has three DAC modules, referred to as DAC1, DAC2, and DAC3. [Figure 11-1](#) shows the MC9S08MP16 Series block diagram with the DAC module highlighted.

### 11.2 Module Configurations

This section provides device-specific information for configuring the DAC on MC9S08MP16 Series.

#### 11.2.1 DAC Clock Gating

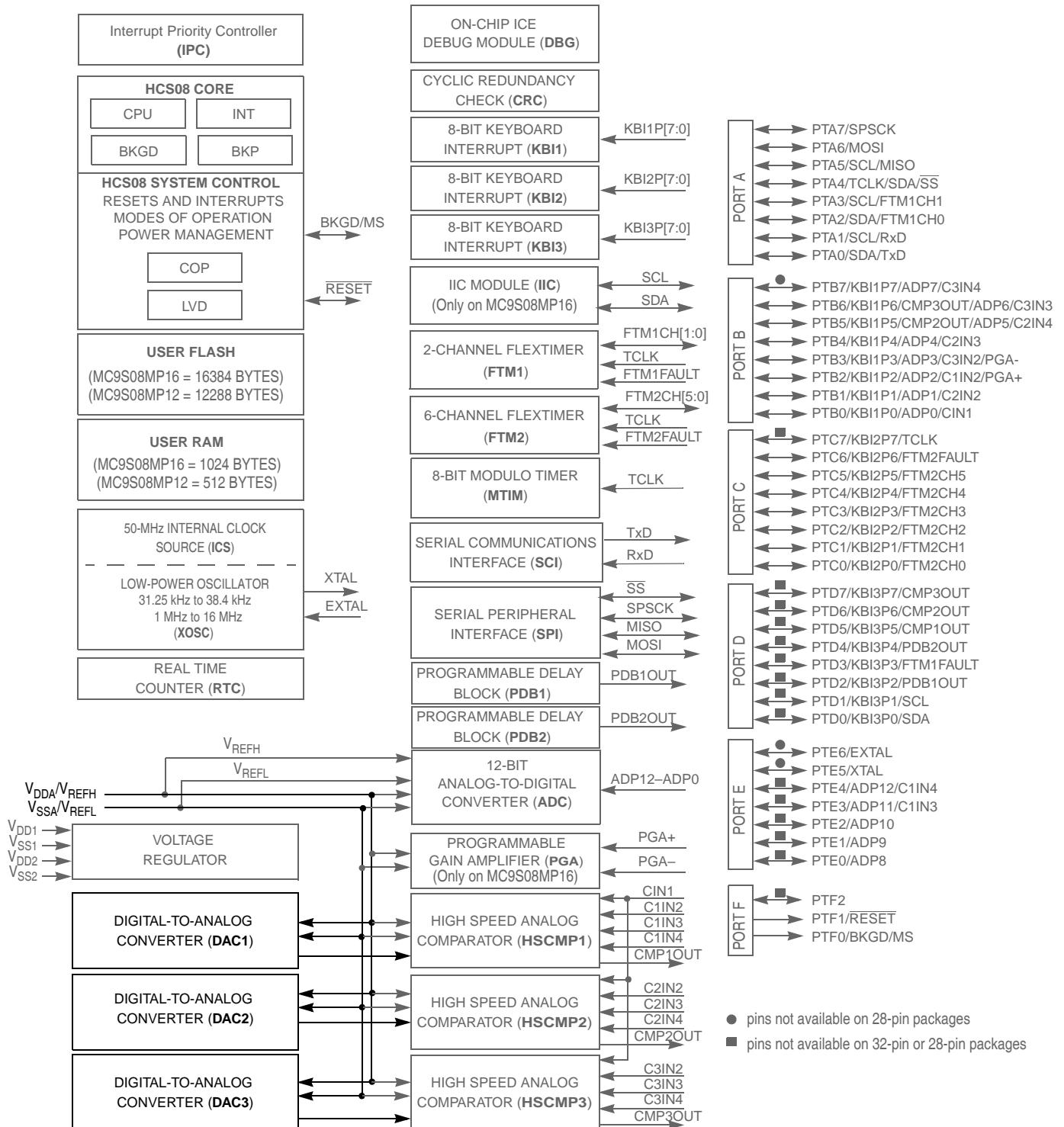
The bus clock to each DAC module can be gated on and off using the associated control bit in SCGC1. The CMPDAC1 bit is used to control the clock to both the HSCMP1 and DAC1 modules. The CMPDAC2 bit is used to control the clock to both the HSCMP2 and DAC2 modules. The CMPDAC3 bit is used to control the clock to both the HSCMP3 and DAC3 modules. These bits are set after any reset, which enables the bus clock to these modules. To conserve power, the CMPDACx bit can be cleared to disable the clock to the associated modules when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 11.2.2 DAC/HSCMP Configuration

Each DAC module is associated with a HSCMP block. DAC1 analog output is connected to the M4 analog input of HSCMP1. DAC2 analog output is connected to the M4 analog input of HSCMP2. DAC3 analog output is connected to the M4 analog input of HSCMP3.

#### 11.2.3 Reference Options

On MC9S08MP16 Series devices, the Vin input supply reference used is  $V_{DDA}$  regardless of VRSEL bit setting.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

Figure 11-1. MC9S08MP16 Series Block Diagram Highlighting DAC Block and Pins

## 11.2.4 Features

- 2.7V–5.5V operation range
- 5-bit resolution
- Selectable supply reference source
- Power down mode to conserve power when it is not being used
- Output can be routed to internal comparator input
- Less than 20 $\mu$ A power consumption
- Can remain operational in STOP3 mode

## 11.2.5 Modes of Operation

Modes of operation: Digital-to-Analog Converter will operate in any mode as long as power down mode is not selected

## 11.2.6 Block Diagram

The block diagram of the DAC module is shown in Figure 1-1. It contains a 32-tap resistor ladder network and a 32-to-1 multiplexer, which selects an output voltage from one of 32 distinct levels that outputs from DACO. It is controlled through DAC Control register (DACCTRL). Its supply reference source can be selected from two sources  $V_{in1}$  and  $V_{in2}$ . The module can be powered down (disabled) when it is not used. When in disable mode, DACO is connected to Vssa

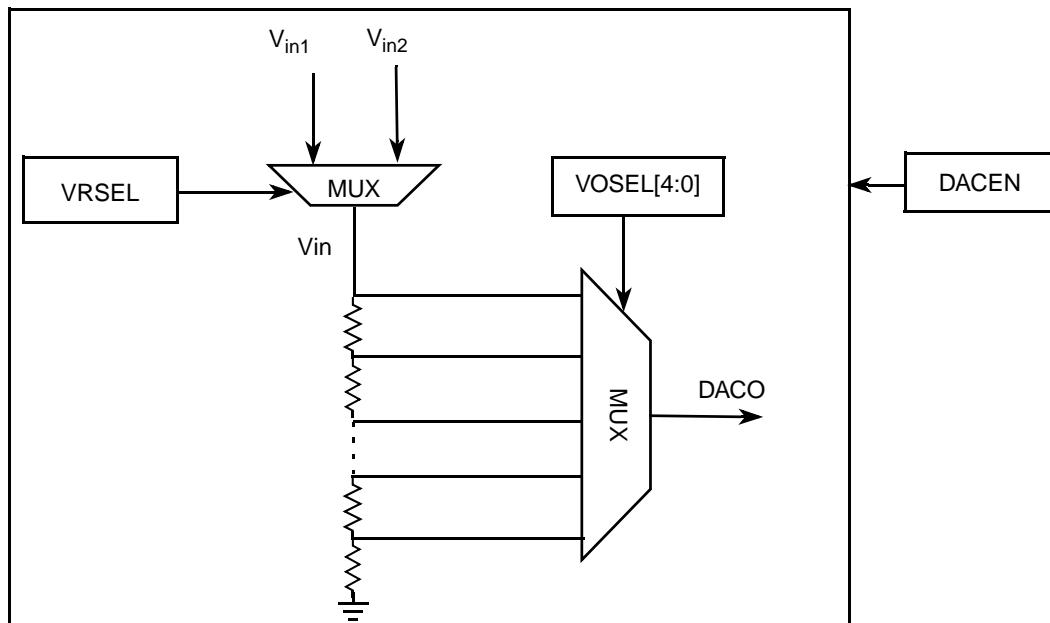


Figure 11-2. 5-bit DAC Block Diagram

## 11.3 Memory Map/Register Definition

Table 11-1. DAC Memory Map

Offset	Register	Description
0x00	DACCTRL	DAC Control Register

### 11.3.1 DAC Control Register (DACCTRL)

This register contains control bits for the DAC. The module is enabled (powered up) if DACEN is set at any operation mode.

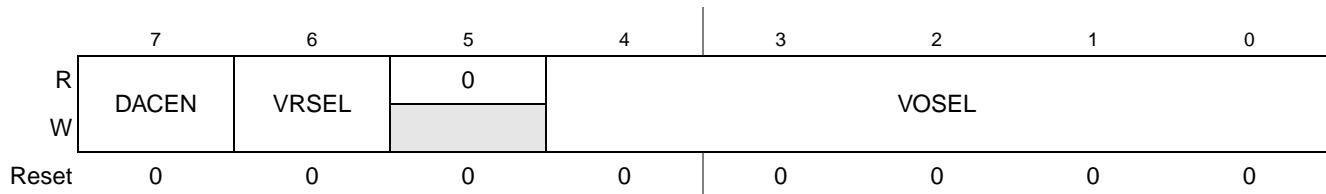


Figure 11-3. DAC Control Register (DACCTRL)

Table 11-2. DACCTRL Field Descriptions

Field	Description
7 DACEN	DAC enable. When the module is disabled, the DAC is powered down to conserve power. 0 DAC is disabled (powered down) and output is Vssa 1 DAC is operational
6 VRSEL	Supply voltage reference source select. This bit selects the supply reference source 0 $V_{in1}$ is selected as resistor ladder network supply reference Vin 1 $V_{in2}$ is selected as resistor ladder network supply reference Vin
5	Reserved, must be cleared.
4:0 VOSEL	DAC output voltage select. Selects an output voltage from one of 32 distinct levels. $DACO = (Vin/32) \times (VSEL[4:0] + 1)$ , DACO ranges from Vin/32 to Vin

### 11.3.2 Functional Description

#### 11.3.2.1 Operation

A 32-to-1 multiplexer selects an output voltage from one of 32 distinct levels which is generated from a 32-tap resistor ladder network. DAC Control register (DACCTRL) controls multiplexer selection, selection of resistor ladder network supply reference source. When the module is disabled, its output, DACO, should be equal to Vssa

#### 11.3.2.2 Voltage reference source select

- $V_{in1}$  should be used to connect to the primary voltage source as supply reference of the 32-tap resistor ladder

- $V_{in2}$  should be used to connect to alternate voltage source (or primary source if alternate voltage source is not available)

## 11.4 Resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

## 11.5 Clocks

This module has a single clock input, the Bus peripheral clock.

## 11.6 Interrupts

This module has no interrupts.



# Chapter 12

## Flextimer Module (S08FTMV2)

### 12.1 Introduction

The FTM uses one input/output (I/O) pin per channel, FTMxCH<sub>n</sub> where x is the FTM number (for example, 1 or 2) and n is the channel number (for example, 0–1). The FTM shares its I/O pins with general-purpose I/O port pins (refer to the [Pins and Connections](#) chapter for more information).

All MC9S08MP16 Series MCUs have two FTM modules. FTM1 is a 2-channel timer and FTM2 is a 6-channel timer. [Figure 12-1](#) shows the MC9S08MP16 Series block diagram with the FTM modules highlighted.

### 12.2 Module Configurations

This section provides device-specific information for configuring the FTM modules on MC9S08MP16 Series.

#### 12.2.1 FTM1 and FTM2 Clocking

FTM1 and FTM2 are designed to support clock rates up to 50MHz. Use the following FTM clock source selection table when programming FTM1 and FTM2. Setting FTMxSC[CLKS] to 01 selects the output of the ICS (ICSOUT) which is at a rate of twice the bus frequency. See [Chapter 14, “Internal Clock Source \(S08ICSV3\)”](#) for information on ICSOUT.

Table 12-1. FTM 1 and FTM 2 Clock Source Selection

CLKS	FTM Clock Source to Prescaler Input
0:0	No clock selected (FTMx disabled)
0:1	2x bus rate clock (ICSOUT)
1:0	Fixed system clock (FFCLK) <sup>1</sup>
1:1	External source (TCLK) <sup>2</sup>

<sup>1</sup> When FFCLK is selected but the ICS is not programmed correctly to supply a valid FFCLK, the FTM clocks at the ICSOUT frequency.

<sup>2</sup> The maximum frequency that is allowed as an external clock is one-fourth of the bus frequency.

#### 12.2.2 FTM External Clock

The FTM modules on the MC9S08MP16 Series use the TCLK pin. The external clock for the FTM module is selected by setting CLKS = 1:1 in FTMxSC, which selects the TCLK pin input. The TCLK input can be enabled as external clock inputs to the MTIM and both FTM modules simultaneously.

### 12.2.3 FTM External Clock Pin Repositioning

The TCLK pin can be repositioned under software control using TCLKPS in SOPT2. When TCLKPS is clear, the reset default condition, PTC7 is the TCLK input pin. When TCLKPS is set, PTA4 is the TCLK input pin.

### 12.2.4 HSCMPx/FTM1 Input Capture

The HSCMP1 module can be configured to connect the COUT output of the analog comparator to a FTM1 input capture channel 0 by setting the ACIC1 bit in SOPT1. With ACIC1 set, the FTM1CH0 pin is not available externally regardless of the configuration of the FTM1 module.

The HSCMP2 module can be configured to connect the COUT output of the analog comparator to a FTM1 input capture channel 1 by setting the ACIC2 bit in SOPT1. With ACIC2 set, the FTM1CH1 pin is not available externally regardless of the configuration of the FTM1 module.

### 12.2.5 FTM Fault

Each FTM module supports up to four FTM fault sources. On MC9S08MP16 Series, the same source signal is used to signal faults to both the FTM1 and FTM2 in several cases. [Table 12-2](#) summarizes the FTM fault sources for MC9S08MP16 Series devices.

**Table 12-2. MC9S08MP16 Series FTM Fault Event Sources**

Fault Source	FTM Fault Input	FTM Filter Available
FTM1FAULT Pin	FTM1 Fault 0	Yes
FTM2FAULT Pin	FTM2 Fault 0	Yes
HSCMP1 COUT	FTM1 Fault 1 FTM2 Fault 1	No No
HSCMP2 COUT	FTM1 Fault 2 FTM2 Fault 2	No No
HSCMP3 COUT	FTM1 Fault 3 FTM2 Fault 3	No No

The fault filtering functions for Fault 1, Fault 2, and Fault 3 are not used on MC9S08MP16 Series devices and writes to FFLTR1EN, FFLTR2EN, and FFLTR3EN bits in FTMxFLTCTRL have no effect.

### 12.2.6 FTM Input Synchronization

Each FTM module supports up to three PWM synchronization sources. On MC9S08MP16 Series, the same source signal is used trigger both the FTM1 and FTM2 in several cases and provides the ability to keep both FTM modules synchronized..

The FTM2T2S bit in SOPT2 register is used to select the trigger source for the FTM2 Trigger 2 input. The HSCMP3 COUT output of the analog comparator is configured to the FTM2 Trigger 2 input by clearing the FTM2T2S. The FTM1 Channel 0 output compare is configured to the FTM2 Trigger 2 input by setting

the FTM2T2S. With FTM2T2S set, the FTM1CH0 pin is not available externally regardless of the configuration of the FTM1 module.

**Table 12-3** summarizes the FTM synchronization sources for MC9S08MP16 Series devices.

**Table 12-3. MC9S08MP16 Series FTM Synchronization Trigger Sources**

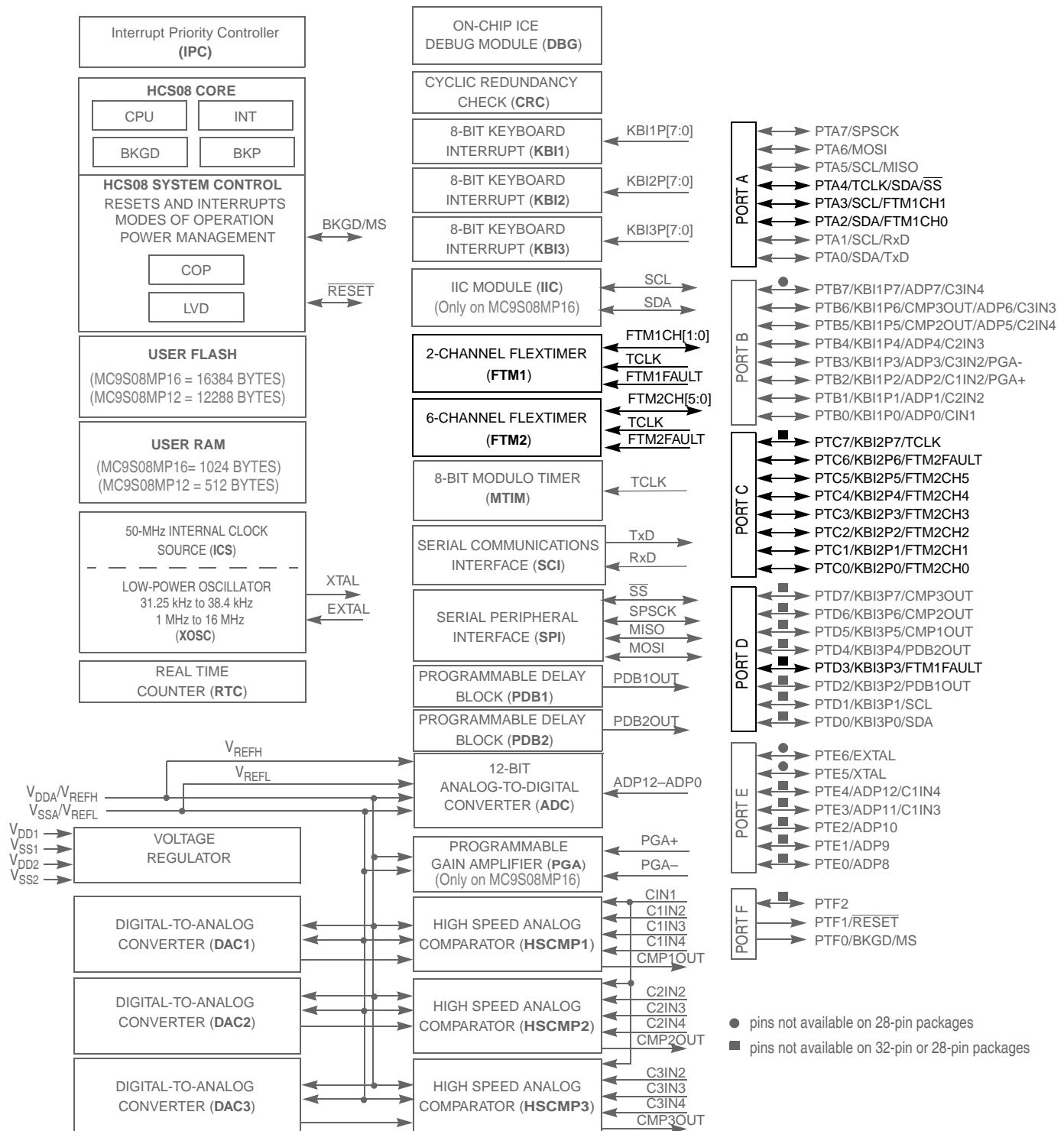
Trigger Source	FTM Synchronization Input
SYNCFTM bit in SOPT3	FTM1 Trigger 0 FTM2 Trigger 0
HSCMP1 COUT	FTM1 Trigger 1 FTM2 Trigger 1
HSCMP2 COUT	FTM1 Trigger 2
HSCMP3 COUT (FTM2T2S=0)	FTM2 Trigger 2
FTM1CH0 output compare(FTM2T2S=1)	

## 12.2.7 FTM/ADC Triggering

Each FTM module supports an ADC hardware trigger function associated with the CHnTRIG bits in FTMXEXTTRIG. On MC9S08MP16 Series devices this trigger event is not used and writes to the associated FTM bits have no effect on device operation.

## 12.2.8 FTM/PDB Triggering

Each FTM module supports an initialization trigger function associated with the INITTRIGEN bit in FTMXEXTTRIG. On MC9S08MP16 Series devices this event is used to trigger both PDB1 and PDB2 module input triggering functions. FTM1 is connected to the TriggerIn4 input of both PDB1 and PDB2. FTM2 is connected to the TriggerIn5 input of both PDB1 and PDB2.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

**Figure 12-1. MC9S08MP16 Series Block Diagram Highlighting FTM Block and Pins**

## 12.2.9 Features

The FTM features include:

- FTM source clock is selectable
  - Source clock can be the system clock, the fixed system clock or a clock from an external pin
  - Fixed system clock source is synchronized to the system clock by an external synchronization circuit to FTM
  - External clock pin may be shared with any FTM channel pin or a separated input pin
  - External clock source is synchronized to the system clock by FTM
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64 or 128
- FTM has a 16-bit counter
  - It can be a free-running counter or a counter with initial and final value
  - The counting can be up or up-down
- Each channel can be configured for input capture, output compare or edge-aligned PWM mode
- In input capture mode
  - the capture can occur on rising edges, falling edges or both edges
  - an input filter can be selected for some channels
- In output compare mode the output signal can be set, cleared or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal (with independent control of both edges of PWM signal)
- The FTM channels can operate as pairs with equal outputs, pairs with complimentary outputs or independent channels (with independent outputs)
- The deadtime insertion is available for each complementary pair
- Generation of triggers (hardware trigger)
- Software control of PWM outputs
- Up to 4 fault inputs for global fault control
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- The generation of an interrupt when the fault condition is detected
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions

## 12.2.10 Modes of Operation

During stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During wait mode, the FTM continues to operate normally. Provided the FTM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, then power can be saved by disabling FTM functions before entering wait mode.

### 12.2.10.1 BDM Mode

When BDM mode is active, the FlexTimer counter and the channels output are frozen. However, the value of FlexTimer counter or the channels output are modified in BDM mode in the following cases.

- Write any value to FTmxCNTH or FTmxCNTL registers ([Section 12.5.3.4, “Counter reset”](#)) resets the FTM counter to the value of FTmxCNTINH:L registers and the channels output to their initial value (except for channels in output compare mode).
- The PWM synchronization with REINIT = 1 ([Section 12.5.11, “PWM synchronization”](#)) resets the FTM counter to the value of FTmxCNTINH:L registers and the channels output to their initial value (except for channels in output compare mode).
- The initialization ([Section 12.5.16, “Initialization”](#)) forces the value of the CHnOI bit to the channel (n) output.

#### NOTE

It is not recommended to use the above cases together the fault control ([Section 12.5.14, “Fault control mode”](#)). If the fault control is enabled and there is the fault condition at the enabled fault input, these cases reset the FTM counter to the value FTmxCNTINH:L and the channels output to their initial value.

## 12.2.11 Block Diagram

The FTM uses one input/output (I/O) pin per channel, FTmxCHn (FTM channel (n)) where n is the channel number (0-7). The FTM 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 12-2](#) shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

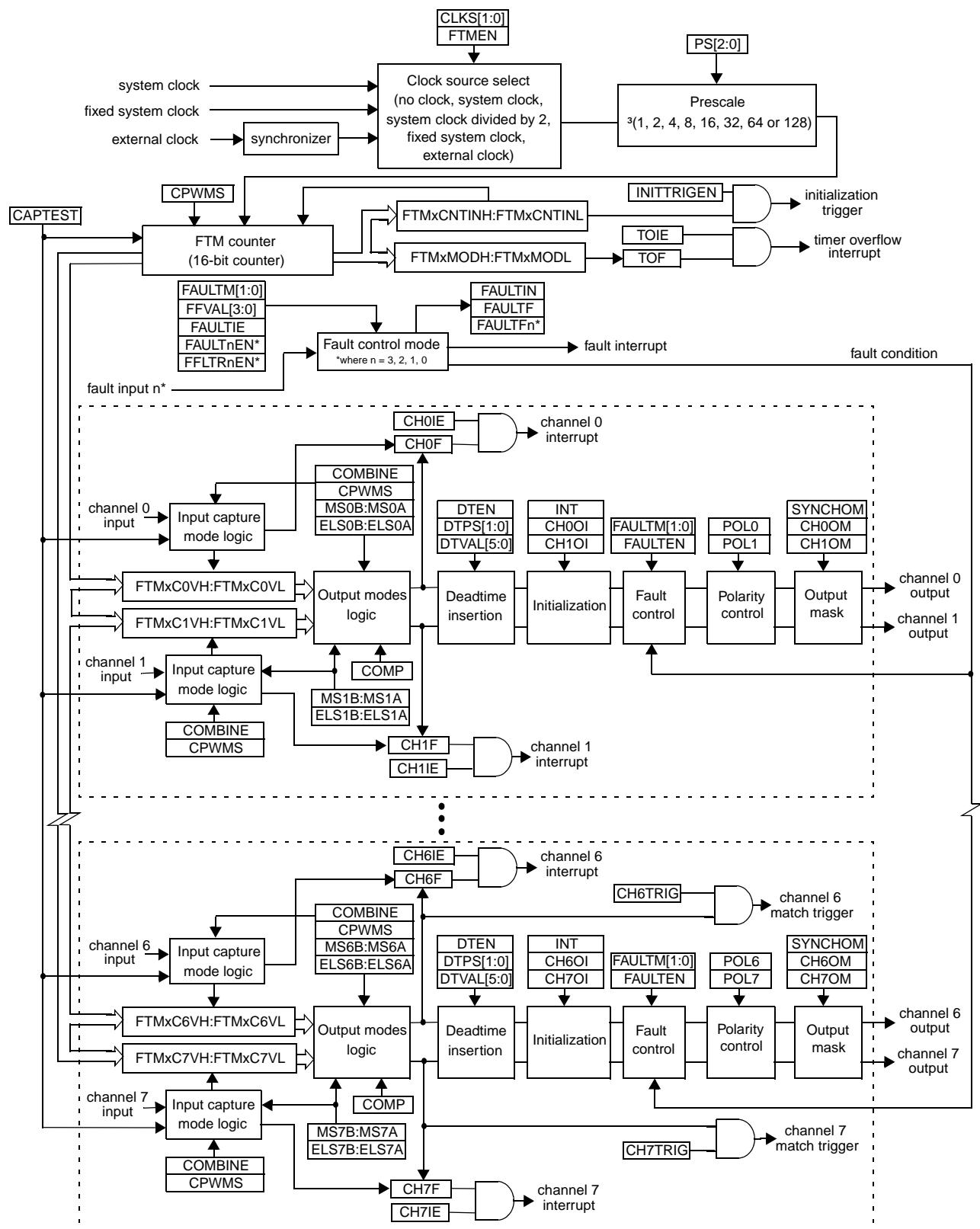


Figure 12-2. FTM Block Diagram

## 12.3 Signal Description

Table 12-4 shows the user-accessible signals for the FTM.

**Table 12-4. Signal Properties**

Name	Function
FTMxCLK <sup>1</sup>	FTM x External Clock - FTM external clock which may be selected to drive the FTM counter
FTMxCHn <sup>2</sup>	FTM x channel (n) - I/O pin associated with FTM channel (n)

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

<sup>2</sup> n=channel number (0 to 7)

### 12.3.1 FTMxCLK — FTM External Clock

When an external clock is included, it can be shared with the same pin as any FTM channel or it can be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

If the external clock signal shares the same pin as a channel I/O pin, then this channel pin will not be available for channel I/O functions when the external clock source is selected. It is the user's responsibility to avoid conflicting settings such as this.

The external clock source is synchronized in the FTM. The system clock clocks the synchronizer, so the frequency of the external source must be no more than one-fourth the frequency of the system clock, to meet Nyquist criteria and allow for jitter.

### 12.3.2 FTMxCHn — FTM Channel (n) I/O Pin(s)

Each FTM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The FTM pins are shared 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.

Immediately after reset, no FTM functions are enabled, so all FTM channels pins revert to general purpose I/O control. The FTM channels pins also revert to general purpose I/O control when (CLKS = 00) or (ELSnB:ELSnA = 0:0).

## 12.4 Memory Map and Register Definition

This section provides a detailed description of all FTM registers accessible to the end user.

### 12.4.1 Module Memory Map

This section presents a high-level view of the FTM registers and how they are mapped. [Table 12-5](#) shows the registers for an 8-channel FTM. Register names include a timer number (x) because it is common for MCU systems to include more than one FTM.

**Table 12-5. 8-channel FTM Module Memory Map**

Address	Use	Access	Section/Page
Base1+0x0000	FTM Status and Control (FTMxCSC)	R/W	<a href="#">12.4.3/-185</a>
Base1+0x0001	FTM Counter Register High (FTMxCNTH)	R <sup>1</sup>	<a href="#">12.4.4/-186</a>
Base1+0x0002	FTM Counter Register Low (FTMxCNTL)	R <sup>1</sup>	<a href="#">12.4.4/-186</a>
Base1+0x0003	FTM Modulo Register High (FTMxMODH)	R/W	<a href="#">12.4.5/-187</a>
Base1+0x0004	FTM Modulo Register Low (FTMxMODL)	R/W	<a href="#">12.4.5/-187</a>
Base1+0x0005	FTM Channel 0 Status and Control (FTMxC0SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x0006	FTM Channel 0 Value High (FTMxC0VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0007	FTM Channel 0 Value Low (FTMxC0VL)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0008	FTM Channel 1 Status and Control (FTMxC1SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x0009	FTM Channel 1 Value High (FTMxC1VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x000A	FTM Channel 1 Value Low (FTMxC1VL)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x000B	FTM Channel 2 Status and Control (FTMxC2SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x000C	FTM Channel 2 Value High (FTMxC2VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x000D	FTM Channel 2 Value Low (FTMxC2VL)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x000E	FTM Channel 3 Status and Control (FTMxC3SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x000F	FTM Channel 3 Value High (FTMxC3VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0010	FTM Channel 3 Value Low (FTMxC3VL)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0011	FTM Channel 4 Status and Control (FTMxC4SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x0012	FTM Channel 4 Value High (FTMxC4VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0013	FTM Channel 4 Value Low (FTMxC4VL)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0014	FTM Channel 5 Status and Control (FTMxC5SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x0015	FTM Channel 5 Value High (FTMxC5VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0016	FTM Channel 5 Value Low (FTMxC5VL)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0017	FTM Channel 6 Status and Control (FTMxC6SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x0018	FTM Channel 6 Value High (FTMxC6VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x0019	FTM Channel 6 Value Low (FTMxC6VL)	R/W	<a href="#">12.4.7/-190</a>

**Table 12-5. 8-channel FTM Module Memory Map**

Base1+0x001A	FTM Channel 7 Status and Control (FTMxC7SC)	R/W	<a href="#">12.4.6/-188</a>
Base1+0x001B	FTM Channel 7 Value High (FTMxC7VH)	R/W	<a href="#">12.4.7/-190</a>
Base1+0x001C	FTM Channel 7 Value Low (FTMxC7VL)	R/W	<a href="#">12.4.7/-190</a>
<b>FlexTimer specific registers</b>			
Base2+0x0000	FTM Counter Initial Value High (FTMxCNTINH)	R/W	<a href="#">12.4.8/-191</a>
Base2+0x0001	FTM Counter Initial Value Low (FTMxCNTINL)	R/W	<a href="#">12.4.8/-191</a>
Base2+0x0002	FTM Capture and Compare Status Register (FTMxSTATUS)	R/W	<a href="#">12.4.9/-192</a>
Base2+0x0003	FTM Features Mode Selection (FTMxMODE)	R/W	<a href="#">12.4.10/-193</a>
Base2+0x0004	FTM Synchronization (FTMxSYNC)	R/W	<a href="#">12.4.11/-195</a>
Base2+0x0005	FTM Initial State for Channels Output (FTMxOUTINIT)	R/W	<a href="#">12.4.12/-196</a>
Base2+0x0006	FTM Output Mask (FTMxOUTMASK)	R/W	<a href="#">12.4.13/-196</a>
Base2+0x0007	FTM Function For Linked Channels (FTMxCOMBINE0)	R/W	<a href="#">12.4.14/-197</a>
Base2+0x0008	FTM Function For Linked Channels (FTMxCOMBINE1)	R/W	<a href="#">12.4.14/-197</a>
Base2+0x0009	FTM Function For Linked Channels (FTMxCOMBINE2)	R/W	<a href="#">12.4.14/-197</a>
Base2+0x000A	FTM Function For Linked Channels (FTMxCOMBINE3)	R/W	<a href="#">12.4.14/-197</a>
Base2+0x000B	FTM Deadtime Insertion Control (FTMxDEADTIME)	R/W	<a href="#">12.4.15/-198</a>
Base2+0x000C	FTM External Trigger (FTMxEXTTRIG)	R/W	<a href="#">12.4.16/-201</a>
Base2+0x000D	FTM Channels Polarity (FTMxPOL)	R/W	<a href="#">12.4.17/-202</a>
Base2+0x000E	FTM Fault Mode Status (FTMxFMS)	R/W	<a href="#">12.4.18/-202</a>
Base2+0x000F	FTM Input Capture Filter Control (FTMxFILTER0)	R/W	<a href="#">12.4.19/-203</a>
Base2+0x0010	FTM Input Capture Filter Control (FTMxFILTER1)	R/W	<a href="#">12.4.19/-203</a>
Base2+0x0011	FTM Fault Input Filter Control (FTMxFLTFILTER)	R/W	<a href="#">12.4.20/-204</a>
Base2+0x0012	FTM Fault Input Control (FTMxFLTCTRL)	R/W	<a href="#">12.4.21/-204</a>
Base2+0x0013	FTM Register for Future Use (FTMxRFU)	R	<a href="#">12.4.22/-205</a>

<sup>1</sup> Read-only for normal access; however, any write to FTMxCNTH or FTMxCNTL causes the 16-bit FTM counter to be updated with the value of FTMxCNTINH:FTMxCNTINL registers.

The FTM memory map can be split into two sets of registers (the first set initial address is Base1 and the second set initial address is Base2) or be only one set (when the second set initial address is Base1 + 0x001D). Refer to the memory map in full-chip specification for the specific chip implementation.

The first set has the original TPM registers. The space of the first set registers not implemented are compressed like in TPM memory map.

The second set has the FTM specific registers and this set can be placed in high memory space. Any second set registers (or bits within these registers) that are used by an unavailable function in the FTM configuration remain in the memory map and in the reset value, so they have no active function.

## NOTE

Although it is possible to write to the FTM specific registers (second set registers) when FTMEN = 0, it is not recommended and the results can be not guaranteed.

### 12.4.2 Register Descriptions

This section consists of register descriptions in address order. A typical MCU system may contain multiple FTMs and each FTM may have up to eight channels, so register names include placeholder characters to identify which FTM and which channel is being referenced. For example, FTMxCnSC refers to FTM (x) and channel (n). FTM1C2SC would be the status and control register for channel 2 of FTM1.

### 12.4.3 FTM Status and Control Register (FTMxSC)

FTMxSC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescale factor. These controls relate to all channels within this module.

	7	6	5	4		3	2	1	0
R	TOF	TOIE	CPWMS	CLKS		PS			
W	0								
Reset	0	0	0	0		0	0	0	0

Figure 12-3. FTM Status and Control Register (FTMxSC)

Table 12-6. FTMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the FTM counter resets to the value of FTMxCNTINH:FTMxCNTINL registers after reaching the modulo value programmed in the FTMxMODH:FTMxMODL registers. Clear TOF by reading the FTM status and control register when TOF is set and then writing a logic 0 to TOF. If another FTM 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.  When the FTM is configured for CPWM mode (up-down counter), TOF is set after the FTM counter has reached the value in the FTMxMODH:FTMxMODL registers, at the transition to the next lower count value. 0 FTM counter has not reached the modulo value or overflowed. 1 FTM counter has overflowed.
6 TOIE	Timer overflow interrupt enable. This read/write bit enables FTM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use software polling). 1 TOF interrupts enabled.
5 CPWMS	Center-aligned PWM select. This read/write bit selects CPWM mode. This mode configures the FTM to operate in up-down counting mode. Reset clears CPWMS. CPWMS is write protected, this bit can only be written if WPDIS = 1. 0 FTM counter operates in up counting mode. 1 FTM counter operates in up-down counting mode.

**Table 12-6. FTMxSC Field Descriptions (continued)**

Field	Description
4–3 CLKS	Clock source select. As shown in <a href="#">Table 12-7</a> , this 2-bit field is used to disable the FTM counter or select one of four clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL or FLL-based system clock. When there is no PLL and no FLL, the fixed-system clock source is the same as the system clock. The external source is synchronized to the system clock by FTM, and the fixed system clock source (when a PLL or a FLL is present) is synchronized to the system clock by an on-chip synchronization circuit. When a PLL or a FLL is present but not enabled, the fixed-system clock source is the same as the system clock. CLKS[1:0] bits are write protected, these bits can only be written if WPDIS = 1.
2–0 PS	Prescale factor select. This 3-bit field selects one of 8 division factors for the FTM clock input as shown in <a href="#">Table 12-8</a> . This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the FTM system. The new prescaler factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits. PS[2:0] bits are write protected, these bits can only be written if WPDIS = 1.

**Table 12-7. FTM Clock Source Selection**

CLKS	FTMEN	FTM Clock Source to Prescaler Input
00	X	No clock selected (FTM counter disable)
01	0	System clock divided by 2 (for TPM compatibility)
	1	System clock
10	X	Fixed system clock
11	X	External source

**Table 12-8. Prescale Factor Selection**

PS	FTM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

#### 12.4.4 FTM Counter Registers (FTMxCNTH:FTMxCNTL)

The two read-only FTM counter registers contain the high and low bytes of the value in the FTM counter. Reading either byte (FTMxCNTH or FTMxCNTL) 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 FTM status and control register (FTMxSC).

Reset clears the FTM counter registers. Writing any value to FTMxCNTH or FTMxCNTL updates the FTM counter (FTMxCNTH:FTMxCNTL) with its initial value (FTMxCNTINH:FTMxCNTINL) and resets the read 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 FTMxCNTH updates the 16-bit counter with its initial value (FTMxCNTINH:FTMxCNTINL)									
Reset	0	0	0	0		0	0	0	0

Figure 12-4. FTM Counter Register High (FTMxCNTH)

	7	6	5	4		3	2	1	0
R	Bit 7	6	5	4		3	2	1	Bit 0
W Any write to FTMxCNTL updates the 16-bit counter with its initial value (FTMxCNTINH:FTMxCNTINL)									
Reset	0	0	0	0		0	0	0	0

Figure 12-5. FTM Counter Register Low (FTMxCNTL)

When BDM is active, the FTM counter is frozen (this is the value that will be read by user); the read 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.

## 12.4.5 FTM Counter Modulo Registers (FTMxMODH:FTMxMODL)

The read/write FTM modulo registers contain the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method ([Section 12.5.3, “Counter”](#)).

Reset sets the FTM counter modulo registers to 0x0000.

Writing to FTMxMODH or FTMxMODL inhibits the TOF bit and overflow interrupts until these registers are updated with the value of their write buffer ([12.5.10, “Load of the registers with write buffers”](#)).

### NOTE

The update of the TOF bit and the generation of the overflow interrupt are disabled when there is a write to FTMxMODH or FTMxMODL. This condition will remain until the FTMxMODH:L registers are updated with the contents of their write buffers. This could be several timer count periods.

Writing to either byte (FTMxMODH or FTMxMODL) latches the value into a buffer. The registers are updated with the value of their write buffer according to [Section 12.5.10, “Load of the registers with write buffers”](#).

If FTMEN = 0, then this write coherency mechanism may be manually reset by writing to the FTMxSC register (whether BDM is active or not).

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

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

**Figure 12-6. FTM Counter Modulo Register High (FTMxMODH)**

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

**Figure 12-7. FTM Counter Modulo Register Low (FTMxMODL)**

It is recommended to initialize the FTM counter (write to FTMxCNTH or FTMxCNTL) before writing to the FTM modulo registers to avoid confusion about when the first counter overflow will occur.

## 12.4.6 FTM Channel (n) Status and Control Register (FTMxCnSC)

FTMxCnSC 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 12-8. FTM Channel (n) Status and Control Register (FTMxCnSC)**

**Table 12-9. FTMxCnSC Field Descriptions**

Field	Description
7 CHnF	<p>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 PWM channel, CHnF is set when the value in the FTM counter matches the value in the FTM channel (n) value registers. When channel (n) is center-aligned PWM channel, CHnF is set twice during each CPWM cycle (one at the start and another at the end of the active duty cycle period) when the value in the FTM counter matches the value in the FTM 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 FTM counter registers matches the value in the FTM channel (n) value registers. When channel (n) is in combine mode, CHnF is always set when the value in the FTM counter matches the value in the FTM channel.</p> <p>A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading FTMxCnSC 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 is completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF.</p> <p>Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect.</p> <ul style="list-style-type: none"> <li>0 No input capture or match event occurred on channel (n)</li> <li>1 Input capture or match event on channel (n)</li> </ul>
6 CHnIE	Channel (n) interrupt enable. This read/write bit enables interrupts from channel (n). Reset clears CHnIE.
5 MSnB	Mode select B for FTM channel (n). Refer to the summary of channel mode and setup controls in <a href="#">Table 12-10</a> . MSnB is write protected, this bit can only be written if WPDIS = 1.
4 MSnA	Mode select A for FTM channel (n). Refer to the summary of channel mode and setup controls in <a href="#">Table 12-10</a> . MSnA is write protected, this bit can only be written if WPDIS = 1. <b>Note:</b> If the associated port pin is not stable for at least two system clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	<p>Edge/level select bits. Depending upon the operating mode for the timer channel as set by COMBINE:CPWMS:MSnB:MSnA bits and shown in <a href="#">Table 12-10</a>, these bits can select which edge of the channel input signal is an input capture event or the level that will be driven in response to the channel or initial match according to the selected output mode.</p> <p>ELSnB and ELSnA bits are write protected, these bits can only be written if WPDIS = 1.</p> <p>Setting ELSnB:ELSnA to 0:0 configures the related FTM pin as a general purpose I/O pin not related to any FTM functions. This function is typically used to temporarily disable an input capture channel or to make the FTM pin available as a general purpose I/O pin.</p> <p>When ELSnB:ELSnA are set to 0:0 the associated FTM channel physical output is disabled however, compare and match events will continue to set the appropriate flags.</p>

**Table 12-10. Mode, Edge, and Level Selection**

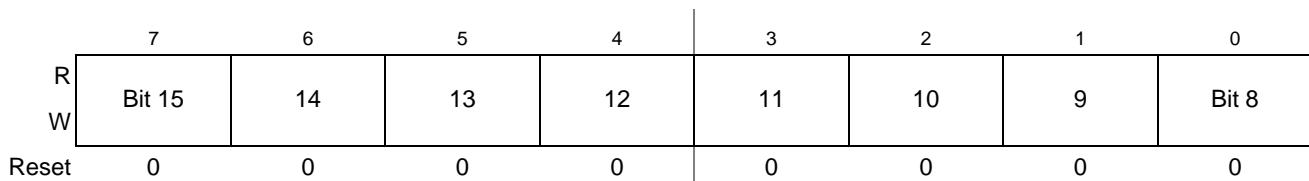
COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	X	XX	00	Pin not used for FTM - revert the channel pin to general purpose I/O or other peripheral control	

**Table 12-10. Mode, Edge, and Level Selection**

COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	0	00	01	Input capture	Capture on rising edge only
			10		Capture on falling edge only
			11		Capture on rising or falling edge
		01	01	Output compare	Toggle output on match
			10		Clear output on match
			11		Set output on match
		1X	10	Edge-aligned PWM	High-true pulses (clear output on match)
			X1		Low-true pulses (set output on match)
		XX	10	Center-aligned PWM	High-true pulses (clear output on match-up)
			X1		Low-true pulses (set output on match-up)
		XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
			X1		Low-true pulses (clear on channel (n) match, and set on channel (n+1) match)

#### 12.4.7 FTM Channel Value Registers (FTMxCnVH:FTMxCnVL)

These read/write registers contain the captured FTM counter value of the input capture function or the match value for the output modes. The channel registers are cleared by reset.



**Figure 12-9. FTM Channel Value Register High (FTMxCnVH)**

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

Figure 12-10. FTM Channel Value Register Low (FTMxCnVL)

In input capture mode, reading either byte (FTMxCnVH or FTMxCnVL) 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 FTMxCnSC 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 read 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 value 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. Any read of the FTMxCnVH and FTMxCnVL registers in BDM mode bypasses the buffer latches and returns the value of these registers and not the value of their read buffer.

In output modes, writing to either byte (FTMxCnVH or FTMxCnVL) latches the value into a buffer. The registers are updated with the value of their write buffer according to [Section 12.5.10, “Load of the registers with write buffers”](#).

If FTMEN = 0, then this write coherency mechanism may be manually reset by writing to the FTMxCnSC 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 write 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 value register are written while BDM is active. Any write to the FTMxCnVH and FTMxCnVL registers bypasses the buffer latches and writes directly to the register while BDM is active. The values written to the channel value registers while BDM is active are used in output modes operation once normal execution resumes. Writes to the channel value registers while BDM is active do not interfere with the partial completion of a coherency sequence. After the write coherency mechanism has been fully exercised, the channel value registers are updated using the buffered values written (while BDM was not active) by the user.

## 12.4.8 FTM Counter Initial Value Registers (FTMxCNTINH:FTMxCNTINL)

The read/write FTM counter initial value registers contain the initial value for the FTM counter.

Writing to either byte (FTMxCNTINH or FTMxCNTINL) latches the value into a buffer and the FTMxCNTINH:FTMxCNTINL registers are updated when the second byte is written.

When BDM is active, the write 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 counter initial value register are written while BDM is active. Any write to the counter initial value registers bypasses the buffer latches and writes directly to the counter initial value register while BDM is active.

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

Figure 12-11. FTM Counter Initial Value Register High (FTMxCNTINH)

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

Figure 12-12. FTM Counter Initial Value Register Low (FTMxCNTINL)

The first time that the FTM clock is selected (first write to change the CLKS[1:0] bits to a non-zero value), FTM counter will start with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the FTM counter initial value registers (FTMxCNTINH:FTMxCNTINL) and then initialize the FTM counter (write a value to FTMxCNTH or FTMxCNTL).

#### 12.4.9 FTM Capture and Compare Status Register (FTMxSTATUS)

FTMxSTATUS contains a copy of the status flag CHnF bit (in FTMxCnSC register) for each FTM channel for simpler software driver.

	7	6	5	4	3	2	1	0
R	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0
= Unimplemented or Reserved								

Figure 12-13. FTM Capture and Compare Status Register (FTMxSTATUS)

**Table 12-11. FTMxSTATUS Field Descriptions**

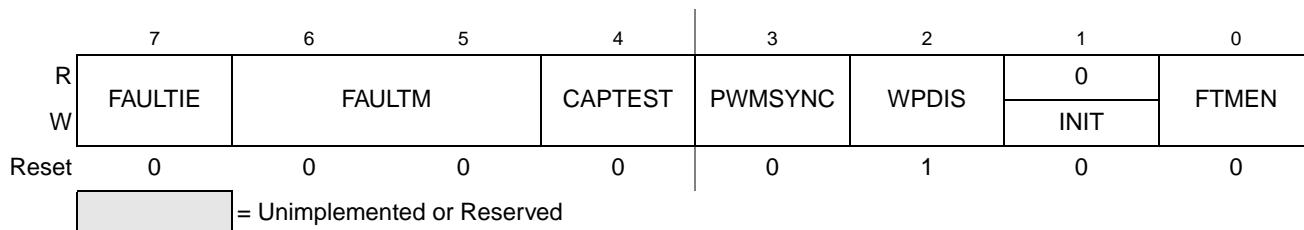
Field	Description
7-0 CHnF	<p>Channel (n) flag. When channel (n) is an input-capture channel, this read/write bit is set when an active edge occurs on the channel (n) pin. When channel (n) is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the FTM counter registers matches the value in the FTM 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 FTM counter registers matches the value in the FTM channel (n) value registers. When channel (n) is in combine mode, CHnF is always set when the value in the FTM counter matches the value in the FTM channel (n) value.</p> <p>Clear CHnF by reading FTMxSTATUS while CHnF is set and then writing a logic 0 to the selected CHnF. If another request to set the CHnF bit occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence is completed for the earlier CHnF.</p> <p>Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect.</p> <p>0 No input capture or match event occurred on channel (n) 1 Input capture or match event on channel (n)</p> <p><b>Note:</b> Each CHnF bit in FTMxSTATUS register is a copy of CHnF bit in FTMxCnSC register. All CHnF bits can be checked using only one read of FTMxSTATUS register. And all CHnF bits can be cleared by a read of FTMxSTATUS register followed by a write 0x00 to FTMxSTATUS register.</p>

#### NOTE

The use of FTMxSTATUS register is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). The use of this register with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

#### 12.4.10 FTM Features Mode Selection Register (FTMxMODE)

This read/write register contains the control bits used to configure the fault interrupt and fault control mode, capture test mode, PWM synchronization, write protection, channel output initialization and enable the enhanced features of the FTM. These controls relate to all channels within this module.



**Figure 12-14. FTM Features Mode Selection Register (FTMxMODE)**

**Table 12-12. FTMxMODE Field Descriptions**

Field	Description
7 FAULTIE	Fault interrupt enable. This read/write bit enables the generation of an interrupt when a fault is detected by FTM and the FTM fault control is enabled. 0 Fault control interrupt is disabled. 1 Fault control interrupt is enabled.
6-5 FAULTM	Fault control mode. These bits define the FTM fault control mode. The fault control mode is selected according to the <a href="#">Table 12-13</a> . FAULTM[1:0] bits are write protected, these bits can only be written if WPDIS = 1.
4 CAPTEST	Capture test mode enable. This read/write bit enables the capture test mode. CAPTEST bit is write protected, this bit can only be written if WPDIS = 1. 0 Capture test mode is disabled. 1 Capture test mode is enabled.
3 PWMSYNC	PWM synchronization mode. This read/write bit selects which triggers can be used by FTMxMODH:L, FTMxCnVH:L, CHnOM and FTM counter synchronization ( <a href="#">Section 12.5.11, “PWM synchronization”</a> ). 0 No restrictions. Software and hardware triggers can be used by FTMxMODH:L, FTMxCnVH:L, CHnOM and FTM counter synchronization. 1 Software trigger can only be used by FTMxMODH:L and FTMxCnVH:L synchronization. And hardware triggers can only be used by CHnOM and FTM counter synchronization.
2 WPDIS	Write protection disable. When write protection is enabled (WPDIS = 0), write protected bits can not be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a logic 1 and then 1 is written to WPDIS. Write 0 to WPDIS has no effect. 0 Write protection is enabled. 1 Write protection is disabled.
1 INIT	Initialize the output channels. When a logic 1 is written to INIT bit the output channels are initialized according to the state of their corresponding bit in the FTMxOUTINIT register. Writing a logic 0 to INIT bit has no effect. 0 The INIT bit is always read as logic 0.
0 FTMEN	FTM enable. When FTMEN = 1 all registers including the FTM specific registers (second set registers) are available for use with no restrictions. When FTMEN = 0 only the TPM compatible registers (first set registers) can be used without any restriction. The use of the FTM specific registers when FTMEN = 0 is not recommended and can result in unpredictable behavior. FTMEN bit is write protected, this bit can only be written if WPDIS = 1. 0 Only TPM compatible registers (first set registers) are available. 1 All FTM registers are available.

**Table 12-13. Fault Control Mode Selection**

FAULTM	Fault Control Mode
00	Fault control mode is disabled for all channels
01	Fault control mode is enabled for even channels only (channels 0, 2, 4 and 6) and the selected mode is the manual fault clearing
10	Fault control mode is enabled for all channels and the selected mode is the manual fault clearing
11	Fault control mode is enabled for all channels and the selected mode is the automatic fault clearing

## 12.4.11 FTM Synchronization Register (FTMxSYNC)

This read/write register configures the PWM synchronization.

A synchronization event can perform the synchronized update of FTMxMODH:FTMxMODL, FTMxCnVH:FTMxCnVL and FTMxOUTMASK registers with the value of their write buffer and the FTM counter initialization.

	7	6	5	4	3	2	1	0
R W	SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
Reset	0	0	0	0	0	0	0	0
	= Unimplemented or Reserved							

Figure 12-15. FTM Synchronization Register (FTMxSYNC)

Table 12-14. FTMxSYNC Field Descriptions

Field	Description
7 SWSYNC	PWM synchronization software trigger. SWSYNC bit selects the software trigger as the PWM synchronization trigger. The software trigger happens when a logic 1 is written to SWSYNC bit. 0 Software trigger is not selected. 1 Software trigger is selected.
6 TRIG2	PWM synchronization external trigger 2. TRIG2 bit selects external trigger 2 as the PWM synchronization trigger. External trigger 2 happens when the FTM detects a rising edge in the trigger 2 input signal. 0 External trigger 2 is not selected. 1 External trigger 2 is selected.
5 TRIG1	PWM synchronization external trigger 1. TRIG1 bit selects external trigger 1 as the PWM synchronization trigger. External trigger 1 happens when the FTM detects a rising edge in the trigger 1 input signal. 0 External trigger 1 is not selected. 1 External trigger 1 is selected.
4 TRIG0	PWM synchronization external trigger 0. TRIG0 bit selects external trigger 0 as the PWM synchronization trigger. External trigger 0 happens when the FTM detects a rising edge in the trigger 0 input signal. 0 External trigger 0 is not selected. 1 External trigger 0 is selected.
3 SYNCHOM	Output mask synchronization. SYNCHOM bit selects when the CHnOM bits in register FTMxOUTMASK are updated with the value of their write buffer. 0 CHnOM bits are updated with the value of the FTMxOUTMASK write buffer in all rising edges of the system clock. 1 CHnOM bits are updated with the value of the FTMxOUTMASK write buffer by the PWM synchronization.
2 REINIT	Reinitialization of the FTM by PWM synchronization (Section 12.5.11, “PWM synchronization”). REINIT bit determines if the FTM is initialized when the selected trigger for the PWM synchronization is detected. 0 FTM counter continues to count normally. 1 FTM counter is updated with its initial value when the selected trigger is detected.

Field	Description
1 CNTMAX	Maximum boundary cycle enable. The CNTMAX bit determines when the FTMxMODH:L and FTMxCnVH:L registers are updated with their write buffer contents following a PWM synchronization event. If CNTMAX is enabled, the registers are updated when the FTM counter reaches its maximum value FTMxMODH:L. 0 The maximum boundary cycle is disabled. 1 The maximum boundary cycle is enabled.
0 CNTMIN	Minimum boundary cycle enable. The CNTMIN bit determines when the FTMxMODH:L and FTMxCnVH:L registers are updated with their write buffer contents following a PWM synchronization event. If CNTMIN is enabled, the registers are updated when the FTM counter reaches its minimum value FTMxCNTINH:L. 0 The minimum boundary cycle is disabled. 1 The minimum boundary cycle is enabled.

### NOTE

The software trigger (SWSYNC bit) and hardware triggers (TRIG0, TRIG1 and TRIG2 bits) have a potential conflict if used together. It is recommended using only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely. The selection of the boundary cycle (CNTMAX and CNTMIN bits) is intended to provide the update of FTMxCnVH:FTMxCnVL across all enabled channels simultaneously. The use of the boundary cycle selection together TRIG0, TRIG1 or TRIG2 bits is likely to result in an unpredictable behavior.

#### 12.4.12 FTM Initial State for Channels Output Register (FTMxOUTINIT)

This read/write register defines the value that is forced in the channels output by the initialization feature (when a logic 1 is written to the INIT bit in register FTMxMODE).

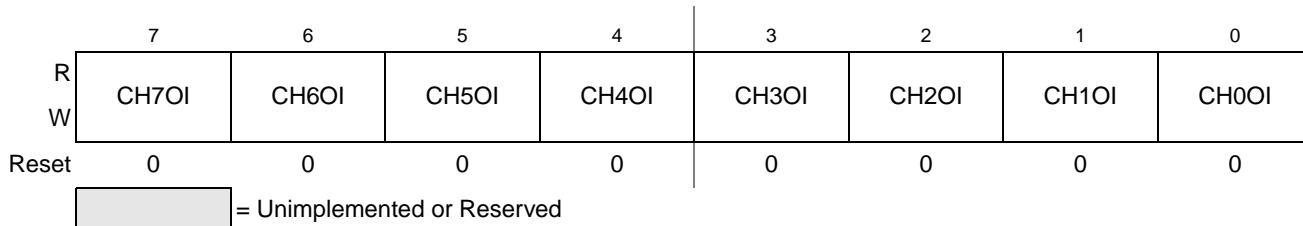


Figure 12-16. FTM Initial State for Channels Output Register (FTMxOUTINIT)

Table 12-15. FTMxOUTINIT Field Descriptions

Field	Description
7-0 CHnOI	Channel (n) output initialization value. CHnOI bit selects the value that is forced into channel (n) output when the initialization occurs. 0 The initialization value is the logical value 0. 1 The initialization value is the logical value 1.

#### 12.4.13 FTM Output Mask Register (FTMxOUTMASK)

This read/write register provides a mask for each FTM channel. The mask of a channel determines if its output will respond (that is, it will be masked or not) when a match occurs. This feature is used for BLDC

control where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the FTMxOUTMASK register, stores the value into a write buffer. The register is updated with the value of its write buffer according to [Section 12.5.11, “PWM synchronization”](#).

	7	6	5	4		3	2	1	0
R W	CH7OM	CH6OM	CH5OM	CH4OM		CH3OM	CH2OM	CH1OM	CH0OM
Reset	0	0	0	0		0	0	0	0
= Unimplemented or Reserved									

**Figure 12-17. FTM Output Mask Register (FTMxOUTMASK)**

**Table 12-16. FTMxOUTMASK Field Descriptions**

Field	Description
7-0 CHnOM	Channel (n) output mask. CHnOM bit defines if the channel (n) output is masked (forced to its inactive state) or unmasked (it continues to operate normally). 0 Channel (n) output is not masked. It continues to operate normally. 1 Channel (n) output is masked. It is forced to its inactive state.

#### 12.4.14 FTM Function For Linked Channels Register (FTMxCOMBINEm)

This read/write register contains the control bits used to configure the fault control, synchronization, deadtime, complementary and combine features of channels (n) and (n+1).

The FTMxCOMBINE0 register configures the control bits for channels 0 and 1; FTMxCOMBINE1 for channels 2 and 3; FTMxCOMBINE2 for channels 4 and 5; and FTMxCOMBINE3 for channels 6 and 7.

	7	6	5	4		3	2	1	0
R W	0	FAULTEN	SYNCEN	DTEN		0	0	COMP	COMBINE
Reset	0	0	0	0		0	0	0	0
= Unimplemented or Reserved									

**Figure 12-18. FTM Function For Linked Channels Register (FTMxCOMBINEm)**

**Table 12-17. FTmxCOMBINEm Field Descriptions**

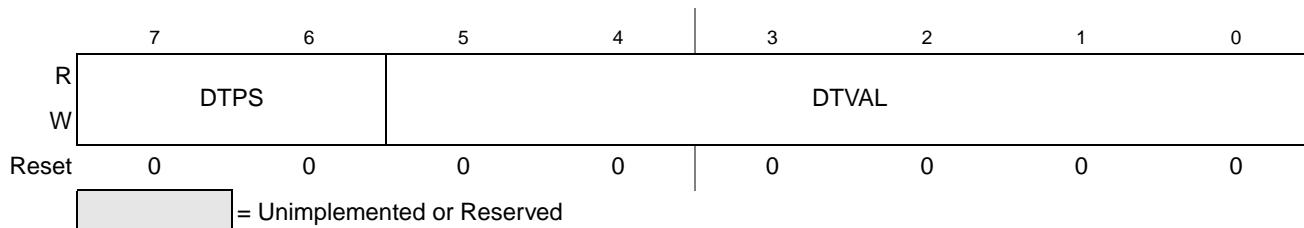
Field	Description
6 FAULTEN	Fault control enable. The FAULTEN bit enables the fault control in channels (n) and (n+1). FAULTEN is write protected, this bit can only be written if WPDIS = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
5 SYNCEN	Synchronization enable. The SYNCEN bit enables PWM synchronization of registers FTmxC(n)VH:FTmxC(n)VL and FTmxC(n+1)VH:FTmxC(n+1)VL. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
4 DTEN	Deadtime enable. The DTEN bit enables the deadtime insertion in the channels (n) and (n+1). DTEN is write protected, this bit can only be written if WPDIS = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
1 COMP	Complement of channel (n). The COMP bit enables complementary mode for the combined channels. In the complementary mode channel (n+1) output is the inverse of channel (n) output. COMP is write protected, this bit can only be written if WPDIS = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
0 COMBINE	Combine channels (n) and (n+1). The COMBINE bit enables the combine feature for channels (n) and (n+1). COMBINE is write protected, this bit can only be written if WPDIS = 1. 0 Channels (n) and (n+1) are independent. 1 Combine mode is enabled for channels (n) and (n+1).

#### **NOTE**

The channel (n) is the even channel and the channel (n+1) is the odd channel of a pair of channels.

#### **12.4.15 FTM Deadtime Insertion Control Register (FTMxDEADTIME)**

This read/write register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.



**Figure 12-19. FTM Deadtime Insertion Control Register (FTMxDEADTIME)**

**Table 12-18. FTmxDEADTIME Field Descriptions**

Field	Description
7–6 DTPS	Deadtime prescaler value. DTPS[1:0] bits select the division factor of the system clock (as shown in <a href="#">Table 12-19</a> ). This prescaled clock is used by the deadtime counter. DTPS[1:0] bits are write protected, these bits can only be written if WPDIS = 1.
5–0 DTVAL	Deadtime value. DTVAL[5:0] bits select the deadtime value using the deadtime prescaled clock (as shown in <a href="#">Table 12-19</a> ). The deadtime insertion in all channels is disabled when DTVAL[5:0] is zero. DTVAL[5:0] bits are write protected, these bits can only be written if WPDIS = 1.

**Table 12-19. Deadtime Prescale Factor Selection**

DTPS	System Clock Divided-by
0X	1
10	4
11	16

**Table 12-20. Deadtime Value (number of system clock periods)**

DTVAL	DTPS = 0X	DTPS = 10	DTPS = 11
0	Deadtime insertion is disabled		
1	1	4	16
2	2	8	32
3	3	12	48
4	4	16	64
5	5	20	80
6	6	24	96
7	7	28	112
8	8	32	128
9	9	36	144
10	10	40	160
11	11	44	176
12	12	48	192
13	13	52	208
14	14	56	224
15	15	60	240
16	16	64	256
17	17	68	272
18	18	72	288
19	19	76	304
20	20	80	320

<b>DTVAL</b>	<b>DTPS = 0X</b>	<b>DTPS = 10</b>	<b>DTPS = 11</b>
21	21	84	336
22	22	88	352
23	23	92	368
24	24	96	384
25	25	100	400
26	26	104	416
27	27	108	432
28	28	112	448
29	29	116	464
30	30	120	480
31	31	124	496
32	32	128	512
33	33	132	528
34	34	136	544
35	35	140	560
36	36	144	576
37	37	148	592
38	38	152	608
39	39	156	624
40	40	160	640
41	41	164	656
42	42	168	672
43	43	172	688
44	44	176	704
45	45	180	720
46	46	184	736
47	47	188	752
48	48	192	768
49	49	196	784
50	50	200	800
51	51	204	816
52	52	208	832
53	53	212	848
54	54	216	864
55	55	220	880
56	56	224	896

DTVAL	DTPS = 0X	DTPS = 10	DTPS = 11
57	57	228	912
58	58	232	928
59	59	236	944
60	60	240	960
61	61	244	976
62	62	248	992
63	63	252	1008

### 12.4.16 FTM External Trigger Register (FTMxEXTTRIG)

This read/write register indicates when a channel trigger was generated, enables the generation of a trigger when the FTM counter is equal to its initial value and selects which channels are used in the generation of the channel triggers.

Channels 0 and 1 are not used to generate channel triggers.

	7	6	5	4		3	2	1	0
R	TRIGF	INITTRIGEN	CH7TRIG	CH6TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG	
W	0								
Reset	0	0	0	0	0	0	0	0	0
= Unimplemented or Reserved									

Figure 12-20. FTM External Trigger Register (FTMxEXTTRIG)

Table 12-21. FTMxEXTTRIG Field Descriptions

Field	Description
7 TRIGF	Channel trigger flag. This read/write bit is set when a channel trigger is generated. Clear TRIGF bit by reading FTMxEXTTRIG while TRIGF is set and then writing a logic 0 to TRIGF. If another channel trigger is generated before the clearing sequence is complete, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF. Reset clears TRIGF. Writing a logic 1 to TRIGF has no effect. 0 No channel trigger was generated 1 A channel trigger was generated
6 INITTRIGEN	Initialization trigger enable. This read/write bit enables the generation of the trigger when the FTM counter is equal to its initial value. 0 The generation of initialization trigger is disabled 1 The generation of initialization trigger is enabled
5-0 CHjTRIG where j = 7, 6, 5, 4, 3, 2	Channel j trigger enable. These read/write bits enable the generation of a channel trigger when the FTM counter is equal to the FTMxC(j)VH:FTMxC(j)VL registers. Several FTM channels can be selected to generate multiple triggers in one PWM period. 0 The generation of channel (j) trigger is disabled. 1 The generation of channel (j) trigger is enabled.

### 12.4.17 FTM Channels Polarity Register (FTMxPOL)

This read/write register defines the output polarity of the FTM channels.

	7	6	5	4	3	2	1	0
R	POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
W	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0
= Unimplemented or Reserved								

Figure 12-21. FTM Channels Polarity Register (FTMxPOL)

Table 12-22. FTMxPOL Field Descriptions

Field	Description
7-0 POLn	Channel (n) polarity. The POLn bit defines the polarity of the channel (n) output. POLn is write protected, this bit can only be written if WPDIS = 1. 0 The channel (n) polarity is high (the active state is logical one and the inactive state is logical zero). 1 The channel (n) polarity is low (the active state is logical zero and the inactive state is logical one).

#### NOTE

The safe value that is driven in the channel (n) output when the fault control is enabled and a fault condition is detected is the inactive state of the channel (n) polarity. The channel (n) safe value is the value of its POL bit.

### 12.4.18 FTM Fault Mode Status Register (FTMxFMS)

This read/write register contains the fault detection flags, write protection enable bit and the logic OR of the enable fault inputs.

	7	6	5	4	3	2	1	0
R	FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0
W	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0
= Unimplemented or Reserved								

Figure 12-22. FTM Fault Mode Status Register (FTMxFMS)

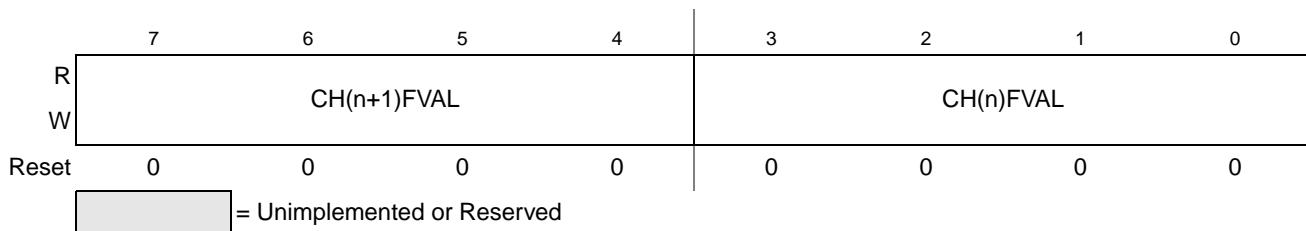
**Table 12-23. FTMxFMS Field Descriptions**

Field	Description
7 FAULTF	Fault detection flag. The FAULTF bit is the logic OR of the FAULTFn bits where n = 3, 2, 1, 0. Clear FAULTF by reading the FTMxFMS register while FAULTF is set and then writing a logic 0 to FAULTF. If another fault condition is detected in an enabled fault input before the clearing sequence is complete, the sequence is reset so FAULTF remains set after the clearing sequence is completed for the earlier fault condition. FAULTF is also cleared when FAULTFn bits are cleared individually.  Reset clears FAULTF. Writing a logic 1 to FAULTF has no effect. 0 No fault condition was detected 1 A fault condition was detected
6 WPEN	Write protection enable. When write protection is enabled (WPEN = 1), write protected bits can not be written. When write protection is disabled (WPEN = 0), write protected bits can be written. The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a logic 1 and then 1 is written to WPDIS. Write 0 to WPEN has no effect. 0 Write protection is disabled. 1 Write protection is enabled.
5 FAULTIN	Fault inputs. The FAULTIN bit is the logic OR of the enabled fault input after its filter (if its filter is enabled) when fault control is enabled. Reset clears FAULTIN. Writing a logic 0 or a logic 1 to FAULTIN has no effect. 0 The value of the fault input is logic 0. 1 The value of the fault input is logic 1.
3-0 FAULTFn where n = 3, 2, 1, 0	Fault detection flag n. The FAULTFn bit is set when fault control is enabled, the fault input n is enabled and a fault condition is detected in the fault input n. Clear FAULTFn by reading the FTMxFMS register while FAULTFn is set and then writing a logic 0 to FAULTFn when the fault input n is low. If another fault condition is detected at fault input n before the clearing sequence is complete, the sequence is reset so FAULTFn remains set after the clearing sequence is completed for the earlier fault condition. FAULTFn bit is also cleared when FAULTF bit is cleared.  Reset clears FAULTFn. Writing a logic 1 to FAULTFn has no effect. 0 No fault condition was detected in the fault input n 1 A fault condition was detected in the fault input n.

#### 12.4.19 FTM Input Capture Filter Control Register (FTMxFILTERm)

This read/write register selects the filter value for the inputs of channels (n) and (n+1).

The FTMxFILTER0 register selects the filter value for the inputs of channels 0 and 1; the FTMxFILTER1 register selects the filter value for the inputs of channels 2 and 3. Channels 4 and 5 do not have an input filter.



**Figure 12-23. FTM Input Capture Filter Control Register (FTMxFILTERm)**

**Table 12-24. FTMxFILTERm Field Descriptions**

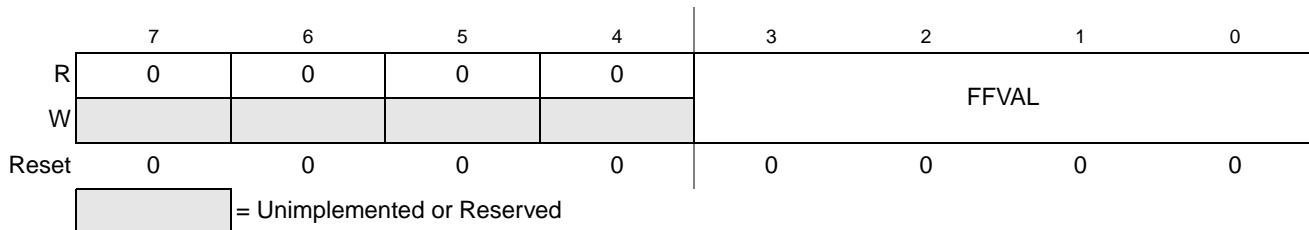
Field	Description
7-4 CH(n+1)FVAL	Channel (n+1) input filter. CH(n+1)FVAL[3:0] bits select the filter value for the channel (n+1) input. The channel (n+1) filter is disabled when CH(n+1)FVAL[3:0] is zero.
3-0 CH(n)FVAL	Channel (n) input filter. CH(n)FVAL[3:0] bits select the filter value for the channel (n) input. The channel (n) filter is disabled when CH(n)FVAL[3:0] is zero.

#### NOTE

Writing to this register has immediate effect and should only be done when the input capture modes of channels (n) and (n+1) are disabled. Failure to do so could result in a missing valid signal.

### 12.4.20 FTM Fault Input Filter Control Register (FTMxFLTFILTER)

This read/write register selects the filter value for the fault inputs.



**Figure 12-24. FTM Fault Input Filter Control Register (FTMxFLTFILTER)**

**Table 12-25. FTMxFLTFILTER Field Descriptions**

Field	Description
3-0 FFVAL	Fault input filter. FFVAL[3:0] bits select the filter value for the fault inputs. The fault filter is disabled when FFVAL[3:0] is zero.

#### NOTE

Writing to this register has immediate effect and should only be done when the fault control or the fault input is disabled. Failure to do so could result in a missing fault detection.

### 12.4.21 FTM Fault Control Register (FTMxFLTCTRL)

This read/write register selects the fault inputs and enables the fault input filter.



**Figure 12-25. FTM Fault Control Register (FTMxFLTCTRL)**

R	FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
W	0	0	0	0	0	0	0	0
Reset = Unimplemented or Reserved								

**Figure 12-25. FTM Fault Control Register (FTMxFLTCTRL)**

**Table 12-26. FTMxFLTCTRL Field Descriptions**

Field	Description
7-4 FFLTRnEN where n = 3, 2, 1, 0	Fault input n filter enable. This read/write bit enables the filter for the fault input n. FFLTRnEN is write protected, this bit can only be written if WPDIS = 1. 0 Fault input n filter is disabled. 1 Fault input n filter is enabled.
3-0 FAULTnEN where n = 3, 2, 1, 0	Fault input n enable. This read/write bit enables the fault input n. FAULTnEN is write protected, this bit can only be written if WPDIS = 1. 0 Fault input n is disabled. 1 Fault input n is enabled.

## 12.4.22 FTM Register for Future Use (FTMxRFU)

This register is reserved, however it will be used in the next versions of FTM. FTMxRFU register is always read as 0x00.

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

**Figure 12-26. FTM Register for Future Use (FTMxRFU)**

## 12.5 Functional Description

The following sections describe the FTM features.

The notation used in this document to represent the counters and the generation of the signals is shown in Figure 12-27.

Channel (n) - high-true EPWM

PS[2:0] = 001

FTMxCNTNH:L = 0x0000

FTMxMODH:L = 0x0004

FTMxCnVH:L = 0x0002

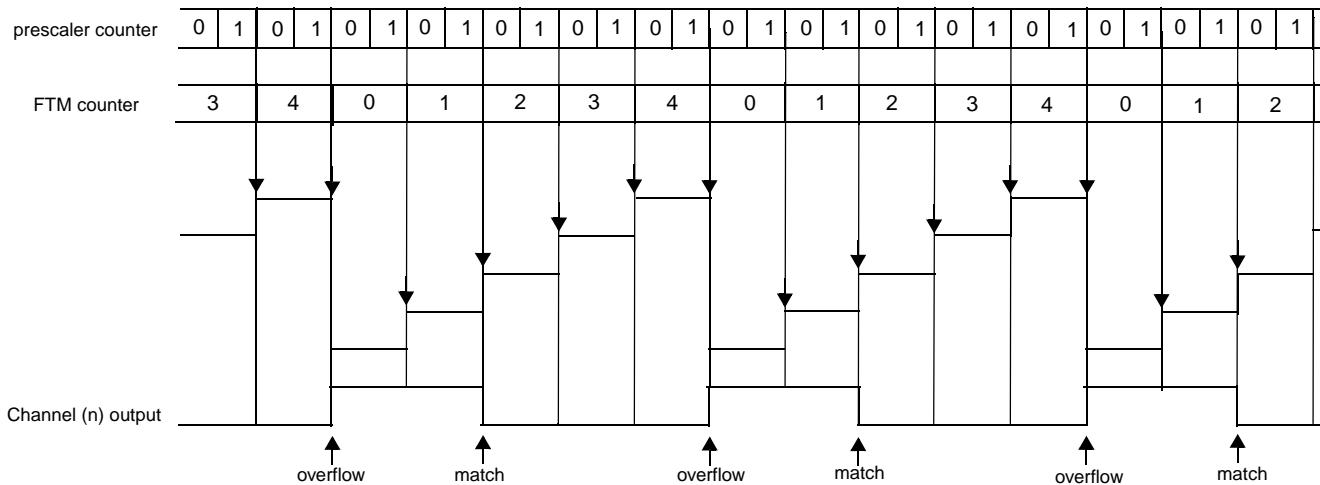


Figure 12-27. Notation used in the FTM Block Guide

## 12.5.1 Clock Source

FTM module has only one clock domain which is the system clock.

### 12.5.1.1 Counter Clock Source

The CLKS[1:0] bits in the FTMxSC register select one of four possible clock sources for the FTM counter or disables the FTM counter, see [Table 12-7](#). After any MCU reset, CLKS[1:0] = 00 so no clock source is selected, and the FTM is in a very low power state.

The CLKS[1:0] bits may be read or written at any time and disabling the FTM counter (writing 00 to the CLKS[1:0]) does not affect the FTM counter value or other registers.

The system clock and system clock divided by 2 require no synchronization. The system clock divided by 2 is provided for TPM compatibility.

The fixed system clock requires no synchronization in FTM module. In fact, a synchronization circuit is used at chip level to synchronize the crystal-related source clock to the system clock.

The external clock source may be connected to any FTM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to system clock transitions. The system clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency. With ideal clocks, the external clock can be as fast as system clock divided by four.

When the external clock source is shared with an FTM 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 FTM channel 0 pin was also being used as the external clock source. It is the user's responsibility to avoid such settings.

The FTM channel can still be used. When ELSnB:ELSnA are set to 0:0 the associated FTM channel physical output is disabled however, compare and match events will continue to set the appropriate flags.

## 12.5.2 Prescaler

The selected counter clock source passes through a prescaler which is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits (Table 12-8). Figure 12-28 shows an example of the prescaler counter and FTM counter.

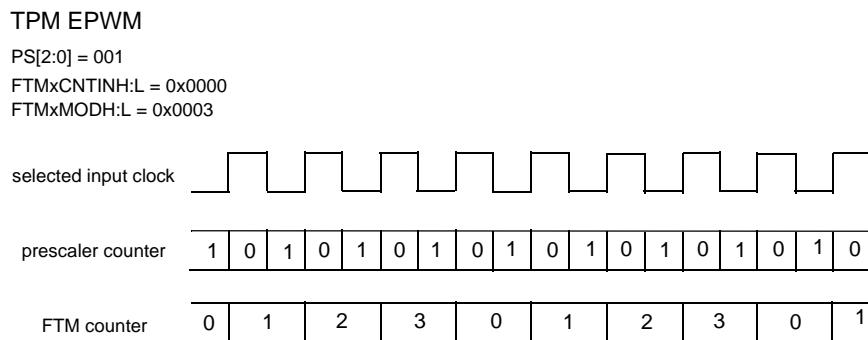


Figure 12-28. Example of the prescaler counter

## 12.5.3 Counter

The FTM has a 16-bit counter which is used to produce the channels output. The FTM counter clock is the clock generated using the prescaler counter (Section 12.5.2, "Prescaler").

The FTM counter can be an up or up-down counter according to the CPWMS bit.

### 12.5.3.1 Up counting

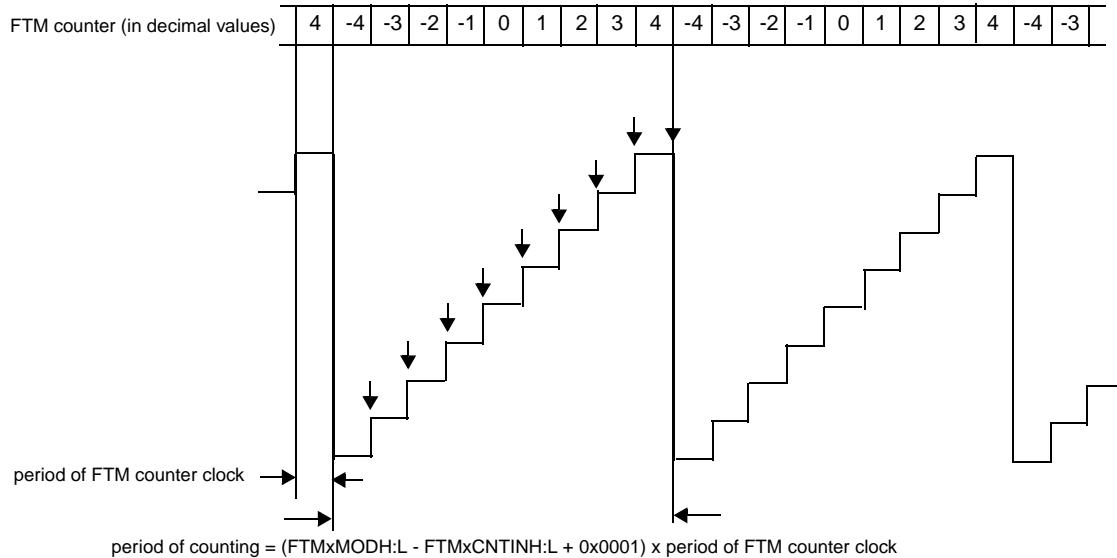
Up counting is selected when CPWMS = 0.

FTMxCNTINH:FTMxCNTINL defines the starting value of the count and FTMxMODH:FTMxMODL defines the final value of the count (Figure 12-29). The value of FTMxCNTINH:FTMxCNTINL is loaded into the FTM counter, and the counter incremented until the value of FTMxMODH:FTMxMODL is reached, at which point the counter is reloaded with the contents of FTMxCNTINH:FTMxCNTINL.

The FTM period when using up counting is  $(FTMxMODH:FTMxMODL - FTMxCNTINH:FTMxCNTINL + 0x0001) \times$  period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from FTMxMODH:FTMxMODL to FTMxCNTINH:FTMxCNTINL.

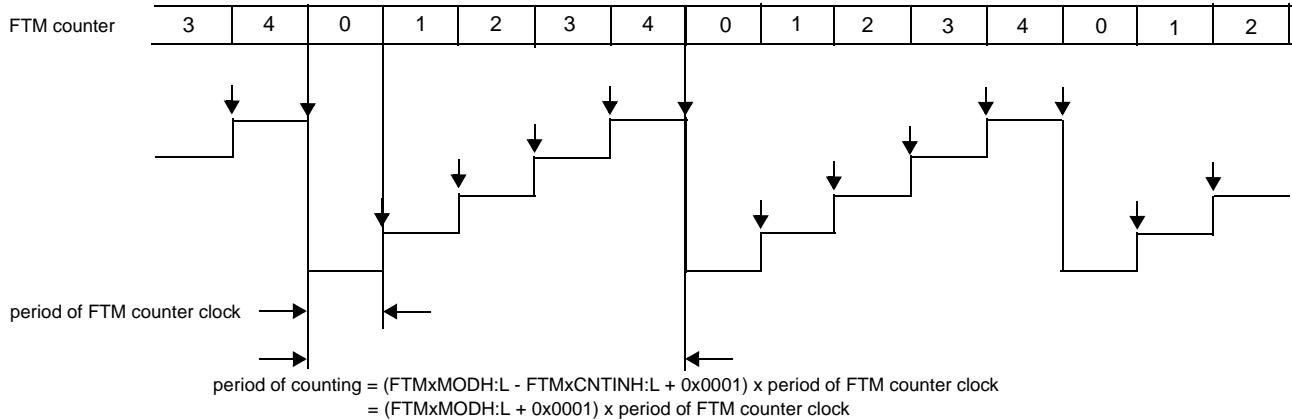
CPWMS = 0 (FTM counting is up)  
 FTmxCNTINH:L = 0xFFFF (in two's complement is equal to -4)  
 FTmxMODH:L = 0x0004



**Figure 12-29. Example of FTM up and signed counting**

If (FTmxCNTINH:FTMxCNTINL = 0x0000), the FTM counting is equivalent to TPM up counting (that is, up and unsigned counting) (Figure 12-30). If (FTmxCNTINH[7] = 1), then the initial value of the FTM counter is a negative number in two's complement, so the FTM counting is up and signed. Conversely if (FTmxCNTINH[7] = 0 and FTmxCNTINH:L not = 0x0000), then the initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned.

CPWMS = 0 (FTM counting is up)  
 FTMxCNTINH:L = 0x0000  
 FTMxMODH:L = 0x0004



**Figure 12-30. Example of FTM up counting with FTMxCNTINH:FTMxCNTINL = 0x0000**

#### NOTE

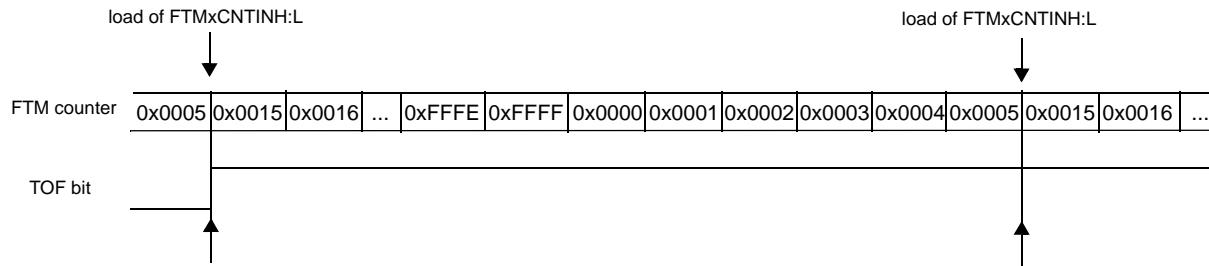
FTM operation is only valid when the value of the FTMxCNTINH:FTMxCNTINL registers is less than the value of the FTMxMODH:FTMxMODL registers (either in the unsigned counting or signed counting). It is the responsibility of the software to ensure that the values in the FTMxCNTINH:FTMxCNTINL and FTMxMODH:FTMxMODL registers meet this requirement. Any values of FTMxCNTINH:FTMxCNTINL and FTMxMODH:FTMxMODL that do not satisfy this criteria can result in unpredictable behavior.

FTMxMODH:FTMxMODL = FTMxCNTINH:FTMxCNTINL is a redundant condition. In this case, the FTM counter is always equal to FTMxMODH:FTMxMODL and the TOF bit will be set in each rising edge of the FTM counter clock.

When FTMxMODH:FTMxMODL = 0x0000 and FTMxCNTINH:FTMxCNTINL = 0x0000 (for example after reset) and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the FTMxMODH:FTMxMODL or FTMxCNTINH:FTMxCNTINL registers.

Setting FTMxCNTINH:L to be greater than the value of FTMxMODH:L is not recommended as this unusual setting may make the FTM operation difficult for users to comprehend. There is no restriction on this configuration however, and an example is shown in [Figure 12-31](#).

FTM counting is up (CPWMS = 0)  
 FTMxMODH:L = 0x0005  
 FTMxCNTINH:L = 0x0015



**Figure 12-31. Example of up counting when the value of FTMxCNTINH:FTMxCNTINL registers is greater than the value of FTMxMODH:FTMxMODL registers**

### 12.5.3.2 Up-down counting

Up-down counting is selected when (FTMEN = 0) and (CPWMS = 1).

FTMxCNTINH:FTMxCNTINL defines the starting value of the count and FTMxMODH:FTMxMODL the final value of the count ([Figure 12-32](#)). The value of FTMxCNTINH:FTMxCNTINL is loaded into the FTM counter, and the counter incremented until the value of FTMxMODH:FTMxMODL is reached, at which point the counter is decremented until it returns to the value of FTMxCNTINH:FTMxCNTINL and the up-down counting restarts.

The FTM period when using up-down counting is  $2 \times (\text{FTMxMODH:FTMxMODL} - \text{FTMxCNTINH:FTMxCNTINL}) \times \text{period of the FTM counter clock}$ .

The TOF bit is set when the FTM counter changes from FTMxMODH:FTMxMODL to (FTMxMODH:FTMxMODL - 1).

If (FTMxCNTINH:FTMxCNTINL = 0x0000), the FTM counting is equivalent to TPM up-down counting (that is, up-down and unsigned counting) ([Figure 12-32](#)).

CPWMS = 1 (FTM counting is up-down)

FTMxCNTINH:L = 0x0000

FTMxMODH:L = 0x0004

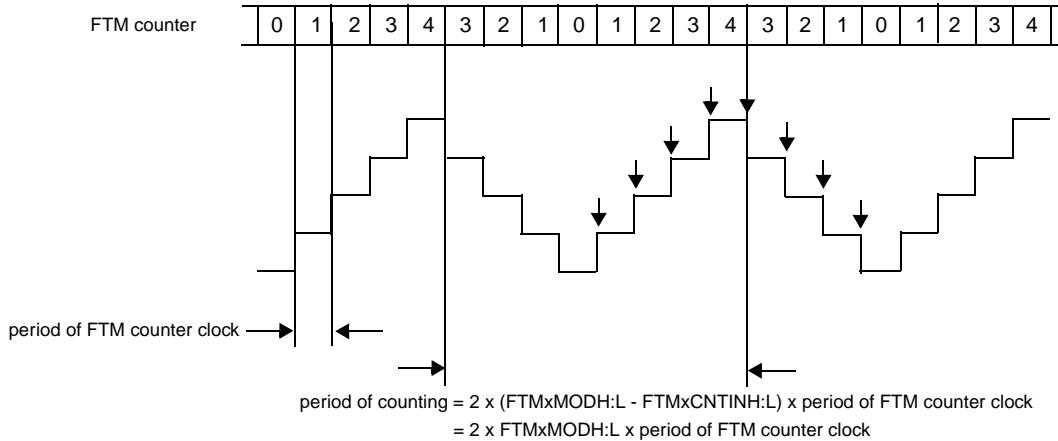


Figure 12-32. Example of up-down counting when  $\text{FTMxCNTINH:FTMxCNTINL} = 0x0000$

#### NOTE

The up-down counting is only available when ( $\text{FTMEN} = 0$ ) and ( $\text{FTMxCNTINH:FTMxCNTINL} = 0x0000$ ). The configuration with ( $\text{FTMEN} = 1$ ) or ( $\text{FTMxCNTINH:FTMxCNTINL} \neq 0x0000$ ) when ( $\text{CPWMS} = 1$ ) is not recommended and its results are not guaranteed.

#### 12.5.3.3 Free running counter

If ( $\text{FTMEN} = 0$ ) and ( $\text{FTMxMODH:FTMxMODL} = 0x0000$  or  $\text{FTMxMODH:FTMxMODL} = 0xFFFF$ ), the FTM counter is a free running counter. In this case the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000 (Figure 12-33).

FTM is compatible with TPM module ( $\text{FTMEN} = 0$ )

$\text{FTMxMODH:L} = 0x0000$

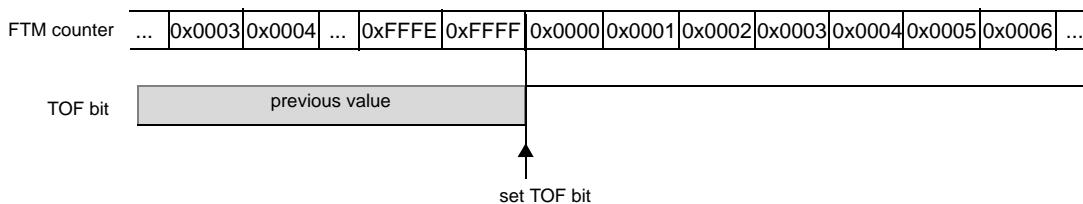


Figure 12-33. Example when the FTM counter is a free running

If ( $\text{FTMEN} = 1$ ) and ( $\text{CPWMS} = 0$ ) and ( $\text{FTMxCNTINH:FTMxCNTINL} = 0x0000$ ) and ( $\text{FTMxMODH:FTMxMODL} = 0xFFFF$ ), the FTM counter is a free running counter. In this case the FTM

counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000.

#### 12.5.3.4 Counter reset

Any write to FTMxCNTH or FTMxCNTL register resets the FTM counter to the value in the FTMxCNTINH:FTMxCNTINL registers.

PWM synchronization can also be used to force the value of FTMxCNTINH:FTMxCNTINL into the FTM counter (please see [Section 12.5.11, “PWM synchronization”](#)).

#### 12.5.4 Input capture mode

The input capture mode is selected when:

- (FTMEN = 0) and (COMBINE = 0) and (CPWMS = 0) and (MSnB:MSnA = 0:0) and (ELSnB:ELSnA not = 0:0)

When a selected edge occurs on the channel input pin, the current value of the FTM counter is captured into the FTMxCnVH:FTMxCnVL registers, the CHnF bit is set and the channel interrupt is generated if (CHnIE = 1)([Figure 12-34](#)). Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.

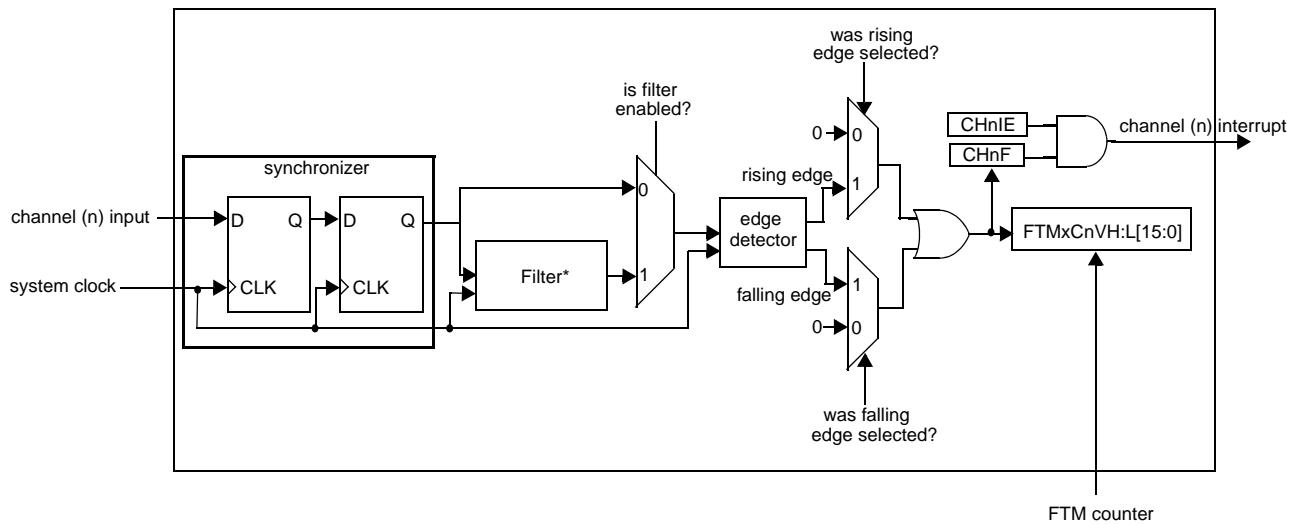
If a channel is configured for input capture, an internal pullup device may be enabled for that channel. The details of how the FTM interacts with pin controls depends upon the micro-controller implementation because the I/O pins and associated general purpose I/O controls are not part of the FlexTimer module. Refer to the I/O port control chapter of the micro-controller specification.

When a channel is configured for input capture, the FTMxCHn pin is forced to act as an edge-sensitive input to the FTM. ELSnB:ELSnA control bits determine which polarity edge or edges will trigger input-capture events. A synchronizer based on the system clock is used to synchronize input edges to the system clock. This implies the minimum pulse width that can be reliably detected on an input capture pin is four system clock periods (with ideal clock pulses as near as two system clocks can be detected). FTM uses this pin as a capture input to override the port data and data direction controls for the same pin.

When either half of the 16-bit capture register (FTMxCnVH:FTMxCnVL) is read, the other half is latched into a buffer to support coherent 16-bit access in big-endian or little-endian order. This read coherency mechanism can be manually reset by writing to FTMxCnSC register.

Writes to the FTMxCnVH:FTMxCnVL registers are ignored in input capture mode.

While in BDM, the input capture function works as configured by the user. When a selected edge event occurs, the FTM counter value (which is frozen because of BDM) is captured into the FTMxCnVH:FTMxCnVL registers and the CHnF bit is set.



\* Filtering function is only available in the inputs of channel 0, 1, 2 and 3

**Figure 12-34. Input capture mode**

If the channel input does not have a filter enabled, then the input signal is always delayed 3 rising edges of the system clock (two rising edges to the synchronizer plus one more rising edge to the edge detector). In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input pin .

#### NOTE

Input capture mode is only available when (FTMEN = 0) and (FTMxCNTINH:FTMxCNTINL = 0x0000). Input capture mode with (FTMEN = 1) or (FTMxCNTINH:FTMxCNTINL not = 0x0000) is not recommended and its results are not guaranteed.

#### 12.5.4.1 Filter for input capture mode

The filter function is only available on channels 0, 1, 2 and 3.

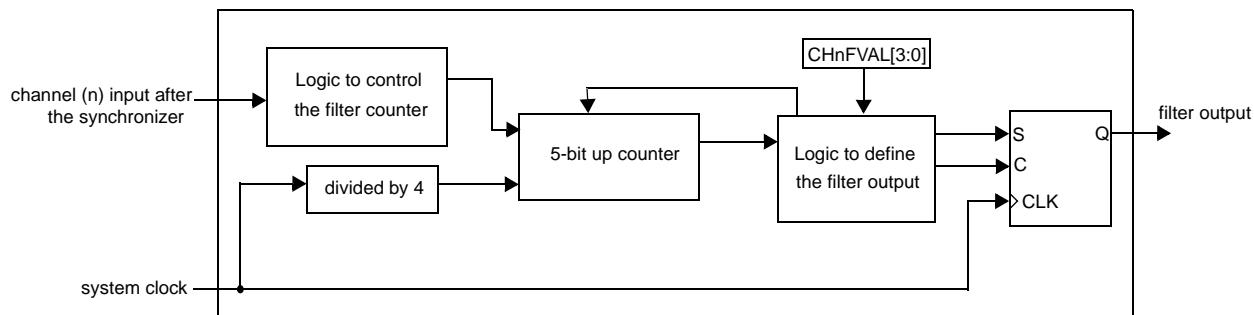
Firstly the input signal is synchronized by the system clock (synchronizer block in [Figure 12-34](#)). Following synchronization, the input signal enters the filter block ([Figure 12-35](#)). When there is a state change in the input signal, the 5-bit counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. If the 5-bit counter overflows (the counter exceeds the value of the CHnFVAL[3:0] bits), the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the input signal before validation (counter overflow), the counter is reset. At the next input transition, the counter will start counting again. Any pulse that is shorter than the minimum value selected by CHnFVAL[3:0] bits (x 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in [Figure 12-36](#).

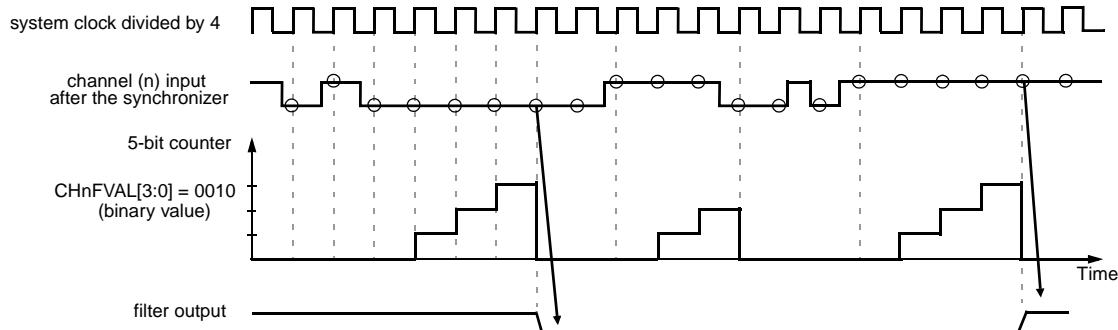
The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case the input signal is delayed 3 rising edges of the system clock. If (CHnFVAL[3:0] not = 0000) then the input signal is delayed by the

minimum pulse width ( $\text{CHnFVAL}[3:0] \times 4$  system clocks) plus a further 4 rising edges of the system clock (two rising edges to the synchronizer, one rising edge to the filter output plus one more to the edge detector). In other words  $\text{CHnF}$  is set ( $(4 + 4 \times \text{CHnFVAL}[3:0])$  system clock periods after a valid edge occurs on the channel input pin.

The clock for the 5-bit counter in the channel input filter is the system clock divided by 4.



**Figure 12-35. Channel Input Filter**



**Figure 12-36. Channel input filter example**

## 12.5.5 Output compare mode

The output compare mode is selected when:

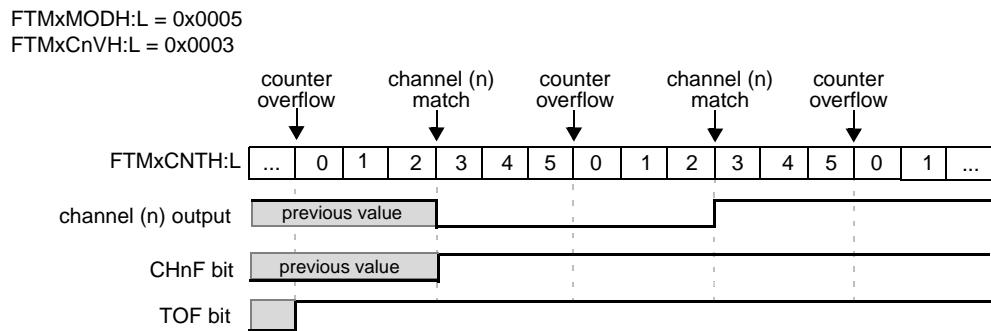
- ( $\text{FTMEN} = 0$ ) and ( $\text{COMBINE} = 0$ ) and ( $\text{CPWMS} = 0$ ) and ( $\text{MSnB:MSnA} = 0:1$ )

In output compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the  $\text{FTMxCnVH:FTMxCnVL}$  registers of an output compare channel, the channel (n) output can be set, cleared or toggled.

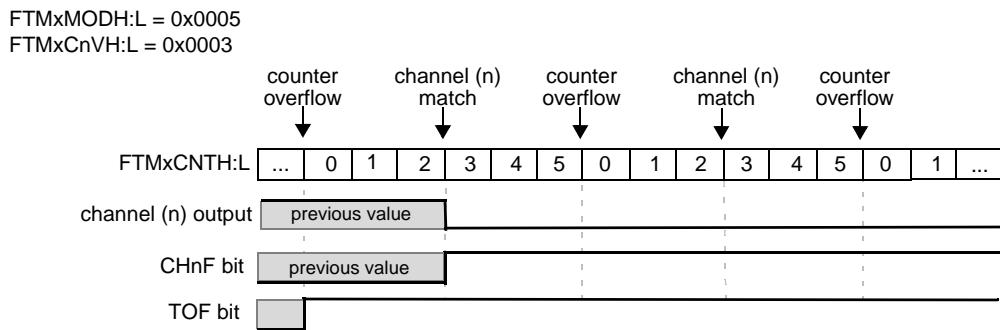
When a channel is initially configured to toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The TOF bit is set and the timer overflow interrupt is generated (if  $\text{TOIE} = 1$ ) at the end of the PWM period (when the FTM counter changes from  $\text{FTMxMODH:FTMxMODL}$  to  $\text{FTMxCNTINH:FTMxCNTINL}$ ).

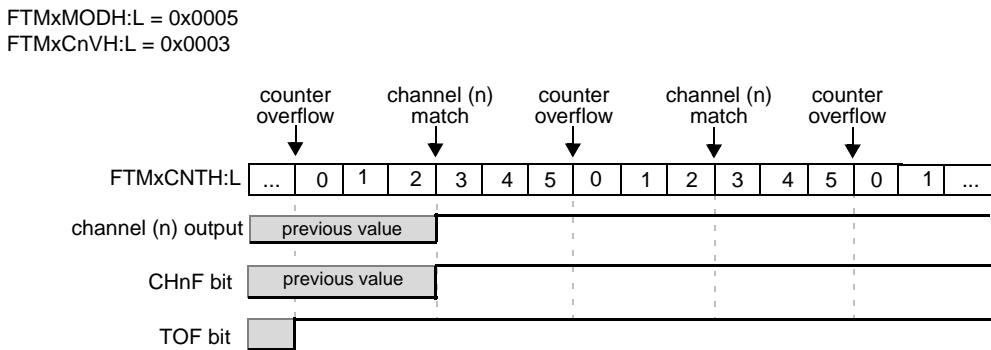
The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (when the FTM counter = FTMxCnVH:FTMxCnVL).



**Figure 12-37. Example of the output compare mode when the match toggles the channel output**



**Figure 12-38. Example of the output compare mode when the match clears the channel output**



**Figure 12-39. Example of the output compare mode when the match sets the channel output**

It is possible to use the output compare mode with (ELSnB:ELSnA = 0:0). In this case, when the counter reaches the value in the FTMxCnVH:FTMxCnVL registers the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not modified and controlled by FTM.

## NOTE

Output compare mode is only available when (FTMEN = 0) and (FTMxCNTINH:FTMxCNTINL = 0x0000). Output compare mode with (FTMEN = 1) or (FTMxCNTINH:FTMxCNTINL not = 0x0000) is not recommended and its results are not guaranteed.

### 12.5.6 Edge-aligned PWM (EPWM) mode

The edge-aligned mode is selected when:

- (FTMEN = 0) and (COMBINE = 0) and (CPWMS = 0) and (MSnB = 1)

The EPWM period is determined by (FTMxMODH:FTMxMODL - FTMxCNTINH:FTMxCNTINL + 0x0001) and the pulse width (duty cycle) is determined by (FTMxCnVH:FTMxCnVL - FTMxCNTINH:FTMxCNTINL).

The TOF bit is set and the timer overflow interrupt is generated (if TOIE = 1) at the end of the EPWM period (when the FTM counter changes from FTMxMODH:FTMxMODL to FTMxCNTINH:FTMxCNTINL). The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the end of the pulse width, that is, at the channel (n) match (when the FTM counter = FTMxCnVH:FTMxCnVL).

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 an FTM.

EPWM mode can be used when other channels in the same FTM are configured for input capture, output compare or combine modes.

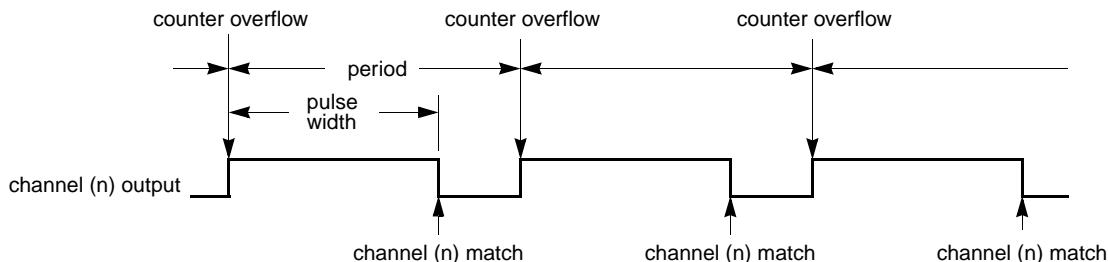
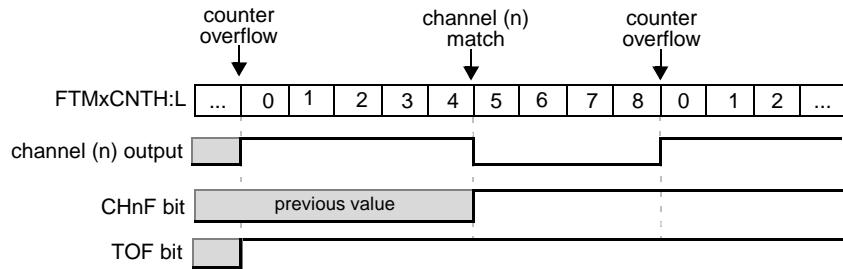


Figure 12-40. EPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the FTMxCnVH:FTMxCnVL registers the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0) then the channel (n) output is forced high at the counter overflow (when the FTMxCNTINH:FTMxCNTINL register contents are loaded into the FTM counter), and it is forced low at the channel (n) match (when the FTM counter = FTMxCnVH:FTMxCnVL) (Figure 12-41).

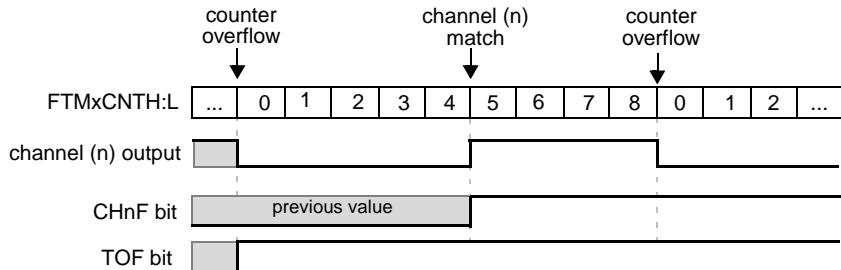
FTMxMODH:L = 0x0008  
FTMxCnVH:L = 0x0005



**Figure 12-41. EPWM signal with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = X:1) then the channel (n) output is forced low at the counter overflow (when the FTMxCNTINH:FTMxCNTINL register contents are loaded into the FTM counter), and it is forced high at the channel (n) match (when the FTM counter = FTMxCnVH:FTMxCnVL) (Figure 12-42).

FTMxMODH:L = 0x0008  
FTMxCnVH:L = 0x0005



**Figure 12-42. EPWM signal with ELSnB:ELSnA = X:1**

If (FTMxCnVH:FTMxCnVL = 0x0000) then the channel (n) output is a 0% duty cycle EPWM signal. If (FTMxCnVH:FTMxCnVL > FTMxMODH:FTMxMODL) then the channel (n) output is a 100% duty cycle EPWM signal. This implies that the modulus setting must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

#### NOTE

EPWM mode is only available when (FTMEN = 0) and (FTMxCNTINH:FTMxCNTINL = 0x0000). EPWM mode with (FTMEN = 1) or (FTMxCNTINH:FTMxCNTINL not = 0x0000) is not recommended and its results are not guaranteed.

### 12.5.7 Center-aligned PWM (CPWM) mode

The center-aligned mode is selected when:

- (FTMEN = 0) and (COMBINE = 0) and (CPWMS = 1)

The CPWM pulse width (duty cycle) is determined by 2 x (FTMxCnVH:FTMxCnVL - FTMxCNTINH:FTMxCNTINL) and the period is determined by 2 x (FTMxMODH:FTMxMODL -

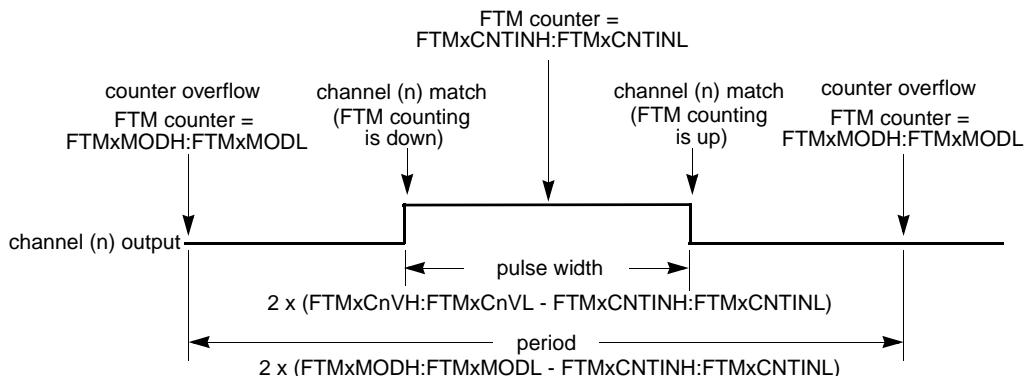
FTMxCNTINH:FTMxCNTINL)([Figure 12-43](#)). FTMxMODH:FTMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode the FTM counter counts up until it reaches FTMxMODH:FTMxMODL and then counts down until it reaches FTMxCNTINH:FTMxCNTINL.

The TOF bit is set and the timer overflow interrupt is generated (if TOIE = 1) at the end of the CPWM period (when the FTM counter changes from FTMxMODH:FTMxMODL to FTMxMODH:FTMxMODL - 0x0001). The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the begin of the pulse width (when there is a channel (n) match while the FTM counting is down) and at the end of the pulse width (when there is a channel (n) match while the FTM counting is up).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of FTMxCNTINH:FTMxCNTINL.

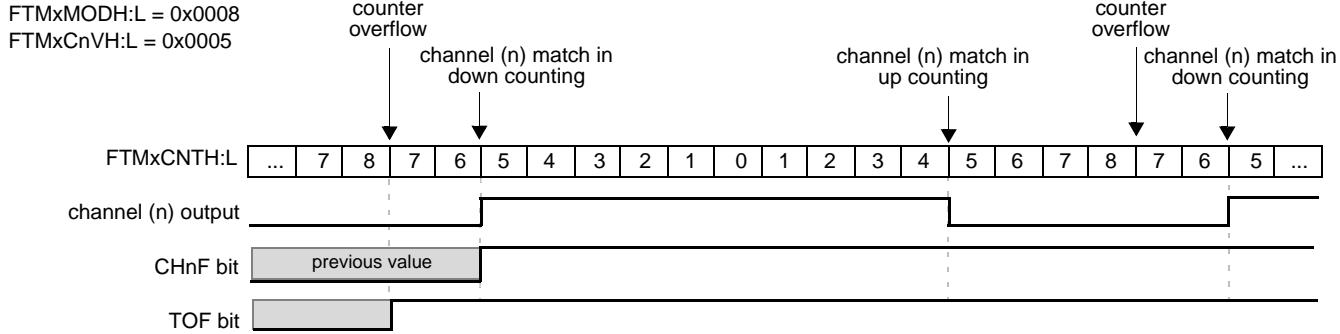
Input capture, output compare, EPWM and combine modes are not compatible with a counter operating in up/down counting mode (CPWMS = 1). Therefore all active channels within an FTM must be used in CPWM mode.



**Figure 12-43. CPWM period and pulse width with ELSnB:ELSnA = 1:0**

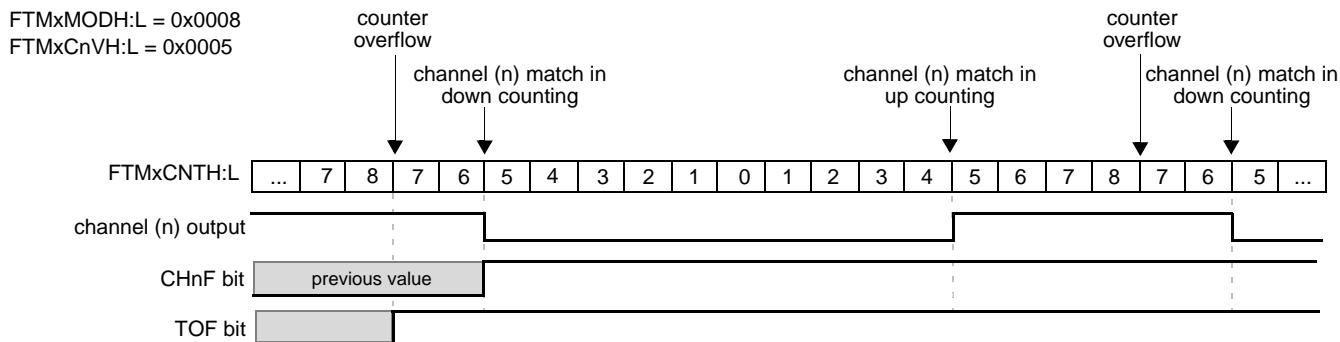
If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the FTMxCnVH:FTMxCnVL registers the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0) then the channel (n) output is forced high at the channel (n) match (FTM counter = FTMxCnVH:FTMxCnVL) when counting down, and it is forced low at the channel (n) match when counting up ([Figure 12-44](#)).



**Figure 12-44. CPWM signal with ELSnB:ELSnA = 1:0**

If (ELSnB:ELSnA = X:1) then the channel (n) output is forced low at the channel (n) match (FTM counter = FTMxCnVH:FTMxCnVL) when counting down, and it is forced high at the channel (n) match when counting up (Figure 12-45).



**Figure 12-45. CPWM signal with ELSnB:ELSnA = X:1**

If (FTMxCnVH:FTMxCnVL = 0x0000) or (FTMxCnVH:FTMxCnVL is a negative value, that is, FTMxCnVH[7] = 1) then the channel (n) output is a 0% duty cycle CPWM signal.

If (FTMxCnVH:FTMxCnVL is a positive value, that is, FTMxCnVH[7] = 0) and (FTMxCnVH:FTMxCnVL >= FTMxMODH:FTMxMODL) and (FTMxMODH:FTMxMODL not = 0x0000) then the channel (n) output is a 100% duty cycle CPWM signal. This implies that the usable range of periods set by FTMxMODH:FTMxMODL is 0x0001 through 0x7FF (0x7FFF if you do not need to generate a 100% duty cycle CPWM signal). This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode should not be used when the FTM counter is a free running counter.

#### NOTE

CPWM mode is only available when (FTMEN = 0) and (FTMxCNTINH:FTMxCNTINL = 0x0000). CPWM mode with (FTMEN = 1) or (FTMxCNTINH:FTMxCNTINL not = 0x0000) is not recommended and its results are not guaranteed.

## 12.5.8 Combine mode

The combine mode is selected when:

- (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0)

In combine mode the channel (n) (an even channel) and channel (n+1) (the adjacent odd channel) are combined to generate a PWM signal in the channel (n) output.

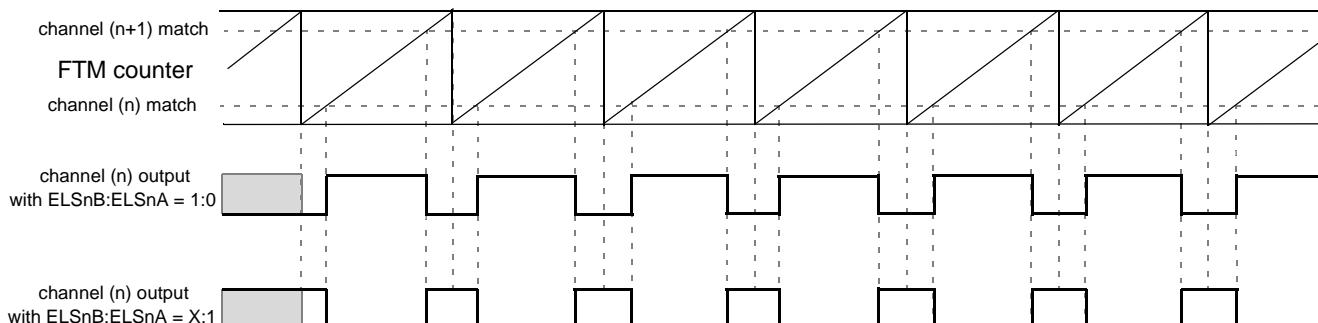
In the combine mode the PWM period is determined by (FTMxMODH:FTMxMODL - FTMxCNTINH:FTMxCNTINL + 0x0001) and the PWM pulse width (duty cycle) is determined by (|FTMxC(n+1)VH:FTMxC(n+1)VL - FTMxC(n)VH:FTMxC(n)VL|).

The TOF bit is set and the timer overflow interrupt is generated (if TOIE = 1) at the end of the PWM period (when the FTM counter changes from FTMxMODH:FTMxMODL to FTMxCNTINH:FTMxCNTINL). The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = FTMxC(n)VH:FTMxC(n)VL). The CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1) at the channel (n+1) match (FTM counter = FTMxC(n+1)VH:FTMxC(n+1)VL).

If (ELSnB:ELSnA = 1:0) then the channel (n) output is forced low at the beginning of the period (FTM counter = FTMxCNTINH:FTMxCNTINL) and at the channel (n+1) match (FTM counter = FTMxC(n+1)VH:FTMxC(n+1)VL). It is forced high at the channel (n) match (FTM counter = FTMxC(n)VH:FTMxC(n)VL)([Figure 12-46](#)).

If (ELSnB:ELSnA = X:1) then the channel (n) output is forced high at the beginning of the period (FTM counter = FTMxCNTINH:FTMxCNTINL) and at the channel (n+1) match (FTM counter = FTMxC(n+1)VH:FTMxC(n+1)VL). It is forced low at the channel (n) match (FTM counter = FTMxC(n)VH:FTMxC(n)VL)([Figure 12-46](#)).

In combine mode the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channel (n) and (n+1) output.

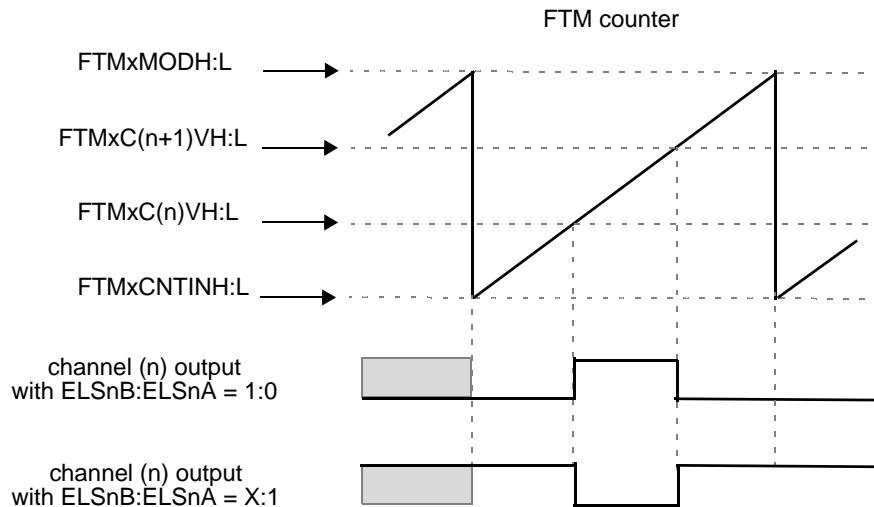


**Figure 12-46. Combine mode**

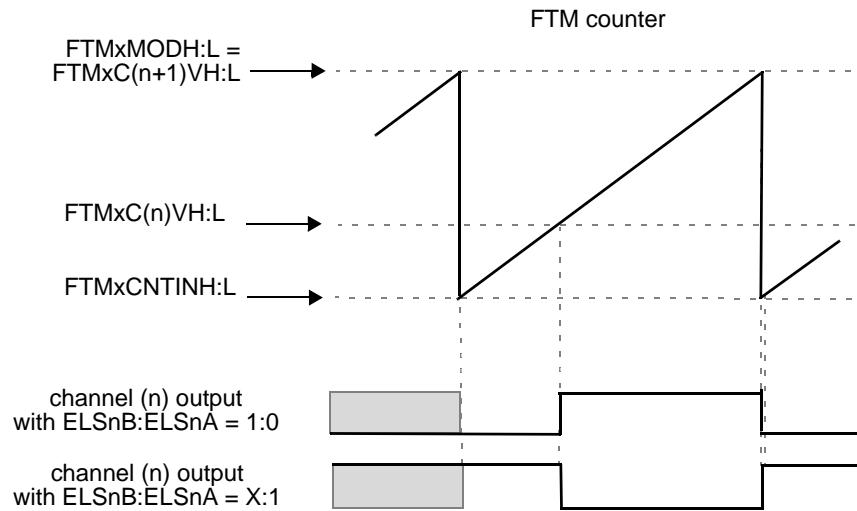
## NOTE

Combine mode is only available when (FTMEN = 1) and (CPWMS = 0). Combine mode with (FTMEN = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

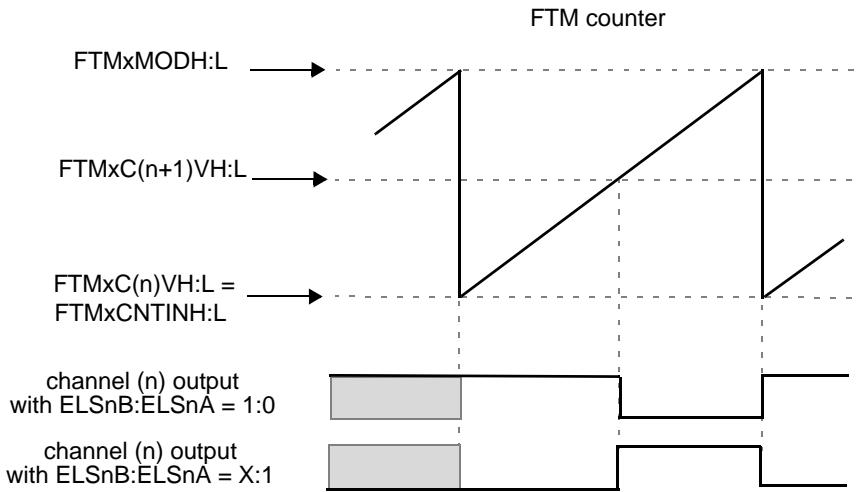
The following figures illustrate the generation of signals using combine mode.



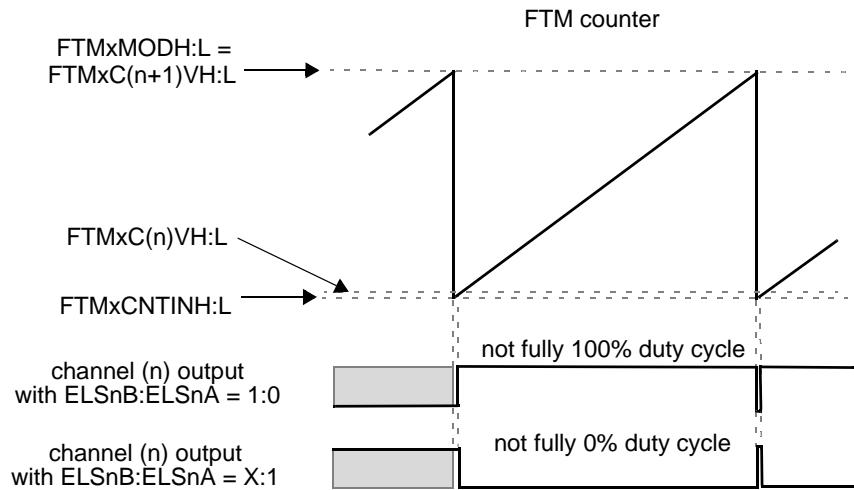
**Figure 12-47. Channel (n) output if ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n+1)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxC}(n)\text{VH:L} < \text{FTMxC}(n+1)\text{VH:L}$ )**



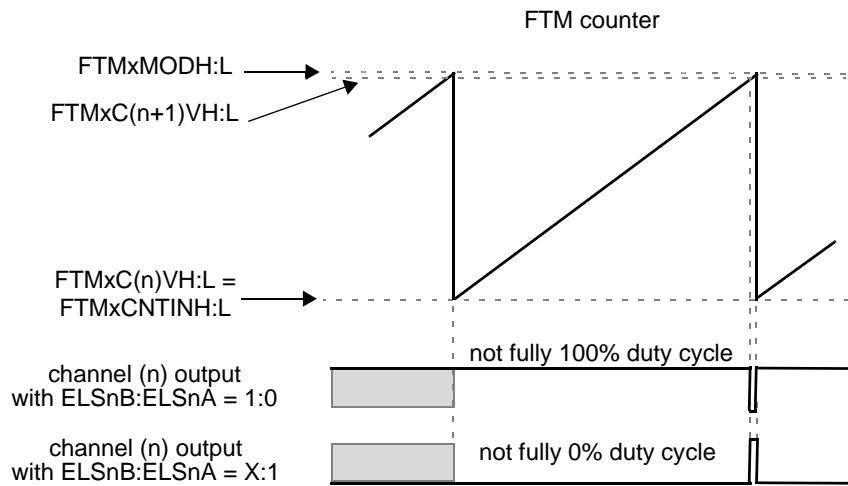
**Figure 12-48. Channel (n) output if ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxC}(n+1)\text{VH:L} = \text{FTMxMODH:L}$ )**



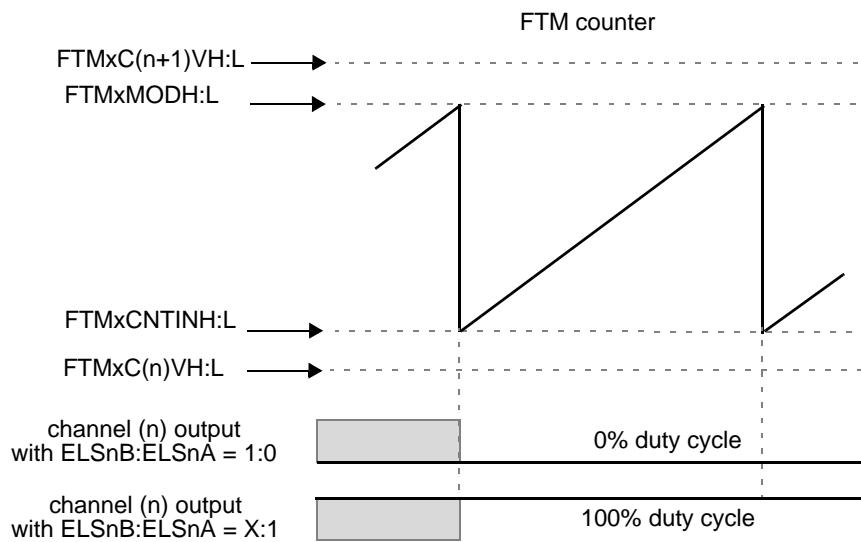
**Figure 12-49. Channel (n) output if ( $\text{FTMxC(n)VH:L} = \text{FTMxCNTINH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC(n+1)VH:L} < \text{FTMxMODH:L}$ )**



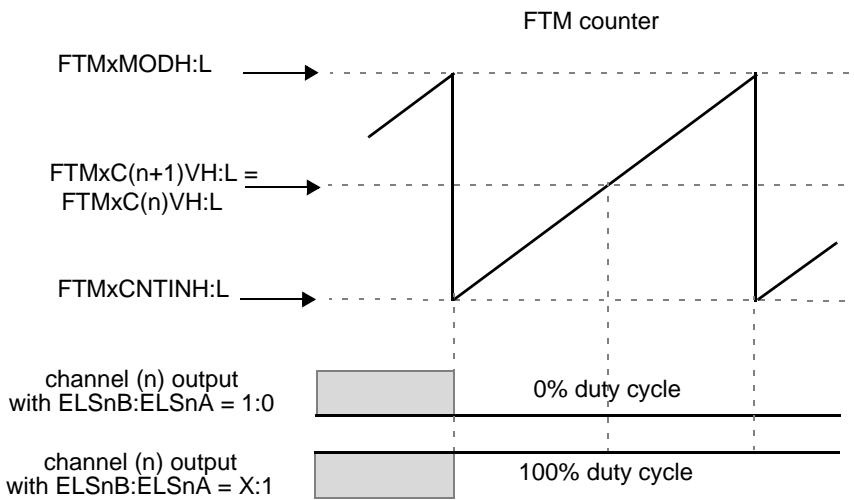
**Figure 12-50. Channel (n) output if ( $\text{FTMxCNTINH:L} < \text{FTMxC(n)VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxC(n)VH:L}$  is almost equal to  $\text{FTMxCNTINH:L}$ ) and ( $\text{FTMxC(n+1)VH:L} = \text{FTMxMODH:L}$ )**



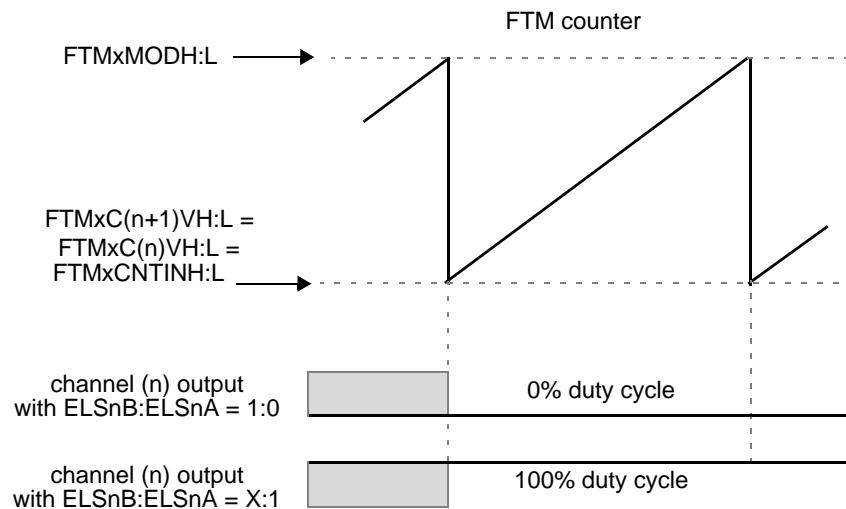
**Figure 12-51. Channel (n) output if  $(FTMxC(n)VH:L = FTMxCNTINH:L)$  and  $(FTMxCNTINH:L < FTMxC(n+1)VH:L < FTMxMODH:L)$  and  $(FTMxC(n+1)VH:L$  is almost equal to  $FTMxMODH:L)$**



**Figure 12-52. Channel (n) output if  $FTMxC(n)VH:L$  and  $FTMxC(n+1)VH:L$  are not between  $FTMxCNTINH:L$  and  $FTMxMODH:L$**



**Figure 12-53. Channel (n) output if ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n+1)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxCnVH:L} = \text{FTMxC}(n+1)\text{VH:L}$ )**



**Figure 12-54. Channel (n) output if ( $\text{FTMxC}(n)\text{VH:L} = \text{FTMxC}(n+1)\text{VH:L} = \text{FTMxCNTINH:L}$ )**

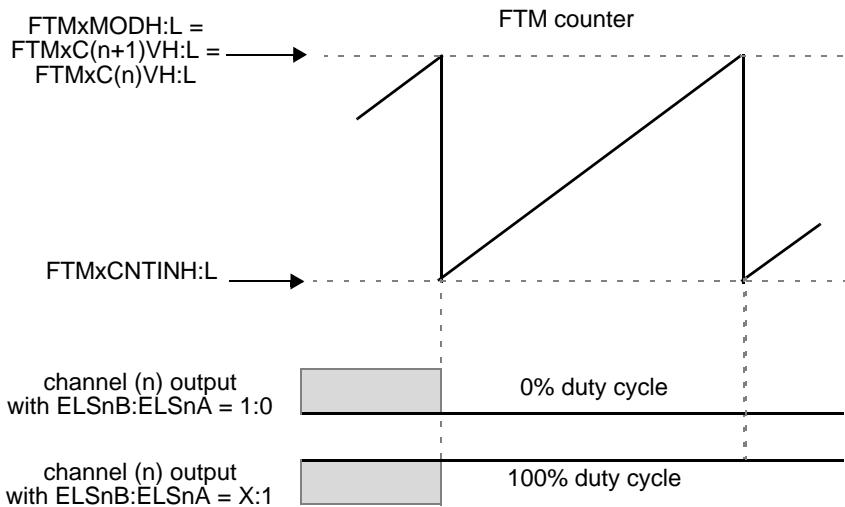


Figure 12-55. Channel (n) output if ( $\text{FTMxC}(n)\text{VH:L} = \text{FTMxC}(n+1)\text{VH:L} = \text{FTMxMODH:L}$ )

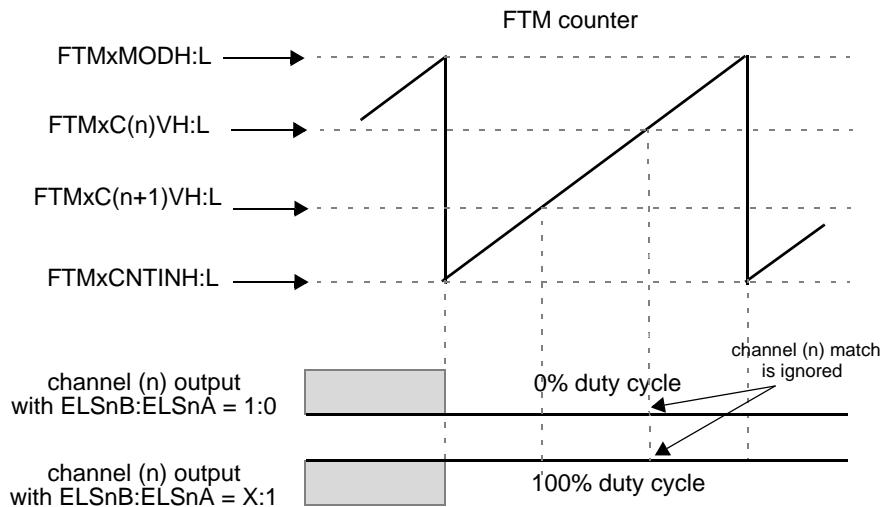
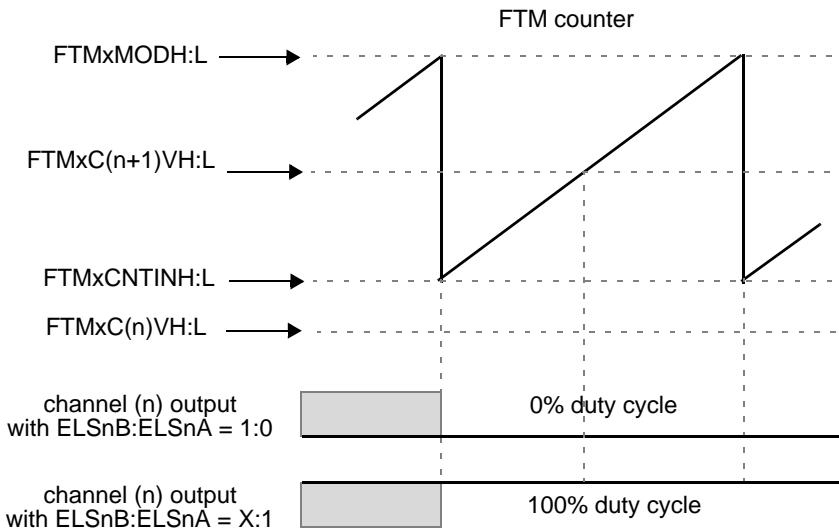
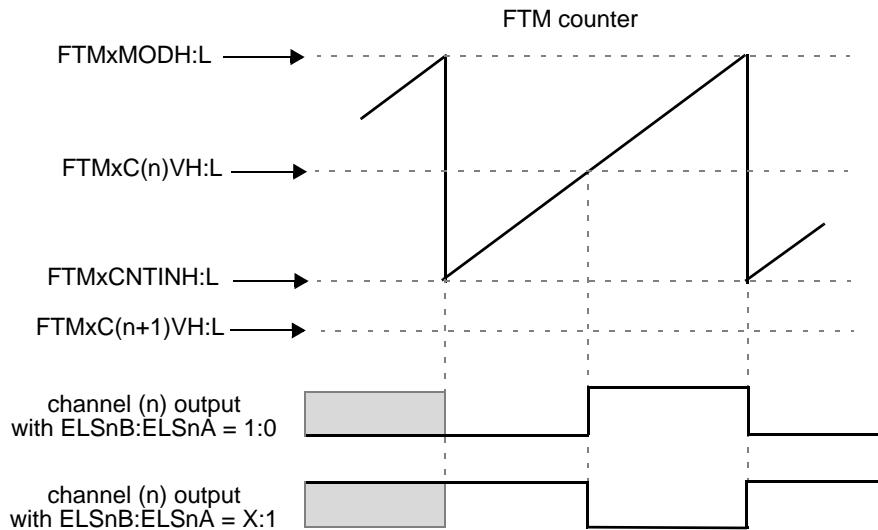


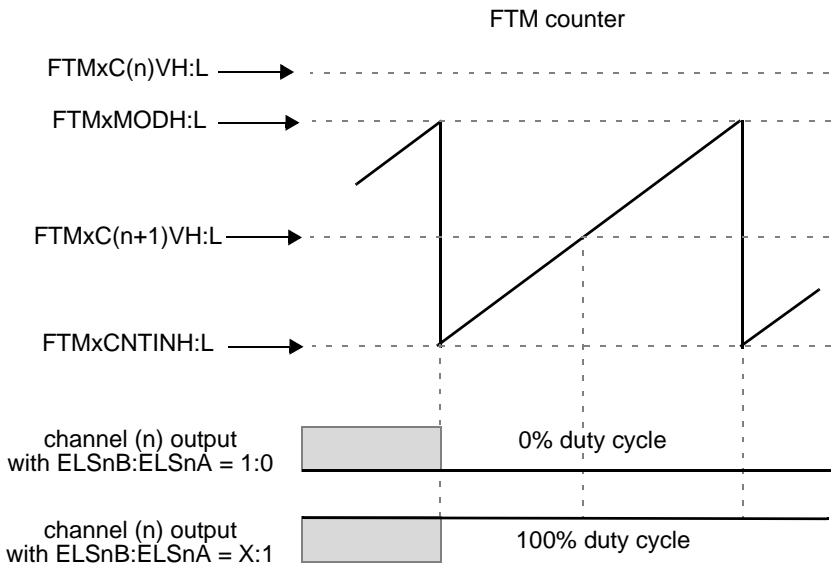
Figure 12-56. Channel (n) output if ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n+1)\text{VH:L} < \text{FTMxMODH:L}$ ) and ( $\text{FTMxC}(n)\text{VH:L} > \text{FTMxC}(n+1)\text{VH:L}$ )



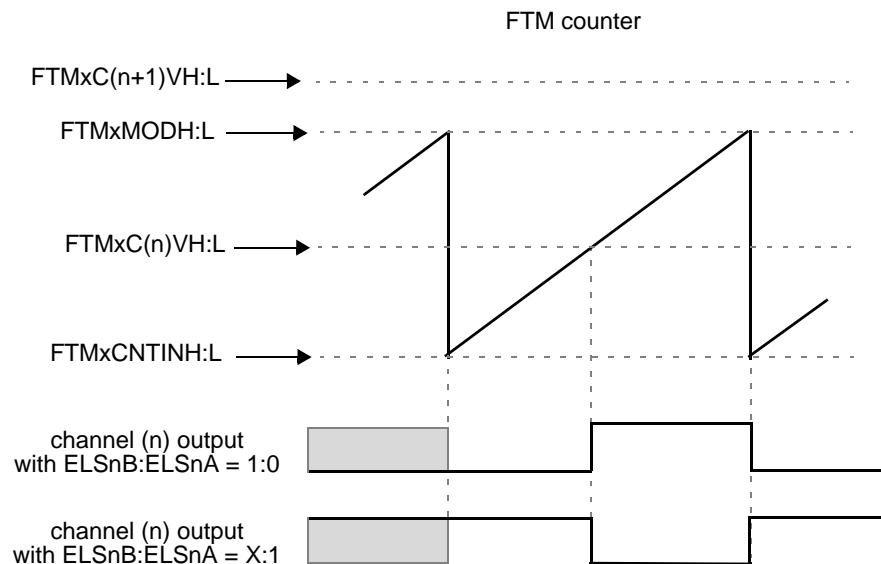
**Figure 12-57. Channel (n) output if ( $\text{FTMxC}(n)\text{VH:L} < \text{FTMxCNTINH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n+1)\text{VH:L} < \text{FTMxMODH:L}$ )**



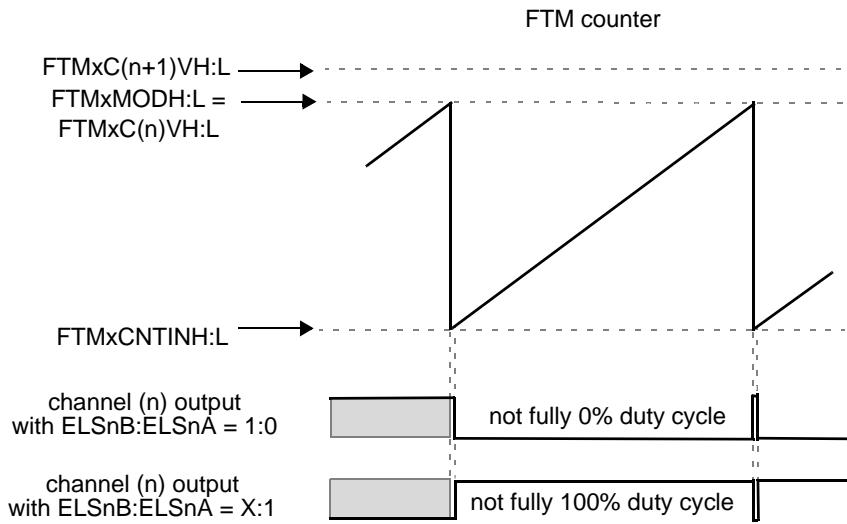
**Figure 12-58. Channel (n) output if ( $\text{FTMxC}(n+1)\text{VH:L} < \text{FTMxCNTINH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} < \text{FTMxMODH:L}$ )**



**Figure 12-59. Channel (n) output if ( $\text{FTMxC}(n)\text{VH:L} > \text{FTMxMODH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n+1)\text{VH:L} < \text{FTMxMODH:L}$ )**



**Figure 12-60. Channel (n) output if ( $\text{FTMxC}(n+1)\text{VH:L} > \text{FTMxMODH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} < \text{FTMxMODH:L}$ )**



**Figure 12-61. Channel (n) output if ( $\text{FTMxC}(n+1)\text{VH:L} > \text{FTMxMODH:L}$ ) and ( $\text{FTMxCNTINH:L} < \text{FTMxC}(n)\text{VH:L} = \text{FTMxMODH:L}$ )**

### 12.5.8.1 Asymmetrical PWM

The combine mode can be used to generate an asymmetrical PWM signal when  $\text{FTMxC}(n)\text{VH:L}$  is different from  $\text{FTMxC}(n+1)\text{VH:L}$ .

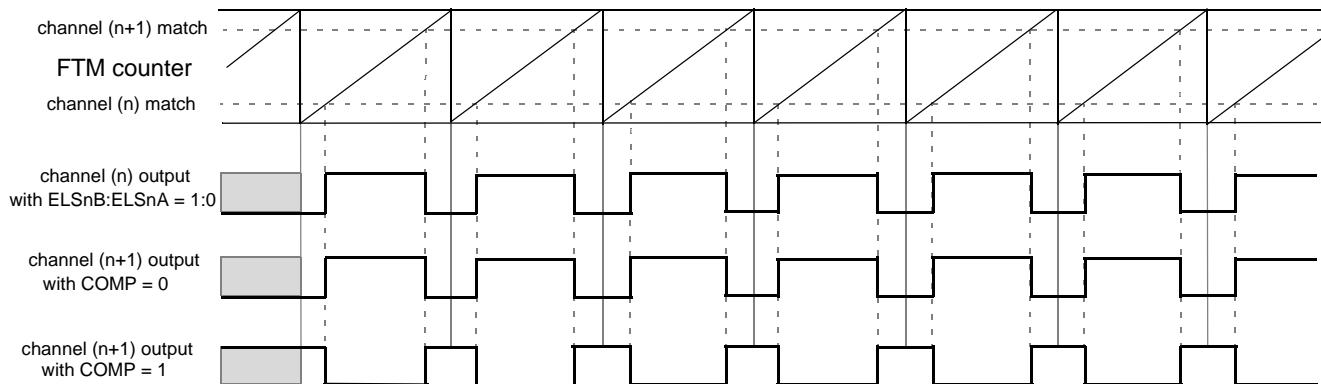
### 12.5.9 Complementary mode

The complementary mode is selected when:

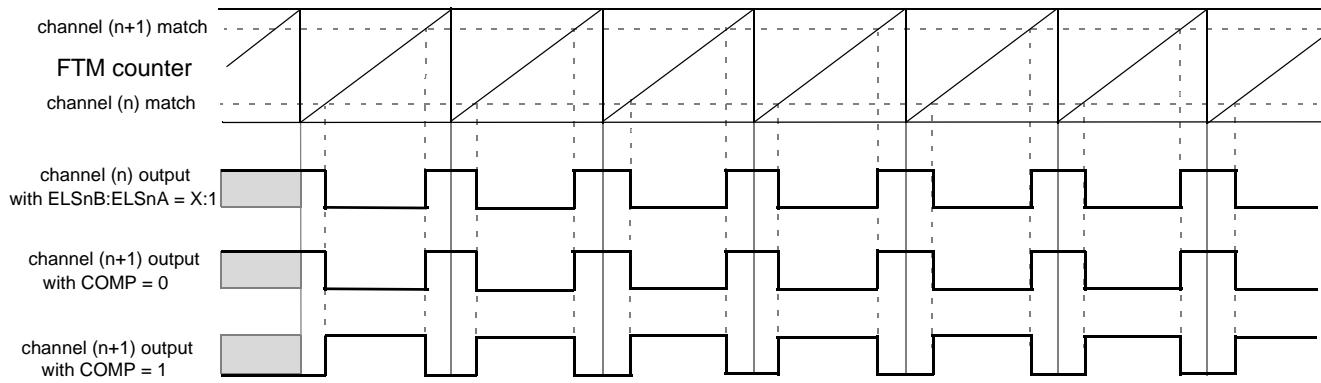
- ( $\text{FTMEN} = 1$ ) and ( $\text{COMBINE} = 1$ ) and ( $\text{CPWMS} = 0$ ) and ( $\text{COMP} = 1$ )

In complementary mode the channel (n+1) output is the inverse of the channel (n) output.

If ( $\text{FTMEN} = 1$ ) and ( $\text{COMBINE} = 1$ ) and ( $\text{CPWMS} = 0$ ) and ( $\text{COMP} = 0$ ), then the channel (n+1) output is the same as the channel (n) output.



**Figure 12-62. Channel (n+1) output in complementary mode with (ELSnB:ELSnA = 1:0)**



**Figure 12-63. Channel (n+1) output in complementary mode with (ELSnB:ELSnA = X:1)**

#### NOTE

Complementary mode is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). Complementary mode with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

#### 12.5.10 Load of the registers with write buffers

FTMxCNTINH:L registers are always updated with their write buffer after both bytes have been written.

If (CLKS[1:0] = 00) then FTMxMODH:L registers are updated when their second byte is written (independent of FTMEN bit). If (CLKS[1:0] not = 00) then FTMxMODH:L registers are updated with their write buffer according to the FTMEN bit. If (CLKS[1:0] not = 00 and FTMEN = 1) then FTMxMODH:L registers are updated by PWM synchronization ([Section 12.5.11, “PWM synchronization”](#)). If (CLKS[1:0] not = 00 and FTMEN = 0) then FTMxMODH:L registers are updated according to the CPWMS bit, that is:

- If (selected mode is not CPWM mode) then FTMxMODH:L registers are updated after both bytes have been written and the FTM counter changes from (FTMxMODH:L) to (FTMxCNTINH:L). If the FTM counter is a free-running counter then this update is made when the FTM counter changes from 0xFFFF to 0x0000.
- If (selected mode is CPWM mode) then FTMxMODH:L registers are updated after both bytes have been written and the FTM counter changes from (FTMxMODH:L) to (FTMxMODH:L - 0x0001).

If (CLKS[1:0] = 00) then FTMxCnVH:L registers are updated when their second byte is written (independent of FTMEN bit). If (CLKS[1:0] not = 00) then FTMxCnVH:L registers are updated with their write buffer according to the FTMEN bit. If (CLKS[1:0] not = 00 and FTMEN = 1) then FTMxCnVH:L registers are updated by PWM synchronization ([Section 12.5.11, “PWM synchronization”](#)). If (CLKS[1:0] not = 00 and FTMEN = 0) then FTMxCnVH:L registers are updated according to the selected mode, that is:

- If (selected mode is output compare mode), then FTMxCnVH:L registers are updated after their second byte is written and on the next change of the FTM counter (end of the prescaler counting).
- If (selected mode is EPWM mode), then FTMxCnVH:L registers are updated after both bytes have been written and the FTM counter changes from (FTMxMODH:L) to (FTMxCNTINH:L). If the FTM counter is a free-running counter then this update is made when the FTM counter changes from 0xFFFF to 0x0000.
- If (selected mode is CPWM mode), then FTMxCnVH:L registers are updated after both bytes have been written and the FTM counter changes from (FTMxMODH:L) to (FTMxMODH:L - 0x0001).

## 12.5.11 PWM synchronization

PWM synchronization provides an opportunity to update registers with the contents of their write buffers. It can also be used to synchronize two or more FlexTimer modules on the same MCU.

PWM synchronization updates the FTMxMODH:L and FTMxCnVH:L registers with their write buffers. It is also possible to force the FTM counter to its initial value and update the CHnOM bits in FTMxOUTMASK using PWM synchronization.

### NOTE

PWM synchronization is only available when (COMBINE = 1) and (CPWMS = 0). PWM synchronization with (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

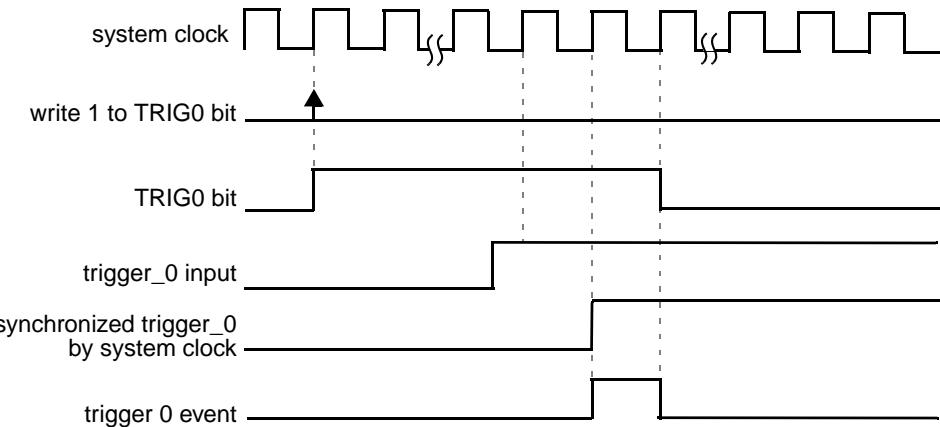
### 12.5.11.1 Hardware trigger

Each hardware trigger (input signals: trigger\_0, trigger\_1 and trigger\_2) is synchronized by the system clock.

A rising edge on the selected hardware trigger input (trigger n event) initiates PWM synchronization. A hardware trigger is selected when its enable bit is set (TRIGn = 1 where n = 0, 1 or 2). The TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

If two or more hardware triggers are enabled (TRIG0 and TRIG1 = 1) and only the trigger 1 event occurs, then only the TRIG1 bit is cleared.

If a trigger n event occurs together with a write to set the TRIGn bit, then the synchronization is made, but the TRIGn bit remains set because of the last write.



**Note**

All hardware trigger (input signals: trigger\_0, trigger\_1 and trigger\_2) have this same behavior

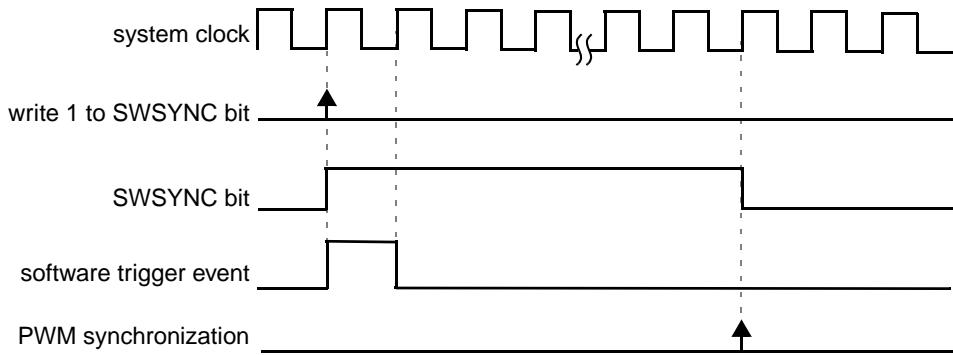
**Figure 12-64. Hardware trigger event**

### 12.5.11.2 Software trigger

A software trigger event occurs when 1 is written to the SWSYNC bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization (initiated by the software event) is complete.

If the software trigger event occurs together with the event that clears the SWSYNC bit, then the synchronization is made using this trigger event and the SWSYNC bit remains set because of the last write.

For example, if PWMSYNC = 0 and REINIT = 0 and there is a software trigger event, then the load of FTmxMODH:L and FTmxCnVH:L registers is only made at the boundary cycle (CNTMIN and CNTMAX). In this case, the SWSYNC bit is cleared only at the boundary cycle, so the user does not know when this bit will be cleared. Thus, it is possible a new write to set SWSYNC happens when FTM is clearing the SWSYNC because it is the selected boundary cycle of PWM synchronization that was started previously by the software trigger event.



**Figure 12-65. Software trigger event**

### 12.5.11.3 Boundary cycle

The CNTMAX and CNTMIN bits select the boundary cycle when the FTMxMODH:L and FTMxCnVH:L registers will be updated with the value of their write buffer by PWM synchronization, except if (PWMSYNC = 0 and REINIT = 1).

If (CNTMIN = 1) then the boundary cycle is the FTMxCNTINH:L value. Registers FTMxMODH:L and FTMxCnVH:L are updated when the FTM counter reaches the FTMxCNTINH:L value. If (CPWMS = 0) then FTMxCNTINH:L is reached when the FTM counter changes from FTMxMODH:L to FTMxCNTINH:L. If (CPWMS = 1) then FTMxCNTINH:L is reached when the FTM counter changes from (FTMxCNTINH:L + 0x0001) to FTMxCNTINH:L.

If (CNTMAX = 1) then the boundary cycle is the FTMxMODH:L value. Registers FTMxMODH:L and FTMxCnVH:L are updated when the FTM counter reaches the FTMxMODH:L value. FTMxMODH:L is reached when the FTM counter changes from (FTMxMODH:L - 0x0001) to FTMxMODH:L, regardless of the CPWMS configuration.

If no boundary cycle was selected (that is, CNTMAX = 0 and CNTMIN = 0), then the update of the FTMxMODH:L and FTMxCnVH:L registers would not be made, except if (PWMSYNC = 0 and REINIT = 1).

If both boundary cycles were selected (that is, CNTMAX = 1 and CNTMIN = 1), then the update of the FTMxMODH:L and FTMxCnVH:L registers would be made in the first boundary cycle that occurs with valid conditions for FTMxMODH:L or FTMxCnVH:L synchronization, except if (PWMSYNC = 0 and REINIT = 1).

The CNTMAX and CNTMIN bits are cleared only by software.

#### NOTE

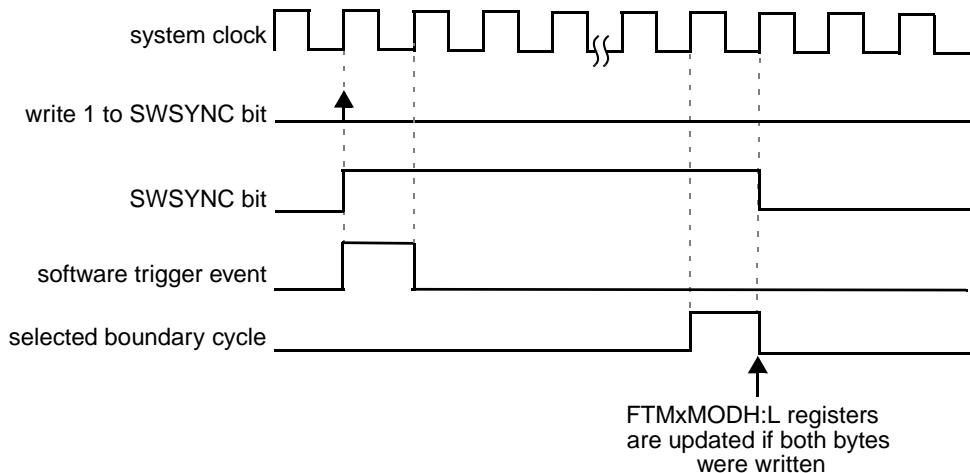
PWM synchronization boundary cycle is only available when (CNTMIN = 1). PWM synchronization with (CNTMAX = 1) is not recommended and its results are not guaranteed.

#### 12.5.11.4 FTMxMODH:FTMxMODL synchronization

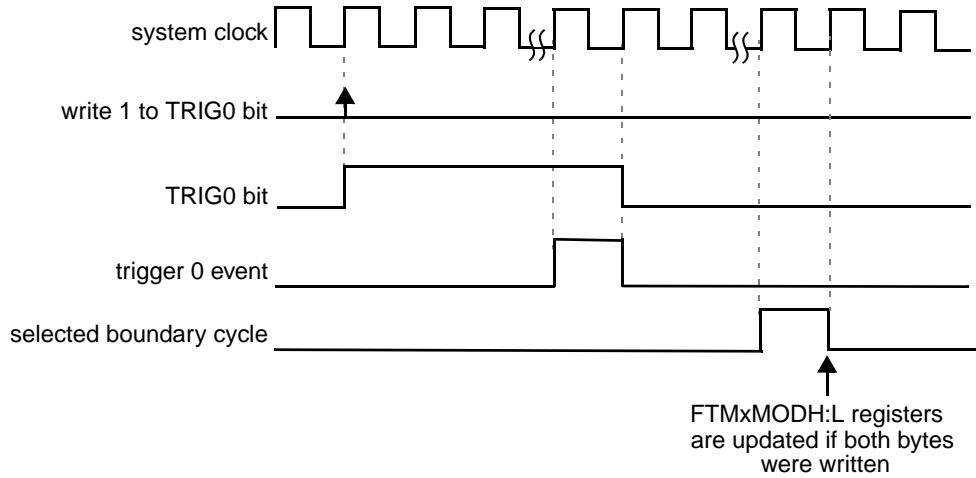
The FTMxMODH:L synchronization occurs when the FTMxMODH:L registers are updated with the value of their write buffer.

The synchronization requires both bytes of FTMxMODH:L to have been written in one of the following situations.

If PWMSYNC = 0 and REINIT = 0, then the synchronization is made on the next selected boundary cycle after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected boundary cycle ([Figure 12-66](#)). If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared when the trigger n event is detected ([Figure 12-67](#)).

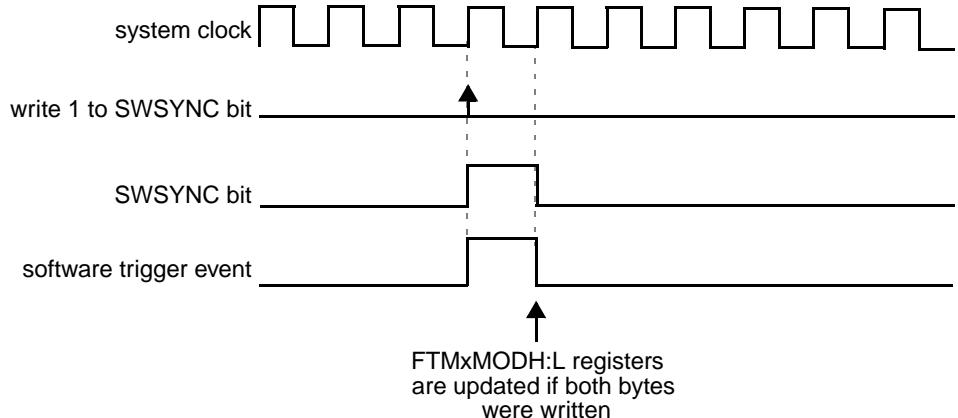


**Figure 12-66. FTMxMODH:L synchronization when (PWMSYNC = 0) and (REINIT = 0) and (software trigger was used)**

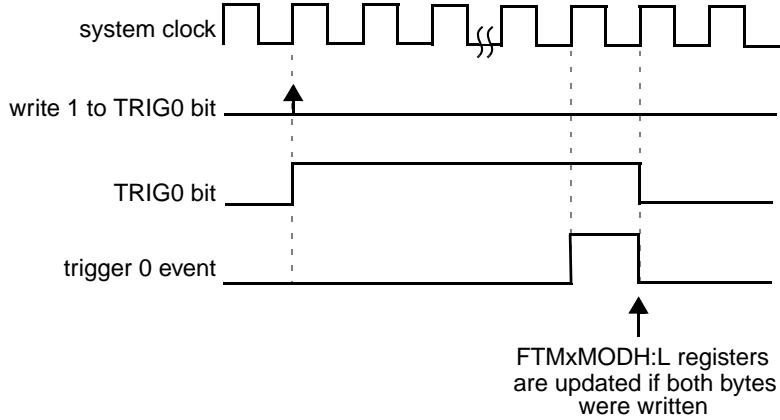


**Figure 12-67. FTMxMODH:L synchronization when (PWMSYNC = 0) and (REINIT = 0) and (a hardware trigger was used)**

If PWMSYNC = 0 and REINIT = 1, then the synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared (Figure 12-68). If the trigger event was a hardware trigger, then the TRIGn bit is cleared (Figure 12-69).

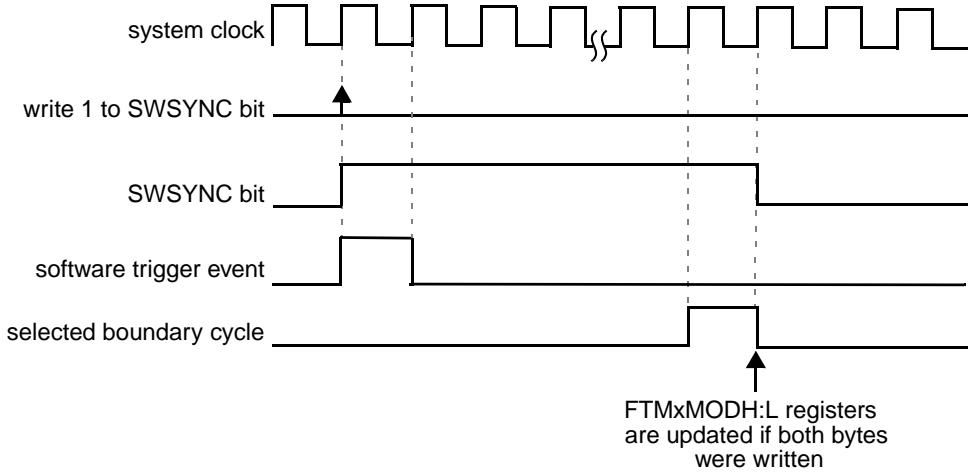


**Figure 12-68. FTMxMODH:L synchronization when (PWMSYNC = 0) and (REINIT = 1) and (software trigger was used)**



**Figure 12-69. FTMxMODH:L synchronization when (PWMSYNC = 0) and (REINIT = 1) and (a hardware trigger was used)**

If PWMSYNC = 1, then the synchronization is made on the next selected boundary cycle after the enabled software trigger event takes place. The SWSYNC bit is cleared on the next selected boundary cycle (Figure 12-70).



**Figure 12-70. FTMxMODH:L synchronization when (PWMSYNC = 1)**

### 12.5.11.5 FTMxCnVH:FTMxCnVL synchronization

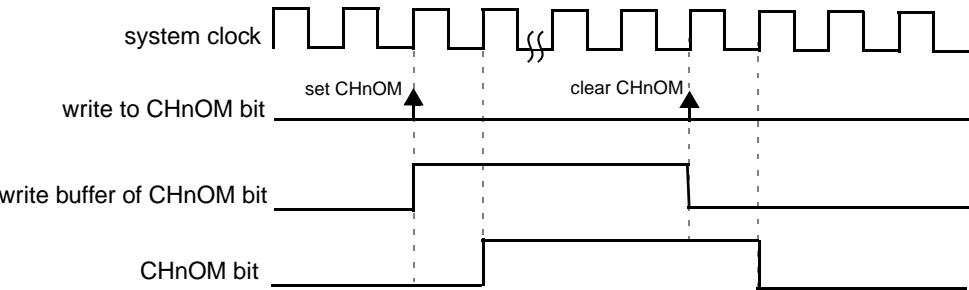
The FTMxCnVH:L synchronization occurs when the FTMxCnVH:L registers are updated with the value of their write buffer.

The synchronization requires both bytes of FTMxCnVH:L to have been written, SYNCEN = 1 and either a hardware or software trigger event as per FTMxMODH:L synchronization (Section 12.5.11.4, “FTMxMODH:FTMxMODL synchronization”).

### 12.5.11.6 CHnOM synchronization

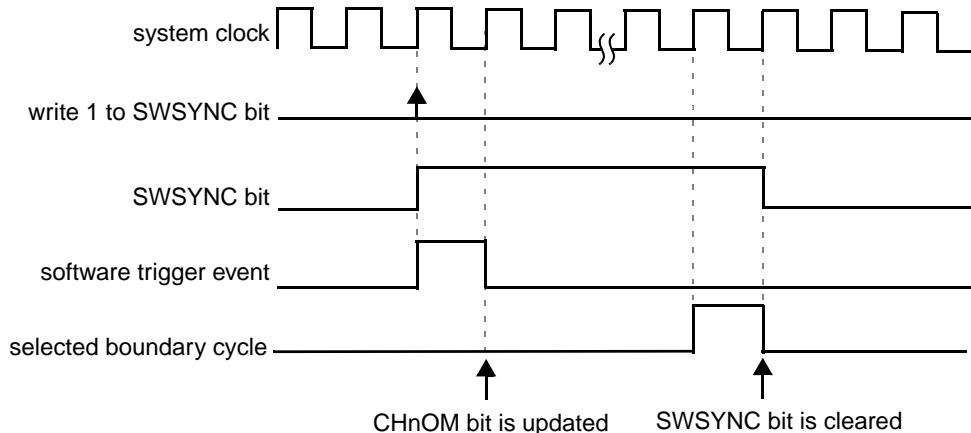
Any write to a CHnOM bit updates the FTMxOUTMASK write buffer. The CHnOM bit is updated with the value of its corresponding bit in the FTMxOUTMASK write buffer according to SYNCHOM and PWMSYNC bits.

If SYNCHOM = 0, then the CHnOM bit is updated with the value of its write buffer equivalent in all rising edges of the system clock.

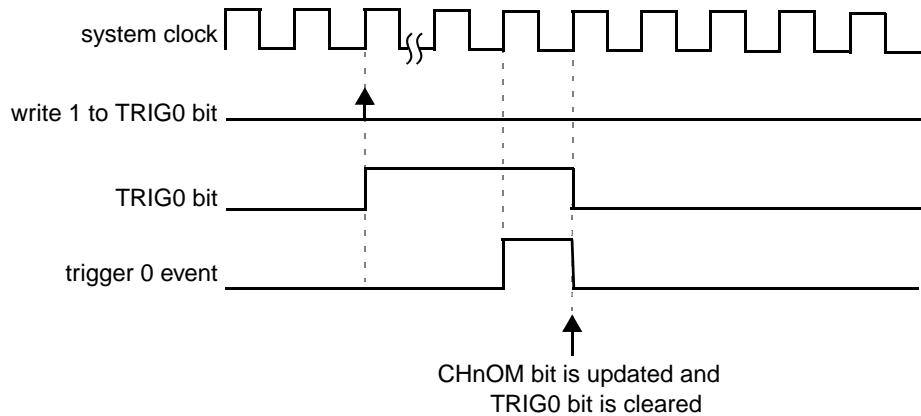


**Figure 12-71. CHnOM synchronization when (SYNCHOM = 0)**

If SYNCHOM = 1 and PWMSYNC = 0, then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected boundary cycle (Figure 12-72). If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared when the trigger n event is detected (Figure 12-73).

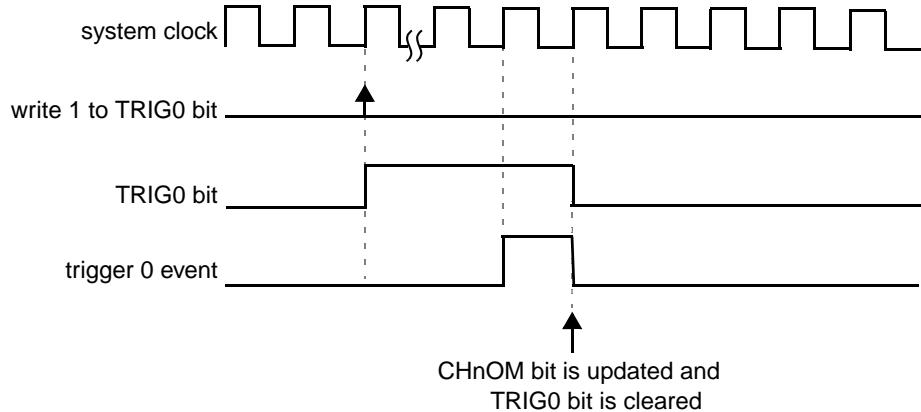


**Figure 12-72. CHnOM synchronization when (SYNCHOM = 1) and (PWMSYNC = 0) and (software trigger was used)**



**Figure 12-73. CHnOM synchronization when (SYNCHOM = 1) and (PWMSYNC = 0) and (a hardware trigger was used)**

If SYNCHOM = 1 and PWMSYNC = 1, then this synchronization is made on the next enabled hardware trigger event. The trigger enable bit (TRIGn) is cleared when the enabled hardware trigger n event is detected ([Figure 12-74](#)).



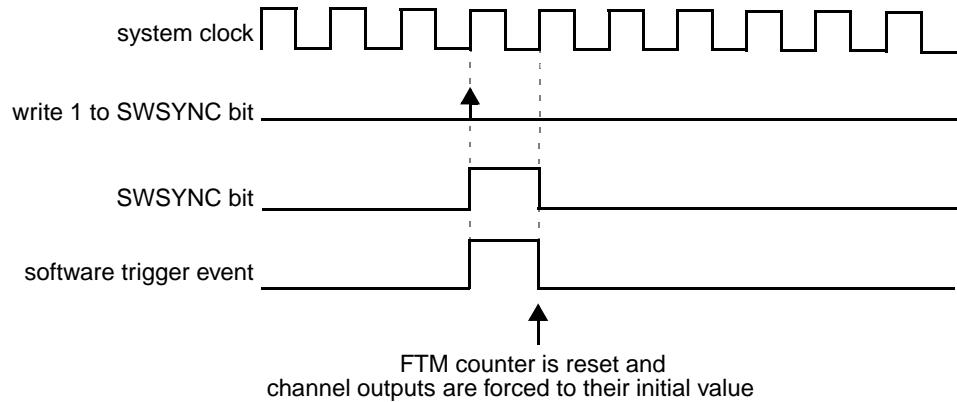
**Figure 12-74. CHnOM synchronization when (SYNCHOM = 1) and (PWMSYNC = 1) and (a hardware trigger was used)**

### 12.5.11.7 FTM counter synchronization

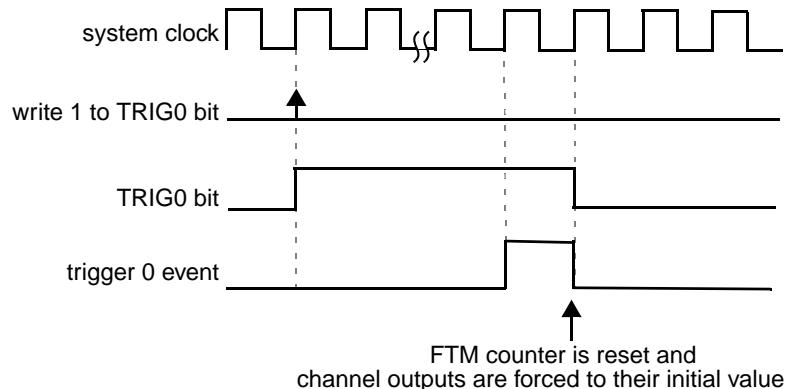
The FTM counter synchronization occurs when the FTM counter is updated with the value of the FTmxCNTINH:L registers and the channel outputs are forced to their initial value as defined by the channel configuration.

If REINIT = 0, then this synchronization is made when the FTM counter changes from FTmxMODH:L to FTmxCNTINH:L.

If REINIT = 1 and PWMSYNC = 0, then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared (Figure 12-75). If the trigger event was a hardware trigger, then the TRIGn bit is cleared (Figure 12-76).

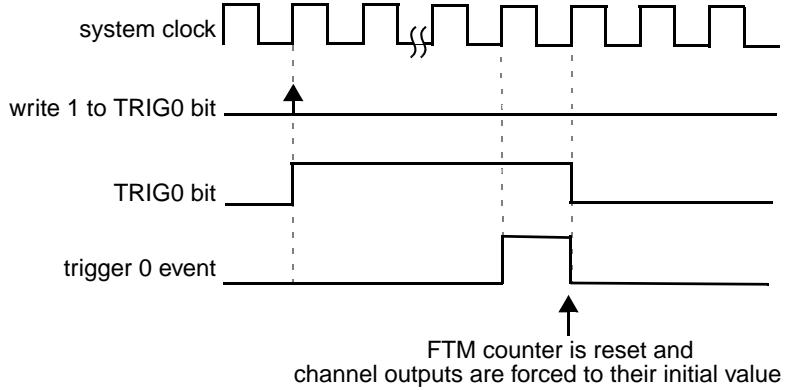


**Figure 12-75. FTM counter synchronization when (REINIT = 1) and (PWMSYNC = 0) and (software trigger was used)**



**Figure 12-76. FTM counter synchronization when (REINIT = 1) and (PWMSYNC = 0) and (a hardware trigger was used)**

If REINIT = 1 and PWMSYNC = 1, then this synchronization is made on the next enabled hardware trigger event. The trigger enable bit (TRIGn) is cleared when the enabled hardware trigger n event is detected (Figure 12-77).



**Figure 12-77. FTM counter synchronization when (REINIT = 1) and (PWMSYNC = 1) and (a hardware trigger was used)**

### 12.5.11.8 Summary of PWM synchronization

Table 12-27 shows the summary of PWM synchronization.

**Table 12-27. Summary of PWM synchronization**

Register or bit	PWMSYNC	REINIT	SYNCHOM	CNTMAX	CNTMIN	SYNCEN	Description
FTMxCNTINH:L	X	X	X	X	X	X	Changes take effect after the second byte is written. Effect is seen after the next TOF or PWM synchronization.
FTMxMODH:L	0	0	X	1	0	X	FTMxMODH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled (hardware or software) trigger has occurred.
	0	0	X	0	1	X	FTMxMODH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled (hardware or software) trigger has occurred.
	0	1	X	X	X	X	FTMxMODH:L are updated with their write buffer contents when the enabled (hardware or software) trigger occurs.
	1	X	X	1	0	X	FTMxMODH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled software trigger has occurred.
	1	X	X	0	1	X	FTMxMODH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled software trigger has occurred.

**Table 12-27. Summary of PWM synchronization**

Register or bit	PWMSYNC	REINIT	SYNCHOM	CNTMAX	CNTMIN	SYNCEN	Description
FTMxCnVH:L	0	0	X	1	0	1	FTMxCnVH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled (hardware or software) trigger has occurred.
	0	0	X	0	1	1	FTMxCnVH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled (hardware or software) trigger has occurred.
	0	1	X	X	X	1	FTMxCnVH:L are updated with their write buffer contents when the enabled (hardware or software) trigger occurs.
	1	X	X	1	0	1	FTMxCnVH:L are updated with their write buffer contents when the counter reaches its maximum value after the enabled software trigger has occurred.
	1	X	X	0	1	1	FTMxCnVH:L are updated with their write buffer contents when the counter reaches its minimum value after the enabled software trigger has occurred.
FTMxCNTH:L	0	1	X	X	X	X	FTMxCNTH:L are forced to the FTM counter initial value when the enabled (hardware or software) trigger occurs.
	1	1	X	X	X	X	FTMxCNTH:L are forced to the FTM counter initial value when the enabled hardware trigger occurs.
FTMxOUTMASK	X	X	0	X	X	X	Changes to FTMxOUTMASK take effect on the next rising edge of the system clock.
	0	X	1	X	X	X	FTMxOUTMASK is updated with its write buffer contents when the enabled (hardware or software) trigger occurs.
	1	X	1	X	X	X	FTMxOUTMASK is updated with its write buffer contents when the enabled hardware trigger occurs.
SWSYNC bit	0	0	X	1	0	X	SWSYNC bit is cleared when the counter reaches its maximum value after the enabled software trigger has occurred.
	0	0	X	0	1	X	SWSYNC bit is cleared when the counter reaches its minimum value after the enabled software trigger has occurred.
	0	1	X	X	X	X	SWSYNC bit is cleared when the enabled software trigger occurs.
	1	X	X	1	0	X	SWSYNC bit is cleared when the counter reaches its maximum value after the enabled software trigger has occurred.
	1	X	X	0	1	X	SWSYNC bit is cleared when the counter reaches its minimum value after the enabled software trigger has occurred.
TRIGn bit	X	X	X	X	X	X	TRIGn bit is cleared when the enabled hardware trigger has occurred.

## 12.5.12 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non-zero).

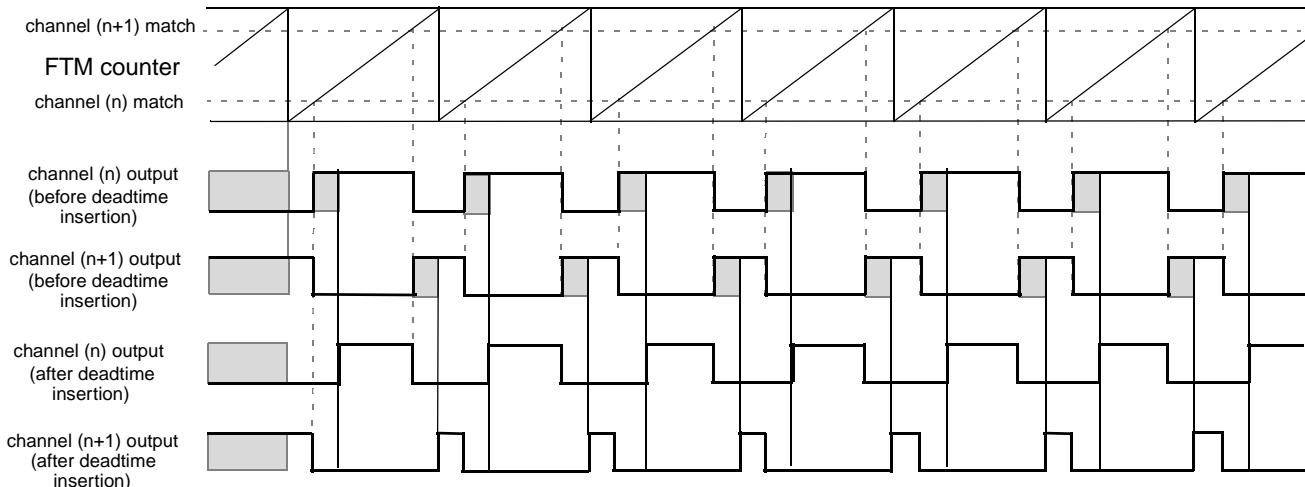
FTMxDEADTIME register defines the deadtime delay which can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo (number of the deadtime prescaler clocks).

The deadtime delay insertion ensures that no two complementary signals (channel (n) and (n+1)) drive the active state at the same time.

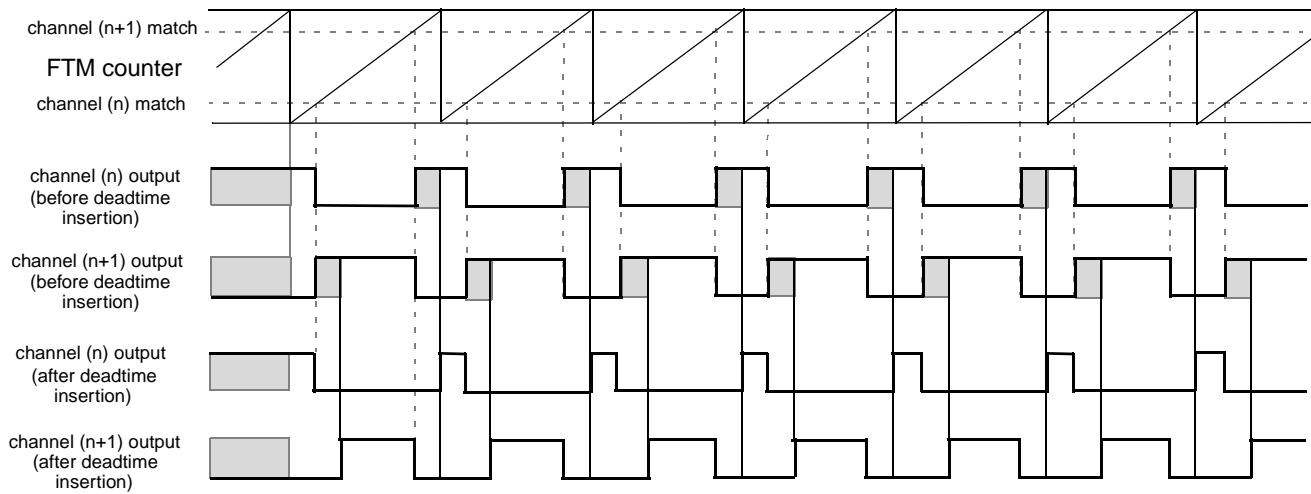
For  $\text{POL}(n) = 0$ ,  $\text{POL}(n+1) = 0$  and deadtime enabled, a rising edge on the output of channel (n) remains low for the duration of the deadtime delay, after which the rising edge appears on the output. Similarly, when a falling edge is due on the output of channel (n), the channel (n+1) output remains low for the duration of the deadtime delay, after which the channel (n+1) output will have a rising edge.

For  $\text{POL}(n) = 1$ ,  $\text{POL}(n+1) = 1$  and deadtime enabled, a falling edge on the output of channel (n) remains high for the duration of the deadtime delay, after which the falling edge appears on the output. Similarly, when a rising edge is due on the output of channel (n), the channel (n+1) output remains high for the duration of the deadtime delay, after which the channel (n+1) output will have a falling edge.

If the deadtime delay is greater than or equal to the channel (n) duty cycle ( $\text{FTMxC}(n+1)\text{VH:L} - \text{FTMxC}(n)\text{VH:L}$ ), then the channel (n) output is always 0. Equally, if the deadtime delay is greater than or equal to the channel (n+1) duty cycle ( $\text{FTMxMODH:L} - \text{FTMxCNTINH:L} - (\text{FTMxC}(n+1)\text{VH:L} - \text{FTMxC}(n)\text{VH:L})$ ), then the channel (n+1) output is always 0.



**Figure 12-78. Deadtime insertion with ELSnB:ELSnA = 1:0**



**Figure 12-79. Deadtime insertion with ELSnB:ELSnA = X:1**

#### NOTE

Deadtime insertion is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0) and (COMP = 1). Deadtime insertion with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) or (COMP = 0) is not recommended and its results are not guaranteed.

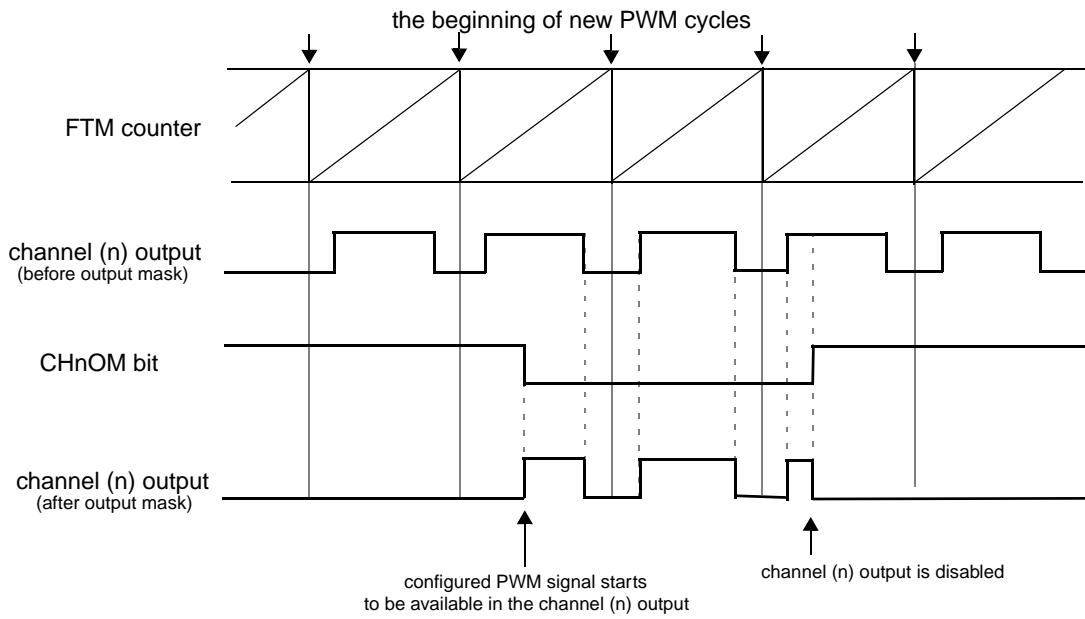
### 12.5.13 Output Mask

The output mask register FTmxOUTMASK can be used to force channel outputs to their inactive state through software (for example: to control a BLDC motor).

Any write to a CHnOM bits updates the FTmxOUTMASK write buffer. The CHnOM bit is updated with the value of its corresponding bit in the FTmxOUTMASK write buffer according to [Section 12.5.11.6, “CHnOM synchronization”](#).

If CHnOM = 1, then the channel (n) output is forced to its inactive state, defined by the POLn bit in register FTmxPOL. If CHnOM = 0, then the channel (n) output is unaffected by the output mask function.

When a CHnOM bit is cleared, the channel (n) output is enabled ([Figure 12-80](#)).



**Figure 12-80. Output mask**

The Table 12-28 shows the output mask result before the polarity control.

**Table 12-28. Output mask result for channel (n) (before the polarity control)**

CHnOM bit	Output mask input	Output mask result
0	inactive state	inactive state
	active state	active state
1	inactive state	inactive state
	active state	

#### NOTE

Output mask is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). Output mask with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

#### 12.5.14 Fault control mode

The fault control mode is enabled if (FTMEN = 1) and (FAULTM[1:0] not = 00).

FTM can have up to four fault inputs. FAULTnEN bit (where n = 0, 1, 2, 3) enables the fault input n and FFLTRnEN bit enables the fault input n filter. FFVAL[3:0] bits select the value of the enabled filter in each enabled fault input.

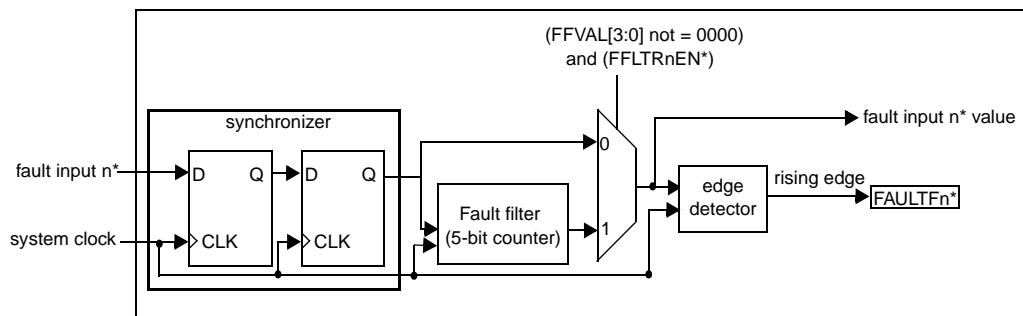
First each fault input signal is synchronized by the system clock (synchronizer block in Figure 12-81). Following synchronization, the fault input n signal enters the filter block. When there is a state change in

the fault input n signal, the 5-bit counter is reset and starts counting up. As long as the new state is stable on the fault input n, the counter continues to increment. If the 5-bit counter overflows (the counter exceeds the value of the FFVAL[3:0] bits), the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the fault input n signal before validation (counter overflow), the counter is reset. At the next input transition, the counter will start counting again. Any pulse that is shorter than the minimum value selected by FFVAL[3:0] bits (\* system clock) is regarded as a glitch and is not passed on to the edge detector.

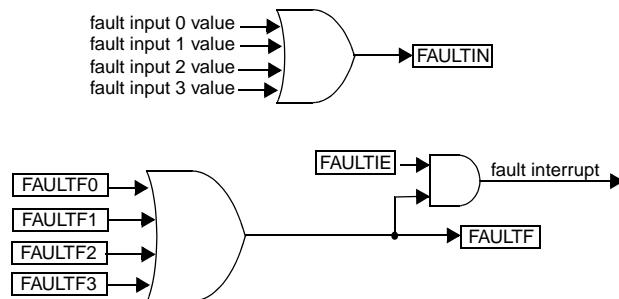
The fault input n filter is disabled when the FFVAL[3:0] bits are zero or when FAULTnEN = 1. In this case the fault input n signal is delayed 2 rising edges of the system clock and the FAULTFn bit is set on 3th rising edge of the system clock after a rising edge occurs on the fault input n.

If (FFVAL[3:0] not = 0000 and FAULTnEN = 1) then the fault input n signal is delayed ( $3 + \text{FFVAL}[3:0]$ ) rising edges of the system clock, that is, the FAULTFn bit is set ( $4 + \text{FFVAL}[3:0]$ ) rising edges of the system clock after a rising edge occurs on the fault input n.



\* where  $n = 3, 2, 1, 0$

**Figure 12-81. Fault input n control Block Diagram**



**Figure 12-82. FAULTF and FAULTIN bits and fault interrupt**

If the fault control and fault input n are enabled and a rising edge at the fault input n signal is detected, then the FAULTFn bit is set (Figure 12-81). The FAULTF bit is the logic OR of FAULTFn[3:0] bits (Figure 12-82).

If the fault control is enabled (FAULTM[1:0] non zero), a fault condition has occurred (rising edge at the logic OR of the enabled fault input) and (FAULTEN = 1), then channel (n) and (n+1) outputs are forced to their safe value (that is, the channel (n) output is forced to the value of POL(n) and the channel (n+1) is forced to the value of POL(n+1)).

The fault interrupt is generated when (FAULTF = 1) and (FAULTIE = 1) (Figure 12-82). This interrupt request remains set until:

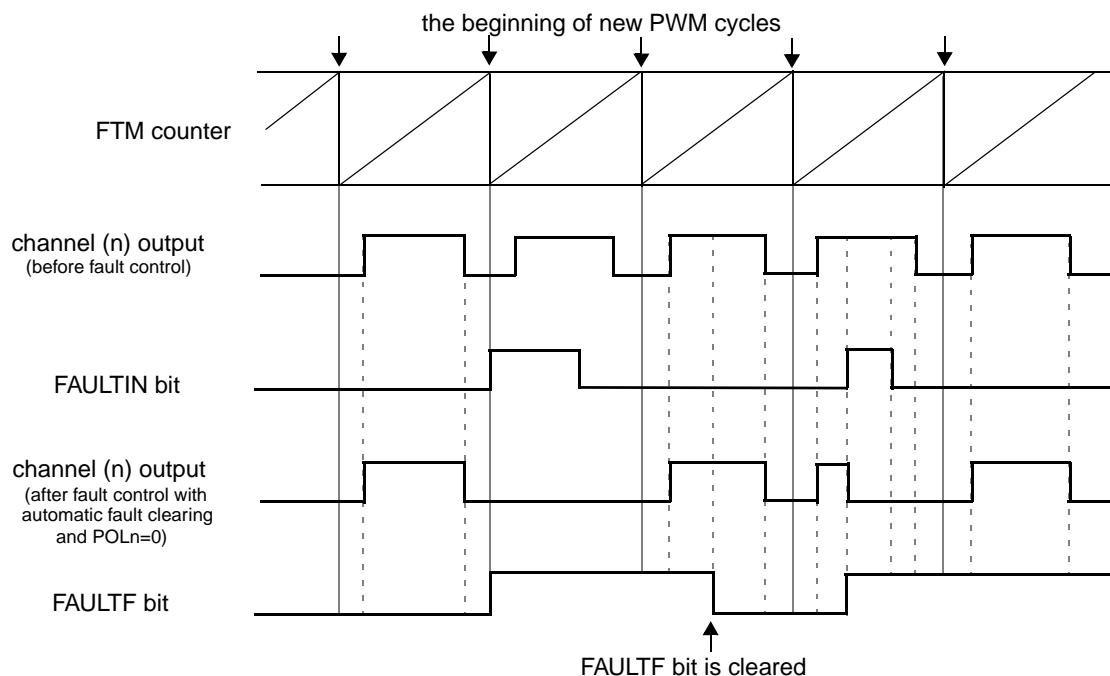
- Software clears the FAULTF bit (by reading FAULTF bit as 1 and writing 0 to it)
- Software clears the FAULTIE bit
- A reset occurs

#### NOTE

Fault control mode is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). Fault control with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

#### 12.5.14.1 Automatic Fault Clearing

If the automatic fault clearing is selected (FAULTM[1:0] = 11), then the disabled channel outputs are enabled when the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins (Figure 12-83).



**Figure 12-83. Fault control with automatic fault clearing**

### 12.5.14.2 Manual Fault Clearing

If the manual fault clearing is selected ( $\text{FAULTM}[1:0] = 01$  or  $10$ ), then disabled channel outputs are enabled when the  $\text{FAULTF}$  bit is cleared and a new PWM cycle begins (Figure 12-84).

It is possible to manually clear a fault, by clearing the  $\text{FAULTF}$  bit, and enable disabled channels regardless of the fault input signal ( $\text{FAULTIN}$ ) (the filter output if the filter is enabled or the synchronizer output if the filter is disabled). However, it is recommended to verify the value of the fault input signal (value of the  $\text{FAULTIN}$  bit) before clearing the  $\text{FAULTF}$  bit to avoid unpredictable results.

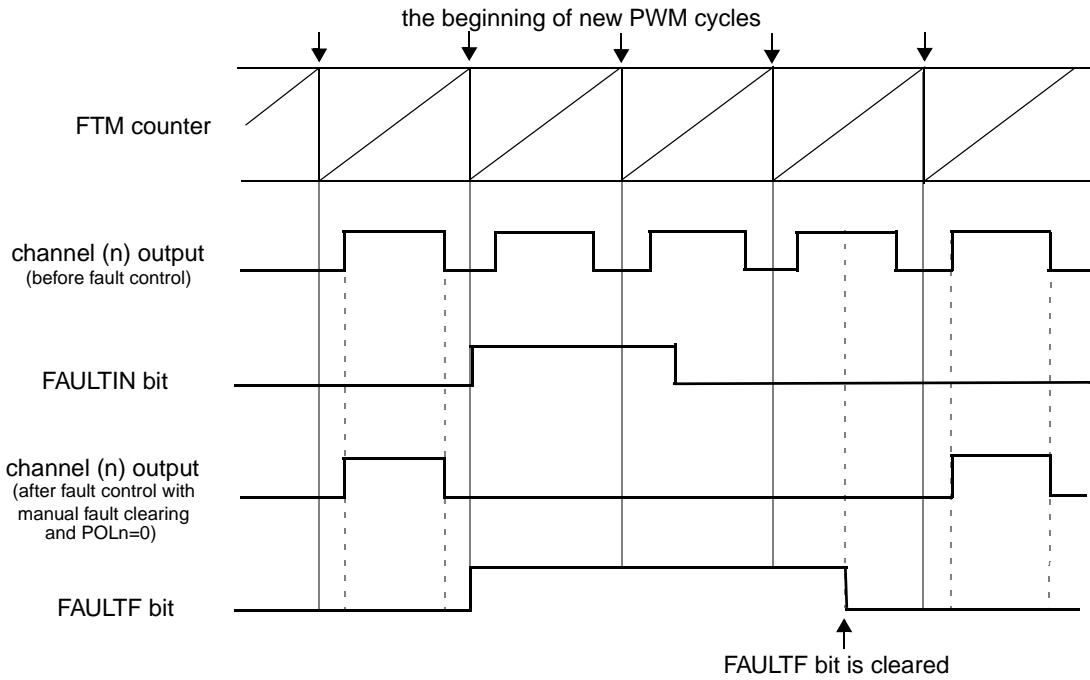


Figure 12-84. Fault control with manual fault clearing

### 12.5.15 Polarity Control

The polarity control mode is enabled if ( $\text{FTMEN} = 1$ ). Each channel has its polarity control defined by its  $\text{POLn}$  bit.

If ( $\text{POLn} = 0$ ), the signal driving the polarity control is the same polarity as the channel (n) output.

If ( $\text{POLn} = 1$ ), the signal driving the polarity control is negated at the channel (n) output.

#### NOTE

Polarity control is only available when ( $\text{FTMEN} = 1$ ) and ( $\text{COMBINE} = 1$ ) and ( $\text{CPWMS} = 0$ ). Polarity control with ( $\text{FTMEN} = 0$ ) or ( $\text{COMBINE} = 0$ ) or ( $\text{CPWMS} = 1$ ) is not recommended and its results are not guaranteed.

## 12.5.16 Initialization

The initialization of the output channels is enabled if (FTMEN = 1). When a logic 1 is written to the INIT bit, the value of the CHnOI bit is forced to the channel (n) output. [Figure 12-85](#) shows the priority of the initialization feature over the other features that are combined to generate the channels (n) and (n+1) output.

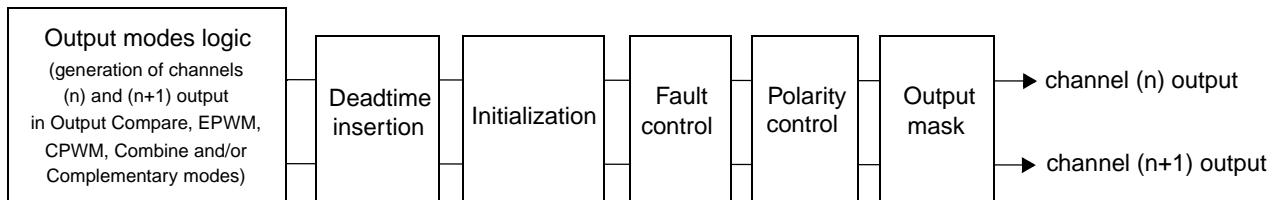
It is recommended using the initialization only if the FTM counter is disabled (that is, CLKS[1:0] = 00), otherwise unpredictable behavior is likely.

### NOTE

Initialization is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). Initialization with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

## 12.5.17 Features Priority

[Figure 12-85](#) shows the priority of the features that can be combined to generate channel (n) and (n+1) outputs.



**Figure 12-85. FTM features priority**

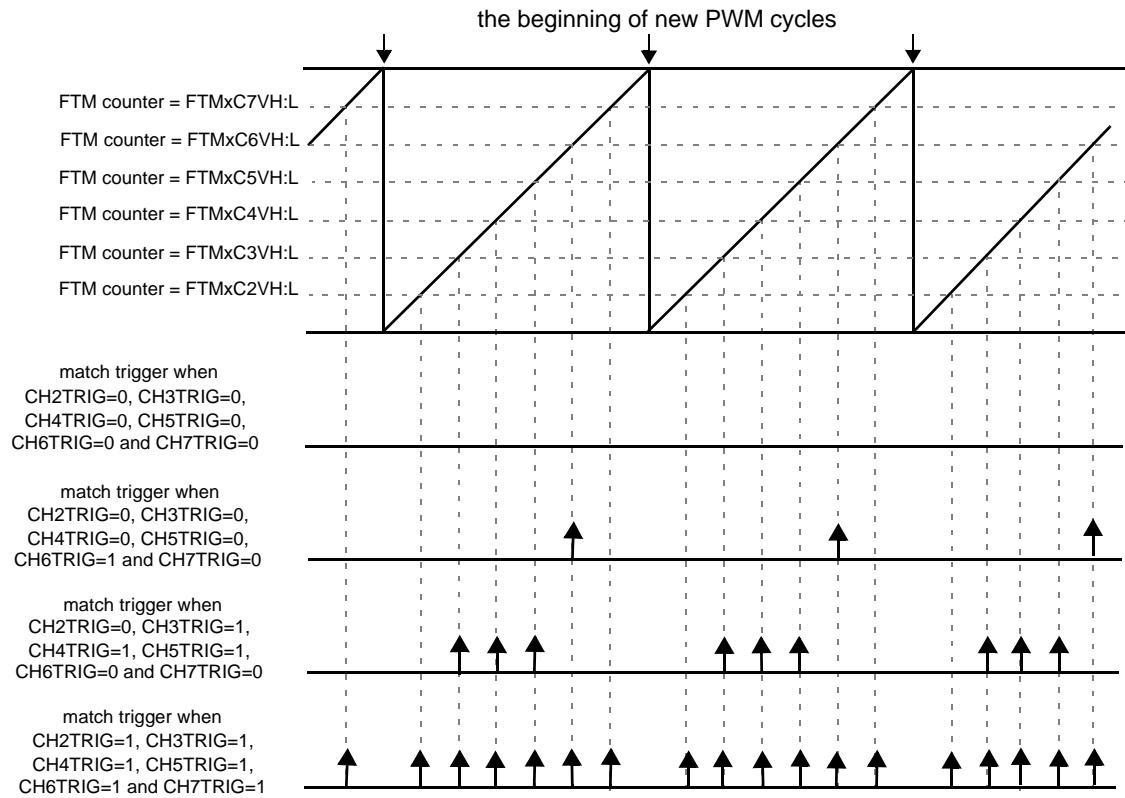
## 12.5.18 Match Trigger

The match triggers are generated if (FTMEN = 1) and (one or more channels were selected by the CHjTRIG bit where  $j = 2, 3, 4, 5, 6$  or 7). The CHjTRIG bit defines if the channel (j) match (that is, FTM counter = FTMXC(j)VH:L) generates the match trigger.

The match trigger is an output signal to provide a trigger for other modules.

One or more FTM channels can be selected to generate multiple match triggers in one PWM period (as shown in [Figure 12-86](#)).

The width of a match trigger is a system clock period.



**Figure 12-86. Match triggers**

#### NOTE

Match trigger is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). Match trigger with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

### 12.5.19 Initialization Trigger

The initialization trigger is generated if (FTMEN = 1) and (INITTRIGEN = 1). The initialization trigger is generated when the value of FTMxCNTINH:L registers are loaded into FTM counter in the following cases:

- the FTMxCNTINH:L value is automatically loaded into the FTM counter after the maximum value has been reached ([Figure 12-87](#))
- when there is a write to FTMxCNTH or FTMxCNTL register ([Figure 12-88](#))
- when there is the FTM counter synchronization ([Figure 12-89](#))
- if (FTMxCNTH:L = FTMxCNTINH:L) and (CLKS[1:0] = 00) and a value different from zero is written to CLKS[1:0] bits ([Figure 12-90](#))

The initialization trigger is an output signal to provide a trigger for other modules.

The width of an initialization trigger is a system clock period.

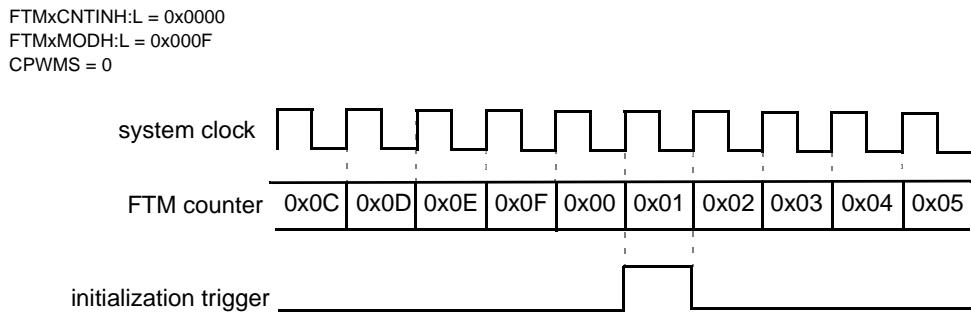


Figure 12-87. Initialization trigger generated when the FTM counting achieved the value of FTMxCNTINH:L

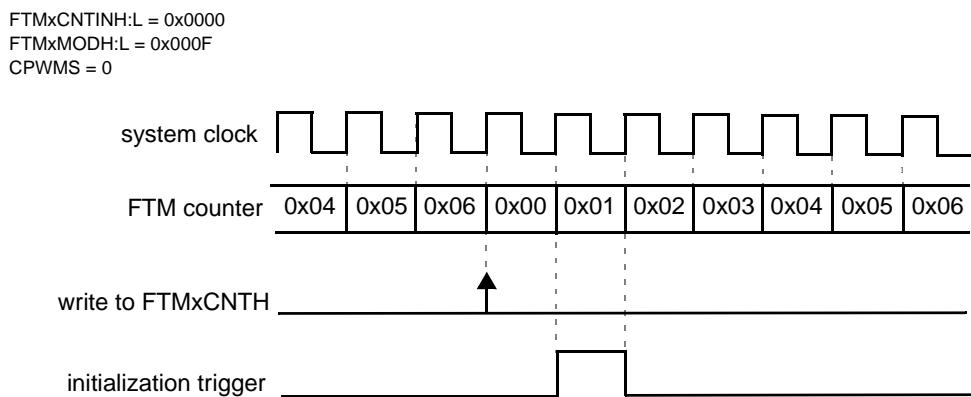


Figure 12-88. Initialization trigger generated when there is a write to FTMxCNTH (or FTMxCNTL)

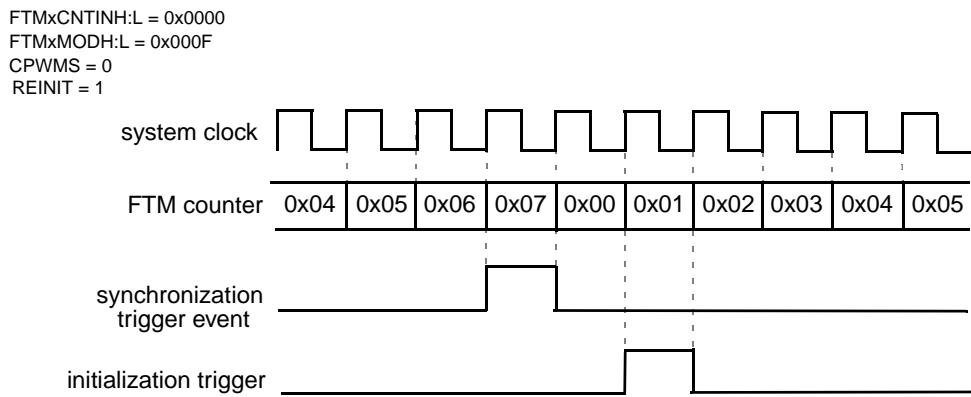
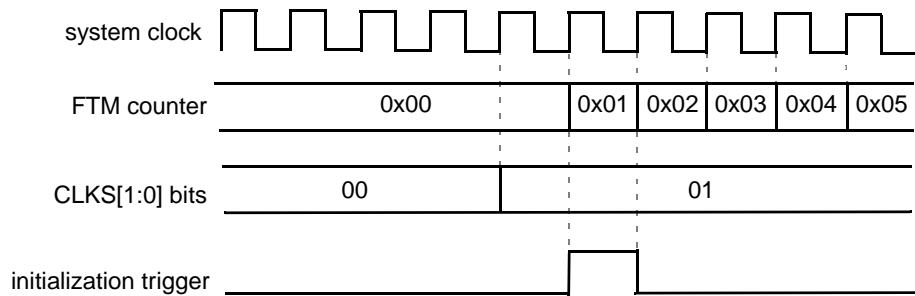


Figure 12-89. Initialization trigger generated when there is the FTM counter synchronization

FTMxCNTINH:L = 0x0000  
FTMxMODH:L = 0x000F  
CPWMS = 0



**Figure 12-90. Initialization trigger generated if (FTMxCNTH:L = FTMxCNTINH:L) and (CLKS[1:0] = 00) and a value different from zero is written to CLKS[1:0] bits**

#### NOTE

Initialization trigger is only available when (FTMEN = 1) and (COMBINE = 1) and (CPWMS = 0). Initialization trigger with (FTMEN = 0) or (COMBINE = 0) or (CPWMS = 1) is not recommended and its results are not guaranteed.

### 12.5.20 Capture Test Mode

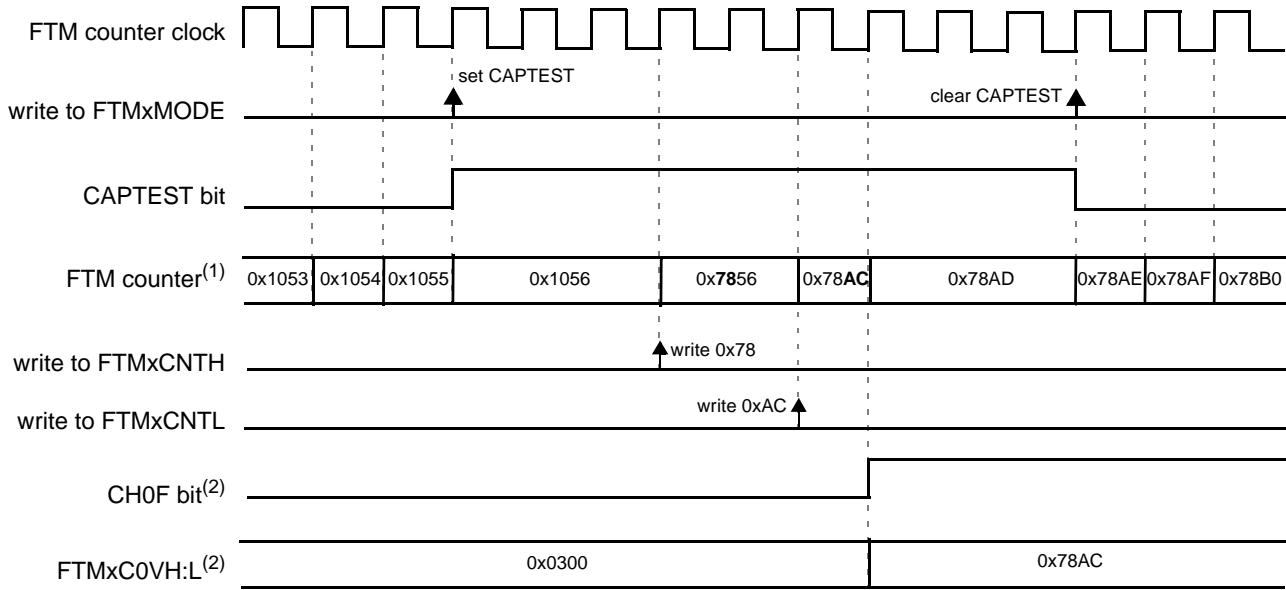
The capture test mode allows to test the FTMxCnVH:L registers, the FTM counter and the interconnection logic between the FTM counter and FTMxCnVH:L registers.

In this test mode all channels must be configured for input capture mode (COMBINE = 0 and CPWMS = 0 and MSnB:MSnA = 0:0).

When the capture test mode is enabled (CAPTEST bit is set) the FTM counter is frozen and any write to FTMxCNTH and FTMxCNTL updates directly the FTM counter ([Figure 12-91](#)). After both bytes were written (independent of the order), all FTMxCnVH:L registers are updated with the value that was written to FTMxCNTH:L registers and CHnF bits are set. Thus the FTM counter is updated with its next value according to its configuration (its next value depends on FTMxCNTINH:L, FTMxMODH:L and the value that was written to FTM counter).

The next reads of FTMxCnVH:L registers return the value that was written to FTM counter and the next reads of FTMxCNTH:L registers return the next value of the FTM counter.

The read coherency mechanism of FTMxCNTH:L and FTMxCnVH:L of the input capture mode remains enabled.



#### Notes

(1) FTM counter configuration: (CPWMS=0), (FTMxCNTINH:L=0x0000) and (FTMxMODH:L=0xFFFF)

(2) FTM channel 0 configuration: input capture mode (COMBINE=0, CPWMS=0, MSnB=0 and MSnA=0)

**Figure 12-91. Capture test mode**

#### NOTE

Capture test mode is only available when (FTMEN = 1). Capture test mode with (FTMEN = 0) is not recommended and its results are not guaranteed.

### 12.5.21 TPM Emulation

This section describe the FTM features that are selected according to the FTMEN bit.

#### 12.5.21.1 FTM counter clock

If (CLKS[1:0] = 01), then FTMEN bit selects if the system clock is divided by 1 or divided by 2 in the FTM (Section 12.4.3, “FTM Status and Control Register (FTMxSC)”).

If (CLKS[1:0] = 01) and (FTMEN = 0), then the clock of FTM counter is the system clock divided by 2.

If (CLKS[1:0] = 01) and (FTMEN = 1), then the clock of FTM counter is the system clock.

#### 12.5.21.2 FTMxMODH:L and FTMxCnVH:L synchronization

If (FTMEN = 0), then the FTMxMODH:L and FTMxCnVH:L registers are updated according to the Section 12.5.10, “Load of the registers with write buffers” and they are not updated by PWM synchronization.

If ( $\text{FTMEN} = 1$ ), then the  $\text{FTMxMODH:L}$  and  $\text{FTMxCnVH:L}$  registers are updated only by PWM synchronization ([Section 12.5.11, “PWM synchronization”](#)).

### 12.5.21.3 Free-running counter

If ( $\text{FTMEN} = 0$ ), then the FTM counter is a free-running counter when ( $\text{FTMxMODH:L} = 0x0000$ ) or ( $\text{FTMxMODH:L} = 0xFFFF$ ) ([Section 12.5.3.3, “Free running counter”](#)).

If ( $\text{FTMEN} = 1$ ), then the FTM counter is a free-running counter when ( $\text{CPWMS} = 0$ ) and ( $\text{FTMxCNTINH:L} = 0x0000$ ) and ( $\text{FTMxMODH:L} = 0xFFFF$ ).

### 12.5.21.4 Write to $\text{FTMxSC}$

If ( $\text{FTMEN} = 0$ ), then a write to the  $\text{FTMxSC}$  register resets the write coherency mechanism of  $\text{FTMxMODH:L}$  registers ([Section 12.4.5, “FTM Counter Modulo Registers \( \$\text{FTMxMODH:FTMxMODL}\$ \)”](#)).

If ( $\text{FTMEN} = 1$ ), then a write to the  $\text{FTMxSC}$  register does not reset the write coherency mechanism of  $\text{FTMxMODH:L}$  registers.

### 12.5.21.5 Write to $\text{FTMxCnSC}$

If ( $\text{FTMEN} = 0$ ), then a write to the  $\text{FTMxCnSC}$  register resets the write coherency mechanism of  $\text{FTMxCnVH:L}$  registers ([Section 12.4.7, “FTM Channel Value Registers \( \$\text{FTMxCnVH:FTMxCnVL}\$ \)”](#)).

If ( $\text{FTMEN} = 1$ ), then a write to the  $\text{FTMxCnSC}$  register does not reset the write coherency mechanism of  $\text{FTMxCnVH:L}$  registers.

## 12.6 Reset Overview

The FTM is reset whenever any MCU reset occurs

## 12.7 FTM Interrupts

### 12.7.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when ( $\text{TOIE} = 1$ ) and ( $\text{TOF} = 1$ ).

### 12.7.2 Channel (n) Interrupt

The channel (n) interrupt is generated when ( $\text{CHnIE} = 1$ ) and ( $\text{CHnF} = 1$ ).

### 12.7.3 Fault Interrupt

The fault interrupt is generated when ( $\text{FAULTIE} = 1$ ) and ( $\text{FAULTF} = 1$ ).

# Chapter 13

## High Speed Analog Comparator 5-V (S08HSCMPV1)

### 13.1 Introduction

The high speed analog comparator module (HSCMP) provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

Some MCUs may have more than one HSCMP, so references to modules and register names include a placeholder character to identify which HSCMP module (use x as placeholder) is being referenced. In cases where only one HSCMP is available on a particular device, the x placeholder is omitted.

The MC9S08MP16 Series has three HSCMP modules, referred to as HSCMP1, HSCMP2, and HSCMP3. [Figure 13-2](#) shows the MC9S08MP16 Series block diagram with the HSCMP highlighted.

### 13.2 Module Configurations

This section provides device-specific information for configuring the HSCMPx modules on MC9S08MP16 Series.

#### 13.2.1 HSCMP Clock Gating

The bus clock to each HSCMP module can be gated on and off using the associated control bit in SCGC1. The CMPDAC1 bit is used to control the clock to both the HSCMP1 and DAC1 modules. The CMPDAC2 bit is used to control the clock to both the HSCMP2 and DAC2 modules. The CMPDAC3 bit is used to control the clock to both the HSCMP3 and DAC3 modules. These bits are set after any reset, which enables the bus clock to these modules. To conserve power, the CMPDACx bit can be cleared to disable the clock to the associated modules when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 13.2.2 Analog Supply Connections

On the MC9S08MP16 Series devices, the VDDA and VSSA power inputs for the HSCMPx modules are supplied by the V<sub>DDA</sub>/V<sub>REFH</sub> and V<sub>SSA</sub>/V<sub>REFL</sub> pins respectively.

#### 13.2.3 HSCMP Analog Input Configuration

Each of the three HSCMP blocks has 8 analog inputs. 4 are connected to a plus side multiplexor (inputs P1 – P4) and 4 are connected to a minus side multiplexor (inputs M1 – M4). On MC9S08MP16 Series, most of the associated HSCMP analog input pins connect to package pins, with most pins connecting to multiple analog inputs. In addition, each DAC module is associated with one HSCMP block. DAC1 output

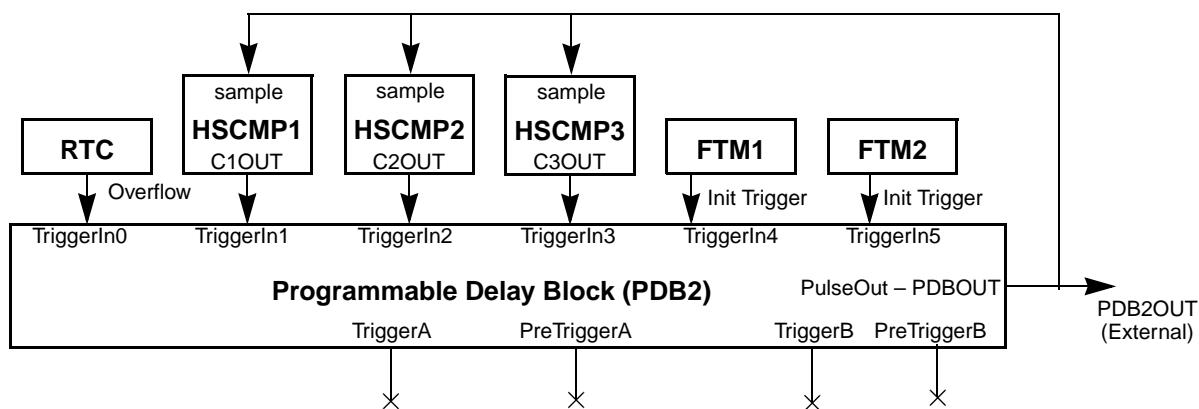
is connected to the M4 input of HSCMP1. DAC2 output is connected to the M4 input of HSCMP2. DAC3 output is connected to the M4 input of HSCMP3. [Table 13-1](#) summarizes the analog comparator input connections.

**Table 13-1. MC9S08MP16 Series HSCMPx Analog Input Connections**

Type	Analog Source	HSCMP Analog Input Signal Description
External	PTB0/ADP0/CIN1	CIN1 connected to P1 and M1 of all three HSCMPx blocks.
External	PTB1/ADP1/C2IN2	C2IN2 is connected to P2 and M2 of HSCMP2.
External	PTB2/ADP2/C1IN2/PGA+	C1IN2 is connected to P2 and M2 of HSCMP1.
External	PTB3/ADP3/C3IN2/PGA-	C3IN2 is connected to P2 and M2 of HSCMP3.
External	PTB4/ADP4/C2IN3	C2IN3 is connected to P3 and M3 of HSCMP2.
External	PTB5/ADP5/C2IN4	C2IN4 is connected to P4 of HSCMP2.
External	PTB6/ADP6/C3IN3	C3IN3 is connected to P3 and M3 of HSCMP3.
External	PTB7/ADP7/C3IN4	C3IN4 is connected to P4 of HSCMP3.
External	PTE3/ADP11/C1IN3	C1IN3 is connected to P3 and M3 of HSCMP1.
External	PTE4/ADP12/C1IN4	C1IN4 is connected to P4 of HSCMP1.
Internal	DAC1 Output	DAC1 is connected to M4 of HSCMP1.
Internal	DAC2 Output	DAC2 is connected to M4 of HSCMP2.
Internal	DAC3 Output	DAC3 is connected to M4 of HSCMP3.

### 13.2.4 PDB2/HSCMP Windowing/Sampling Input Configuration

On MC9S08MP16 Series, each of the HSCMP modules can be configured for Windowing or Sampling modes of operation. The PDB2 PulseOut output is used as the Window/Sample input to all three HSCMP modules. [Figure 13-1](#) shows the configuration used for PDB2.



**Figure 13-1. PDB2 Interface to HSCMP1/HSCMP2/HSCMP3 Windowing**

## 13.2.5 HSCMP Output Configuration

Each HSCMP module has 2 associated outputs – CMPxOUT and CxOUT. The CMPxOUT is used for external pin connections and is differentiated with the intention of being part of an external hysteresis circuit. The CxOUT is used for all internal connections not related to external pins.

### 13.2.5.1 HSCMP/PDB Trigger

On MC9S08MP16 Series, the COUT output from each of the HSCMP modules can be configured to generate trigger events for the PDB1 and PDB2 modules. HSCMP1 is configured to the TriggerIn1 input of both PDB1 and PDB2 modules. HSCMP2 is configured to the TriggerIn2 input of both PDB1 and PDB2 modules. HSCMP3 is configured to the TriggerIn3 input of both PDB1 and PDB2 modules.

### 13.2.5.2 HSCMP/FTM Fault

On MC9S08MP16 Series, the COUT output from each of the HSCMP modules can be configured to generate fault events for the FTM1 and FTM2 modules. [Table 13-2](#) summarizes the FTM fault sources for MC9S08MP16 Series devices.

**Table 13-2. MC9S08MP16 Series HSCMP to FTM Fault Configuration**

Fault Source	FTM Fault Input
HSCMP1 COUT	FTM1 Fault 2 FTM2 Fault 2
HSCMP2 COUT	FTM1 Fault 3 FTM2 Fault 3
HSCMP3 COUT	FTM1 Fault 4 FTM2 Fault 4

### 13.2.5.3 HSCMP/FTM Synchronization

On MC9S08MP16 Series, the COUT output from each of the HSCMP modules can be configured to trigger PWM synchronization events within the FTM1 and FTM2 modules. [Table 13-1](#) summarizes the FTM synchronization sources for MC9S08MP16 Series devices.

**Table 13-3. MC9S08MP16 Series HSCMP to FTM Synchronization Configuration**

Trigger Source	FTM Synchronization Input
HSCMP1 COUT	FTM1 Trigger 1 FTM2 Trigger 1
HSCMP2 COUT	FTM1 Trigger 2
HSCMP3 COUT <sup>1</sup>	FTM2 Trigger 2

<sup>1</sup> The FTM2T2S bit in SOPT2 register must be clear to configure HSCMP3 COUT as FTM2 Trigger 2 source.

### 13.2.5.4 HSCMP/FTM Input Capture

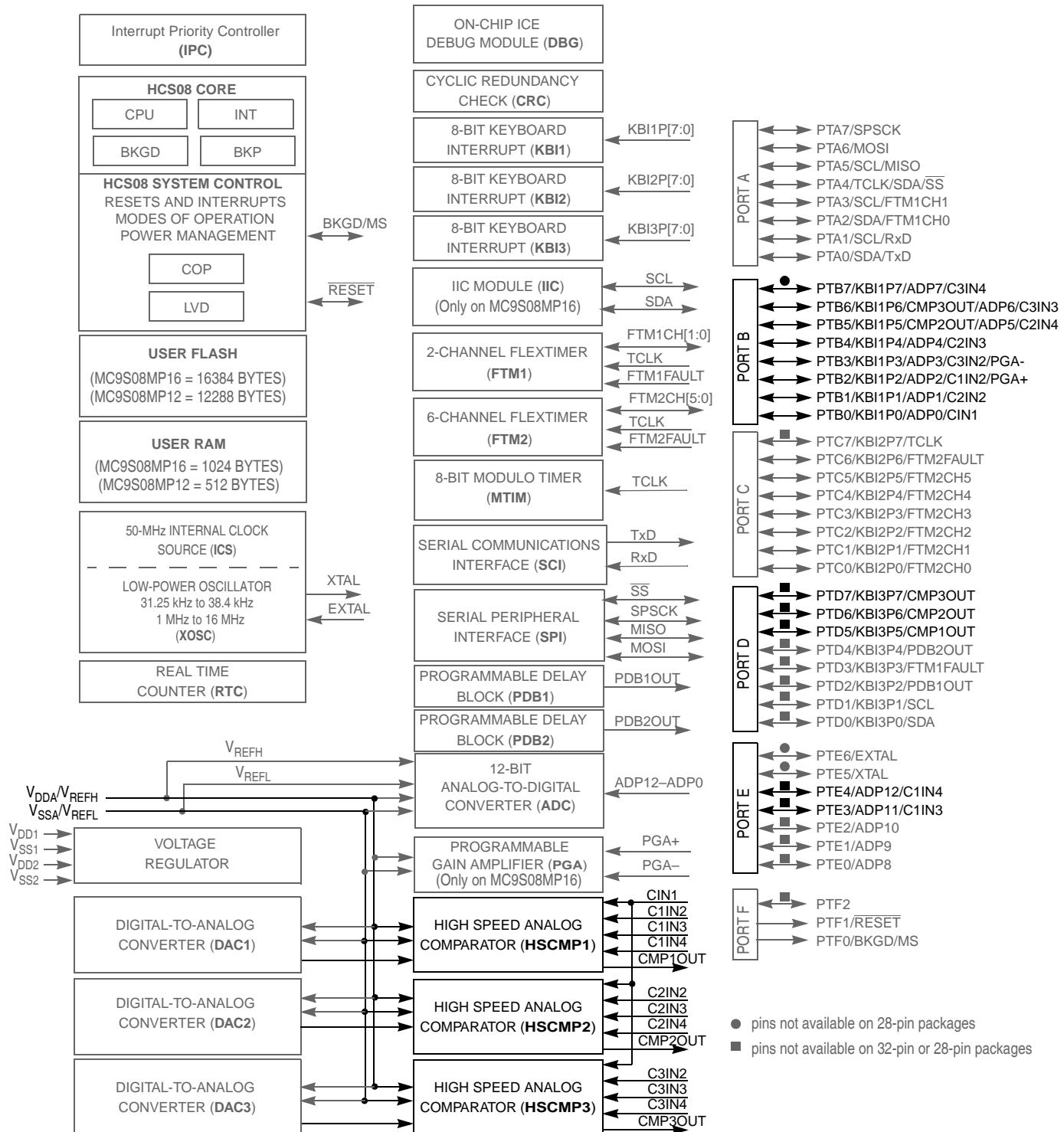
The HSCMP1 module can be configured to connect the COUT output of the analog comparator to a FTM1 input capture channel 0 by setting the ACIC1 bit in SOPT1. With ACIC1 set, the FTM1CH0 pin is not available externally regardless of the configuration of the FTM1 module.

The HSCMP2 module can be configured to connect the COUT output of the analog comparator to a FTM1 input capture channel 1 by setting the ACIC2 bit in SOPT1. With ACIC2 set, the FTM1CH1 pin is not available externally regardless of the configuration of the FTM1 module.

### 13.2.5.5 CMP2OUT and CMP3OUT Pin Repositioning

The CMP2OUT pin can be repositioned under software control using CMP2OPS in SOPT2. When CMP2OPS is clear, the reset default condition, PTD6 is the CMP2OUT output pin. When CMP2OPS is set, PTB5 is the CMP2OUT output pin.

The CMP3OUT pin can be repositioned under software control using CMP3OPS in SOPT2. When CMP3OPS is clear, the reset default condition, PTD7 is the CMP3OUT output pin. When CMP3OPS is set, PTB6 is the CMP3OUT output pin.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

$V_{DD2}$  pad is tied internally on 32-pin and 28-pin packages,

$V_{SS2}$  pad is tied internally on 28-pin packages

Figure 13-2. MC9S08MP16 Series Block Diagram Highlighting HSCMP Blocks and Pins

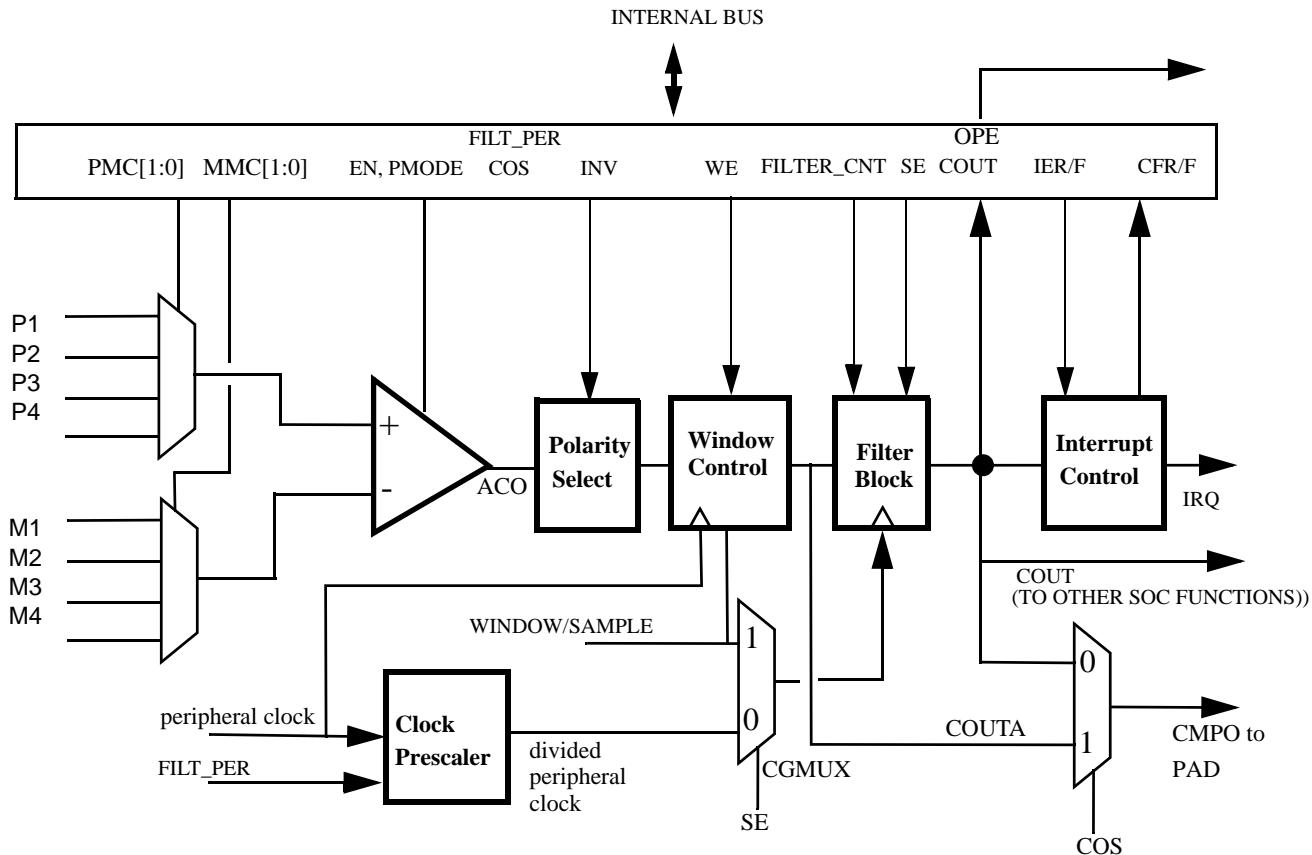
### 13.3 Features

The HSCMP has the following features:

- Operates over the entire supply range
- Inputs may range from rail to rail.
- Less than 40 mV of input offset.
- Less than 15mV of hysteresis.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Selectable inversion on comparator output
- Comparator output may be:
  - Sampled
  - Windowed (ideal for certain PWM zero-crossing-detection applications)
  - Digitally Filtered
    - Filter can be bypassed
    - may be clocked via external SAMPLE signal or scaled peripheral clock
- External hysteresis can be used at the same time that the output filter is used for internal functions.
- The positive and minus inputs of the comparator are both driven from 4-to-1 muxes which allow additional flexibility in assigning IO as comparator inputs during PCB design.
- Two software selectable performance levels
  - shorter propagation delay at the expense of higher power. This mode can be used only when the VDDA rail is above the low voltage interrupt trip point.
  - low power, with longer propagation delay.

## 13.4 Block Diagram

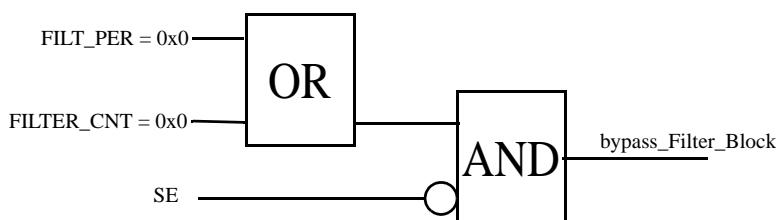
The block diagram for the High Speed Comparator module is shown in [Figure 13-3](#).



**Figure 13-3. High Speed Comparator Module Block Diagram**

In [Figure 13-3](#):

- The window control block is completely bypassed when WE = 0
- If WE = 1, the comparator output will be sampled on every peripheral clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter Block is bypassed when not in use.



**Figure 13-4. Filter Block Bypass Logic**

- The Filter Block acts as a simple sampler if `bypass_Filter_Block && FILTER_CNT = 0x1`.

- The Filter Block filters based on multiple samples when `bypass_Filter_Block && FILTER_CNT > 0x1`
  - If SE = 1, the external SAMPLE input is used as sampling clock
  - If SE = 0, the divided peripheral clock is used as sampling clock
- If enabled, the Filter Block will incur up to 1 IP Bus additional latency penalty on COUT due to the fact that COUT (which is crossing clock domain boundaries) must be resynchronized to the peripheral clock.
- WE and SE are mutually exclusive.

## 13.5 Pin Descriptions

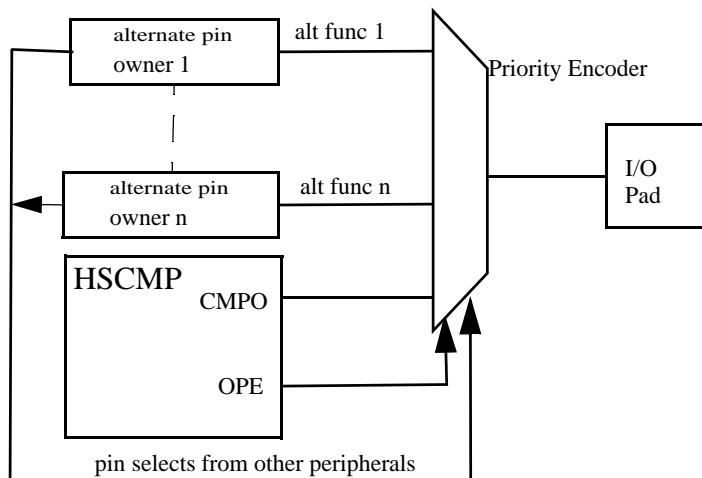
### 13.5.1 External Pins

The HSCMP has up to eight external analog input pins and one external output pin. Each input pin can accept an input voltage that varies across the full operating voltage range of the MCU. If the module is not enabled, these pins can be used as digital inputs or outputs. Unused inputs can also be used as digital inputs or outputs. Comparator input pins which are not associated with specific SoC IO pads will normally be tied to VSSA. The user is responsible for ensuring that both comparator inputs are not simultaneously tied to VSSA.

Consult the specific MCU documentation to determine what functions are shared with analog inputs. As shown in the block diagram, the P<sub>n</sub> pins are connected to the comparator non-inverting input. M<sub>n</sub> pins are connected to the inverting input of the comparator.

The user can select either filtered or unfiltered comparator outputs for use on an external IO pad.

Operation of HSCMPxCR1[OPE] varies from device to device. For this S08 device, a simplified example is shown in [Figure 13-5](#).



**Figure 13-5. OPE Operation on S08 and Flexis Devices**

## 13.6 Functional Description

The High Speed Comparator can be used to compare two analog input voltages applied to P<sub>n</sub> and M<sub>n</sub>. The analog comparator output (ACO) is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting HSCMPxCR1[INV] = 1.

The HSCMPxSCR[IER] and HSCMPxSCR[IEF] bits are used to select the condition which will cause the comparator module to assert an interrupt to the processor. HSCMPxSCR[CFR] bit is set on a rising edge of the comparator output. HSCMPxSCR[CFF] is set on a falling edge of the comparator output. The (optionally filtered) comparator output can be read directly through the HSCMPxSCR[COUT] bit.

### 13.6.1 HSCMP Functional Modes

There are three main sub-blocks to the comparator module: the comparator itself, the window function and the filter function. The filter can be clocked from an internally generated clock, or via an external sample input. Additionally, the filter is programmable with respect to how many samples must agree before a change on the output is registered. In the simplest case, only 1 sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using HSCMPxCR1[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The “windowing mode” is enabled by setting HSCMPxCR1[WE]. When set, the comparator output is sampled only when the WINDOW input signal is equal to one. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in [Table 13-4](#). Individual modes are discussed below.

**Table 13-4. Comparator Sample/Filter Controls**

Mode #	EN	WE	S E	FILT_CNT	FILT_PER	Operation
1	0	X	X	X	X	<b>Disabled</b> <a href="#">Section 13.6.1.1, “Disabled Mode (# 1)”</a>
2A	1	0	0	0x0	X	<b>Continuous Mode</b> <a href="#">Section 13.6.1.2, “Continuous Mode (#s 2A &amp; 2B)”</a>
2B	1	0	0	X	0x00	
3A	1	0	1	0x1	X	<b>Sampled, Non-Filtered mode</b> <a href="#">Section 13.6.1.3, “Sampled, Non-Filtered Mode (#s 3A &amp; 3B)”</a>
3B	1	0	0	0x1	> 0x0	
4A	1	0	1	> 0x1	X	<b>Sampled, Filtered mode</b> <a href="#">Section 13.6.1.4, “Sampled, Filtered Mode (#s 4A &amp; 4B)”</a>
4B	1	0	0	> 0x1	> 0x0	
5A	1	1	0	0x0	X	<b>Windowed mode</b> Comparator output is sampled on every rising peripheral clock edge when SAMPLE=1 to generate COUTA <a href="#">Section 13.6.1.5, “Windowed Mode (#s 5A &amp; 5B)”</a>
5B	1	1	0	X	0x00	

**Table 13-4. Comparator Sample/Filter Controls (continued)**

<b>Mode #</b>	<b>EN</b>	<b>WE</b>	<b>S E</b>	<b>FILT_CNT</b>	<b>FILT_PER</b>	<b>Operation</b>
6	1	1	0	0x1	0x01–0xFF	<b>Windowed/Resampled mode</b> Comparator output is sampled on every rising peripheral clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by FILT_PER to generate COUT. <a href="#">Section 13.6.1.6, "Windowed/Resampled Mode (# 6)"</a>
All other combinations of EN, WE, SE, FILT_CNT and FILT_PER are illegal.						
7	1	1	0	> 0x1	0x01 - 0xFF	<b>Windowed/Filtered mode</b> Comparator output is sampled on every rising peripheral clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT. <a href="#">Section 13.6.1.7, "Windowed/Filtered Mode (#7)"</a>

For cases where a comparator is used to drive a fault input of the PWM, it should generally be configured to operate in continuous mode, so that an external fault can immediately pass through the comparator to the PWM fault circuitry.

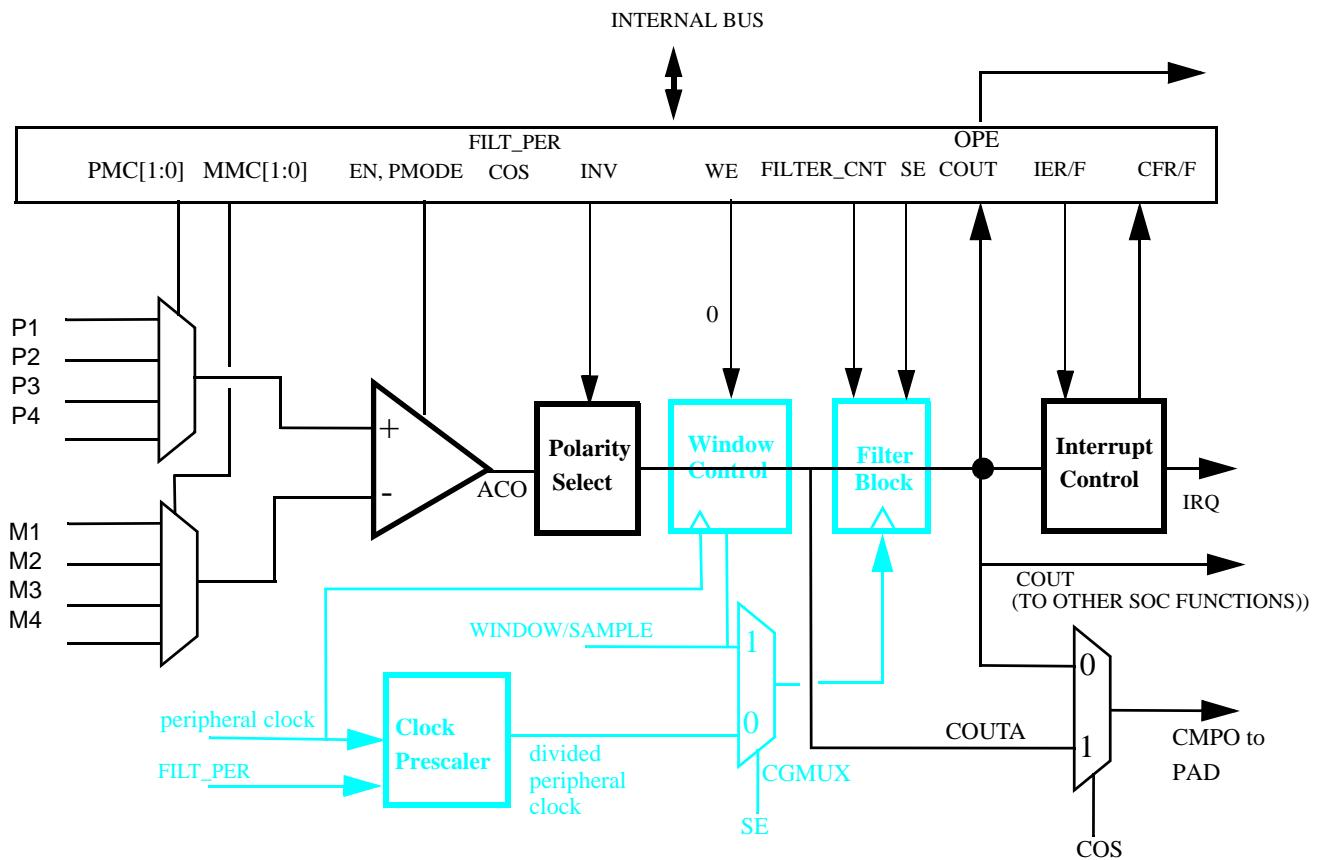
#### **NOTE**

Filtering and sampling settings should be changed only after setting SE=0 and FILTER\_CNT=0x0. This has the effect of resetting the filter to a known state.

#### **13.6.1.1 Disabled Mode (# 1)**

In disabled mode, the analog comparator is non-functional and consumes no power. The output of the analog comparator block (ACO) is zero in this mode. ColdFire V1 and DSC devices can further reduce power consumed in the digital block by disabling the peripheral clock to the comparator logic. This is usually a function of the System Integration Module, or SIM.

### 13.6.1.2 Continuous Mode (#s 2A & 2B)

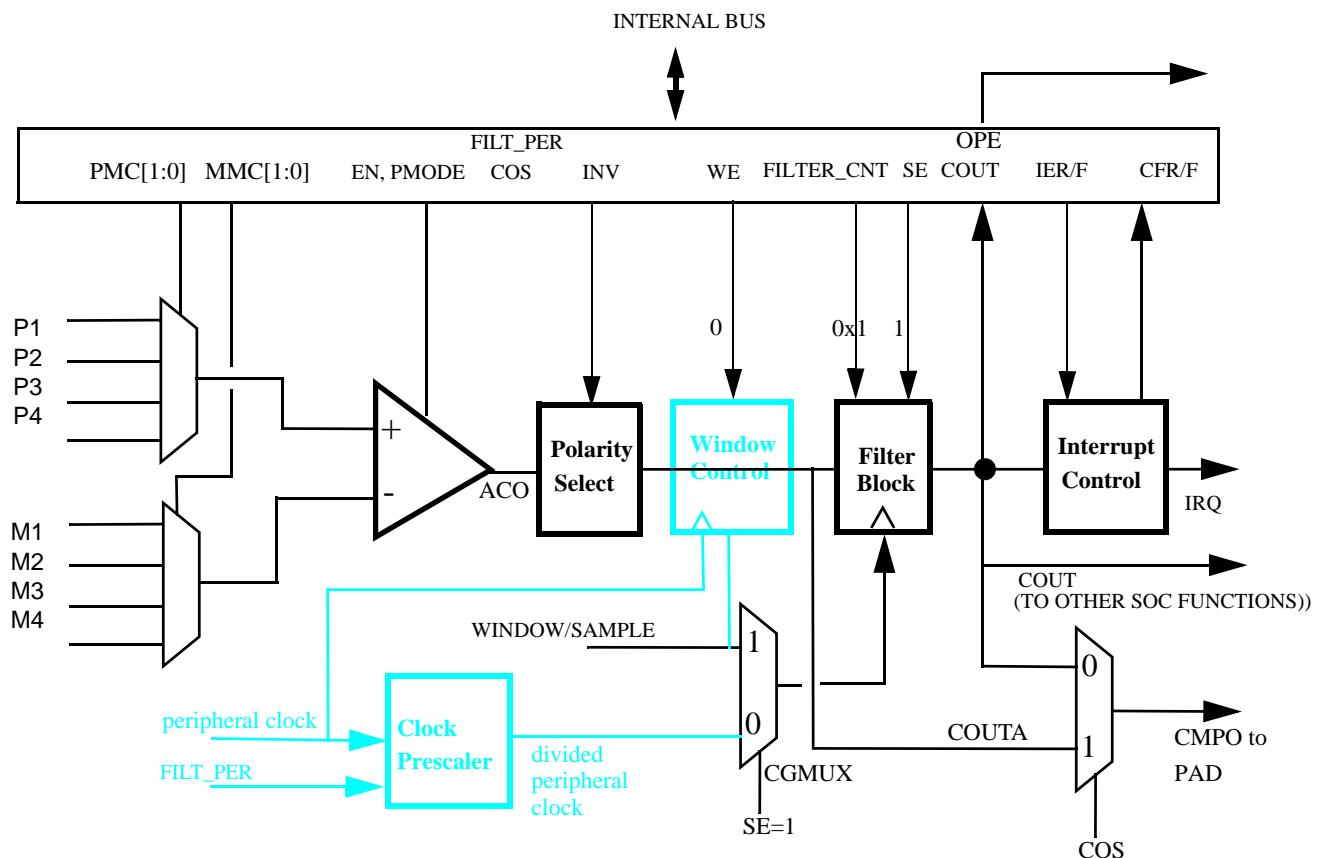


**Figure 13-6. Comparator Operation in Continuous Mode**

The analog comparator block is powered and active. ACO may be optionally inverted, but is not subject to external sampling or filtering. Both Window Control and Filter Blocks are completely bypassed. HSCMPxSCR[COUT] is updated continuously. The path from comparator inputs pins to output pin is operating in combinational (unclocked) mode. COUT and COUTA are identical.

See [Figure 13-4](#) for control configurations which result in disabling the Filter Block.

### 13.6.1.3 Sampled, Non-Filtered Mode (#s 3A & 3B)



**Figure 13-7. Sampled, Non-Filtered (# 3A): Sampling point externally driven**

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational (unclocked). Windowing Control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the Filter Block clock input.

The only difference in operation between [Figure 13-7](#) and [Figure 13-8](#) is in how the clock to the Filter Block is derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode.

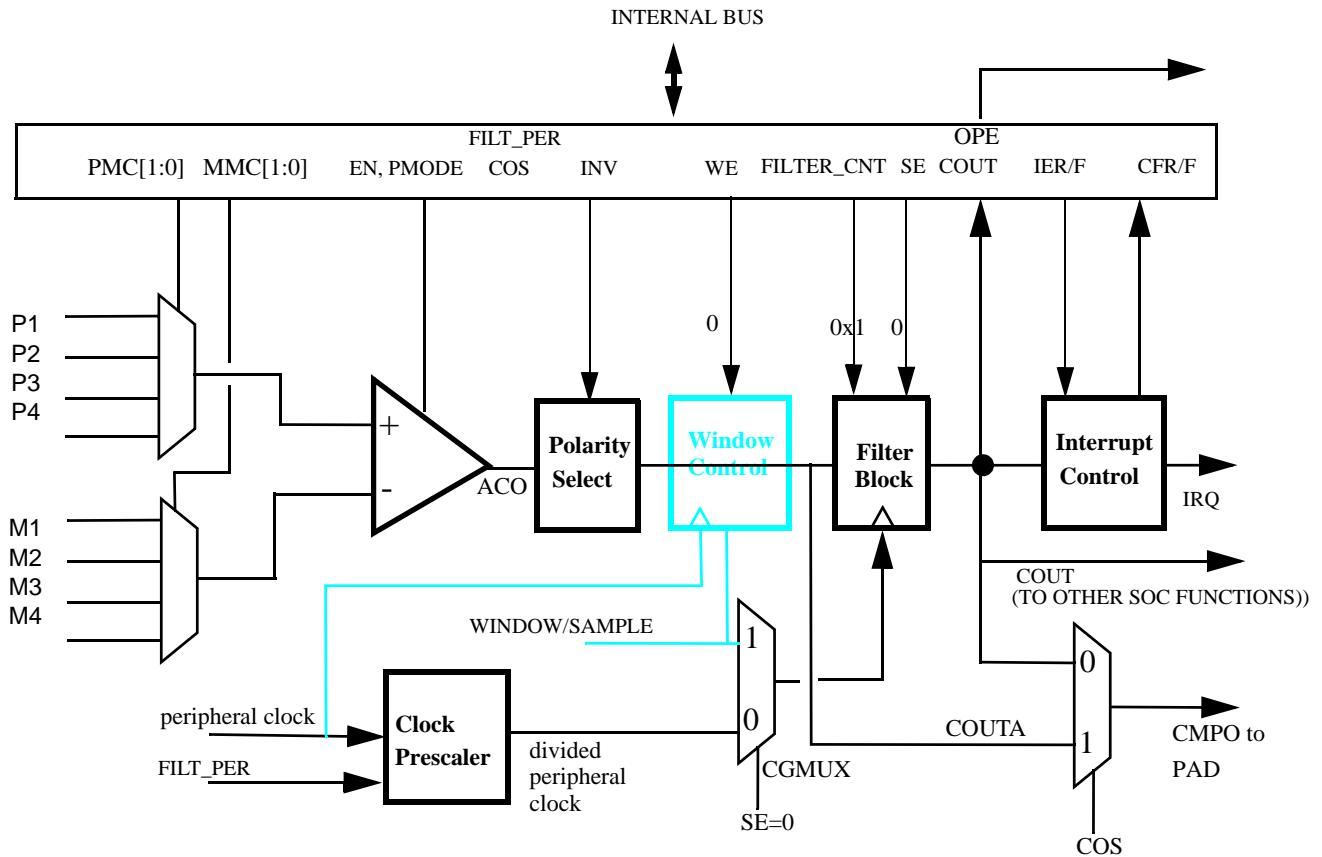
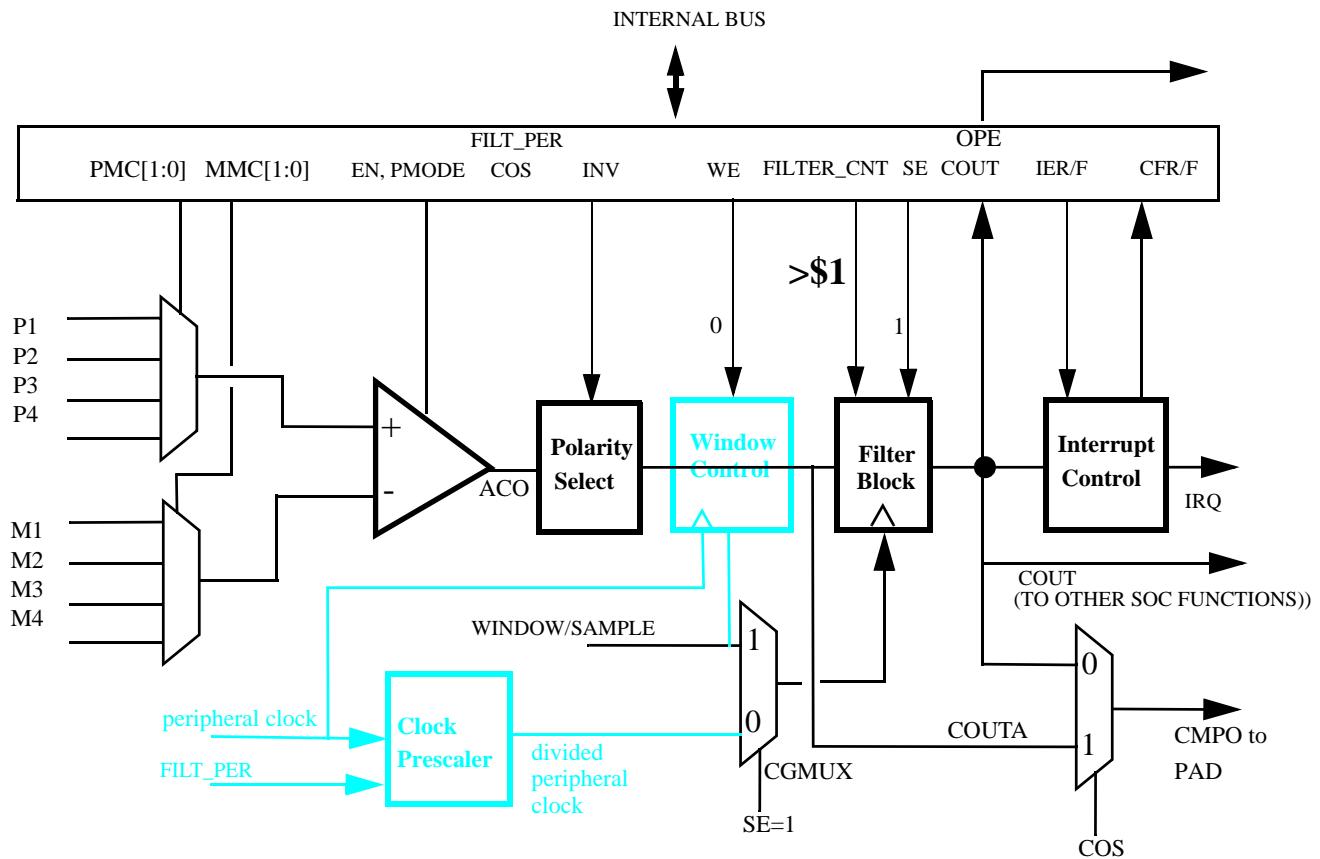


Figure 13-8. Sampled, Non-Filtered (# 3B): Sampling interval internally derived

### 13.6.1.4 Sampled, Filtered Mode (#s 4A & 4B)

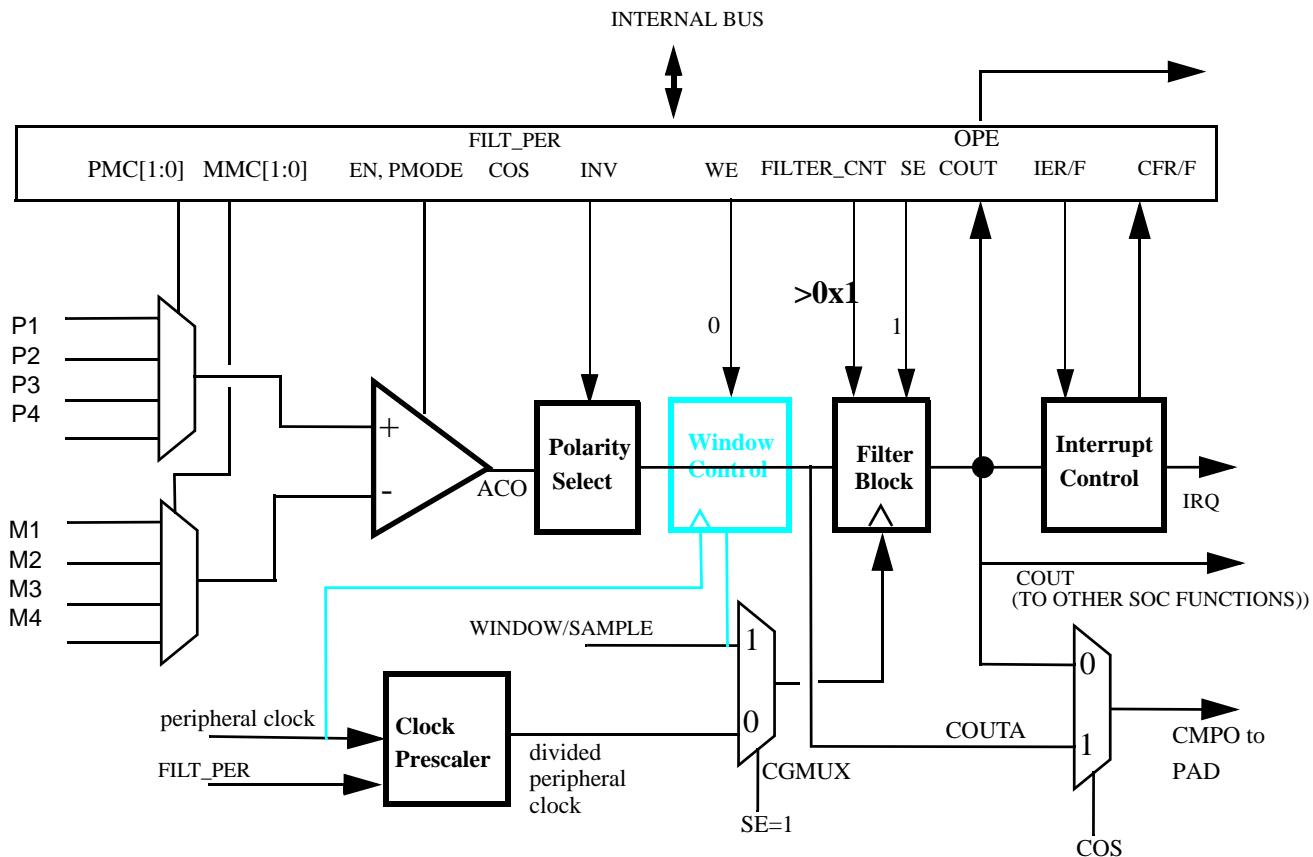
In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational (unclocked). Windowing Control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the Filter Block clock input.

The only difference in operation between [Figure 13-7](#) (Sampled , Non-Filtered # 3A) and [Figure 13-9](#) (Sampled, Filtered # 4A) is that FILTER\_CNT is now greater than 1, which activates filter operation.



**Figure 13-9. Sampled, Filtered (# 4A): Sampling point externally driven**

The only difference in operation between [Figure 13-10](#) (Sampled , Non-Filtered # 3B) and [Figure 13-10](#) (Sampled, Filtered # 4B) is that FILTER\_CNT is now greater than 1, which activates filter operation.



**Figure 13-10. Sampled, Filtered (# 4B): Sampling point internally driven**

### 13.6.1.5 Windowed Mode (#s 5A & 5B)

Figure 13-11 illustrates comparator operation in the windowed mode, ignoring latency of the analog comparator, polarity select and Window Control block. It also assumes that the Polarity Select is set to “non-inverting”. Note that the analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one peripheral clock cycle plus the combinational path delay through the comparator and polarity select logic.

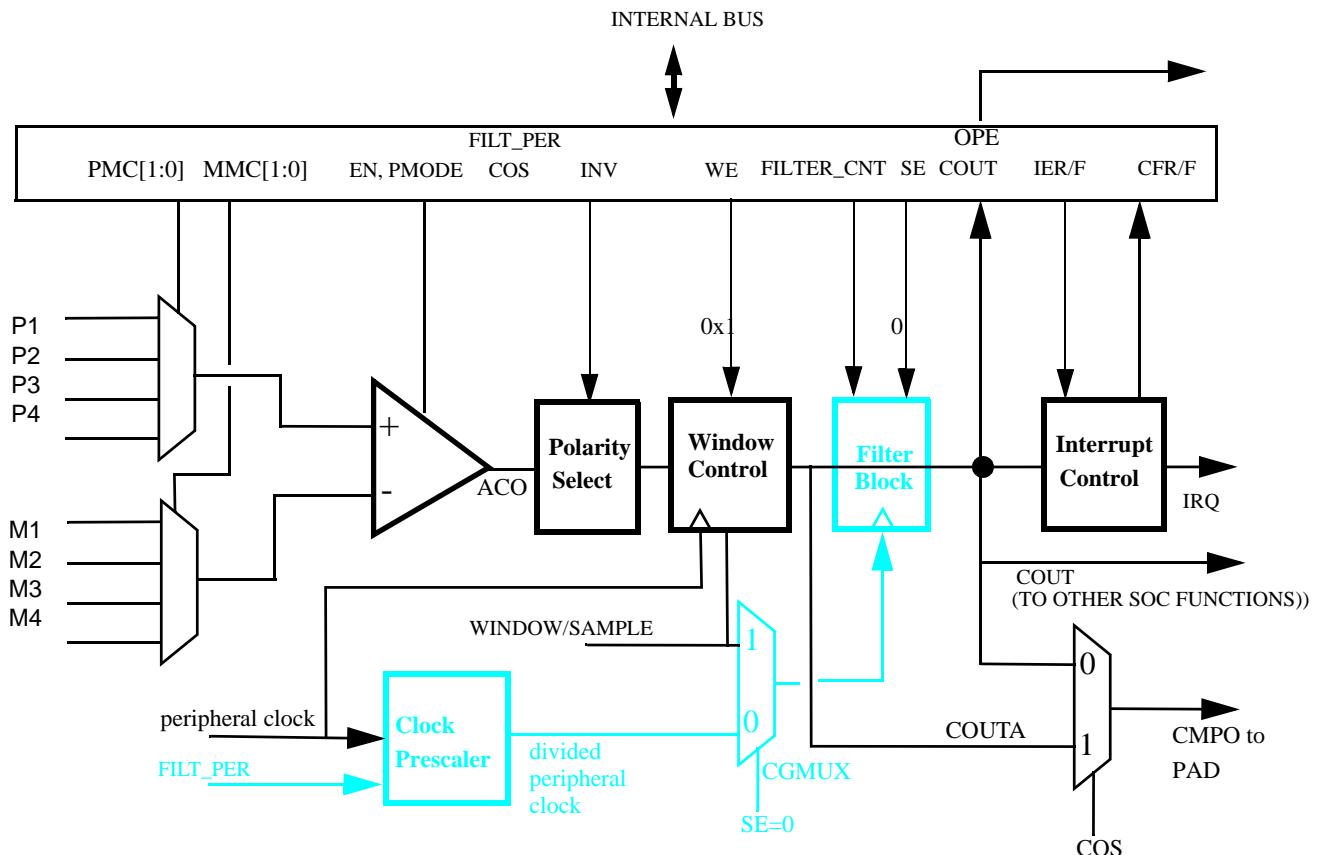
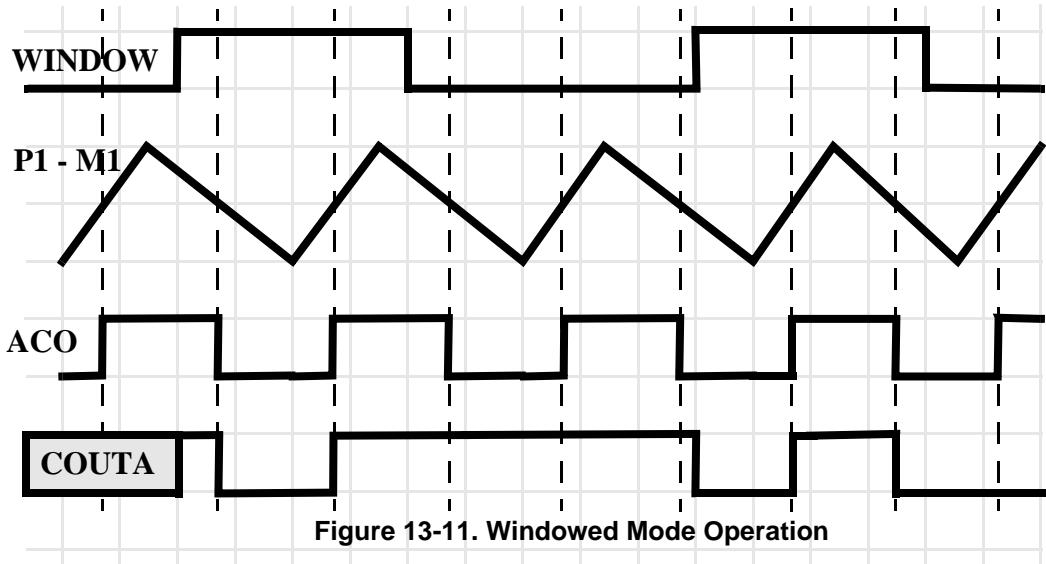


Figure 13-12. Windowed Mode

See [Figure 13-4](#) for control configurations which result in disabling the Filter Block.

When any windowed mode is active, COUTA is clocked by the peripheral clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

### 13.6.1.6 Windowed/Resampled Mode (# 6)

Figure 13-13 uses the same input stimulus shown in Figure 13-11, and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows. Again, prop delays and latency is ignored for clarity's sake. This example was generated solely to demonstrate operation of the comparator in windowing / resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

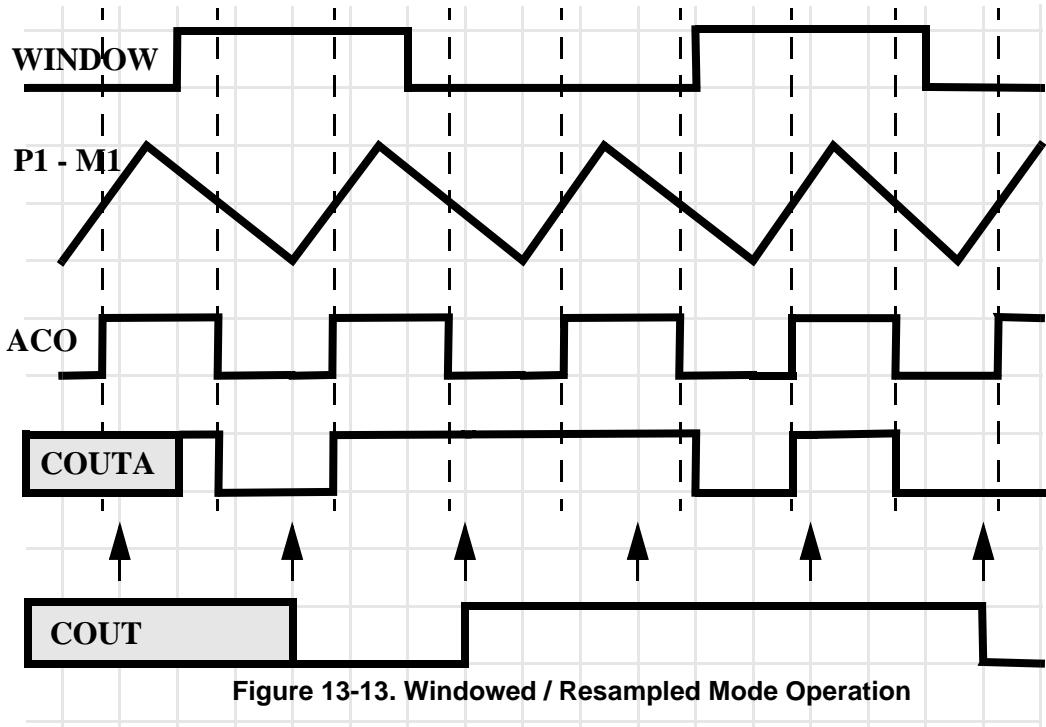


Figure 13-13. Windowed / Resampled Mode Operation

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FILT\_PER and the peripheral clock rate. Configuration for this mode is virtually identical to that for the Windowed/Filtered Mode shown in the next section. The only difference is that the value of FILTER\_CNT must be exactly one.

### 13.6.1.7 Windowed/Filtered Mode (#7)

This is the most complex mode of operation for the comparator block, as it utilizes both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 1 peripheral clock synchronization in the window function + ((FILT\_CNT X FILT\_PER) + 1) X peripheral clock for the filter function.

When any windowed mode is active, COUTA is clocked by the peripheral clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

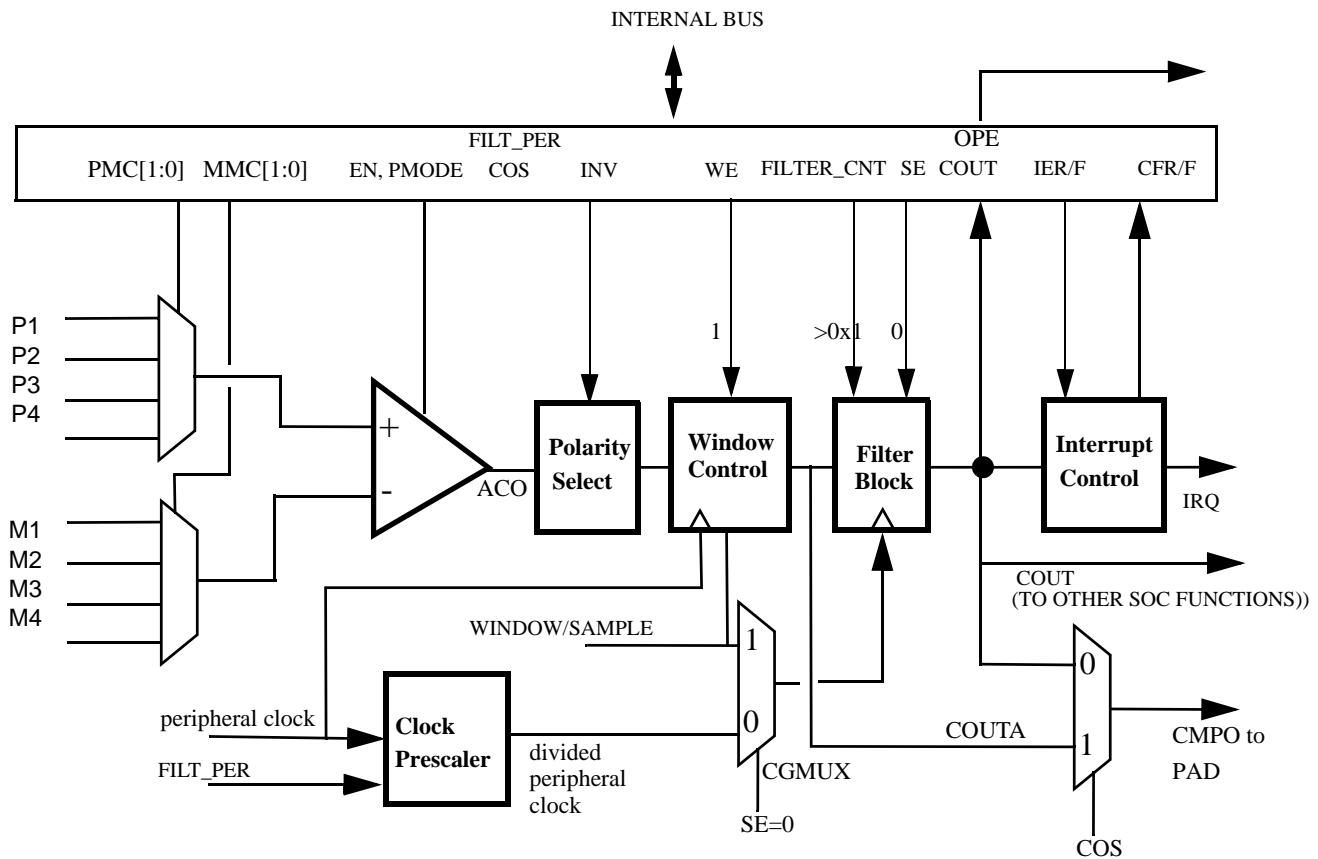


Figure 13-14. Windowed/Filtered Mode

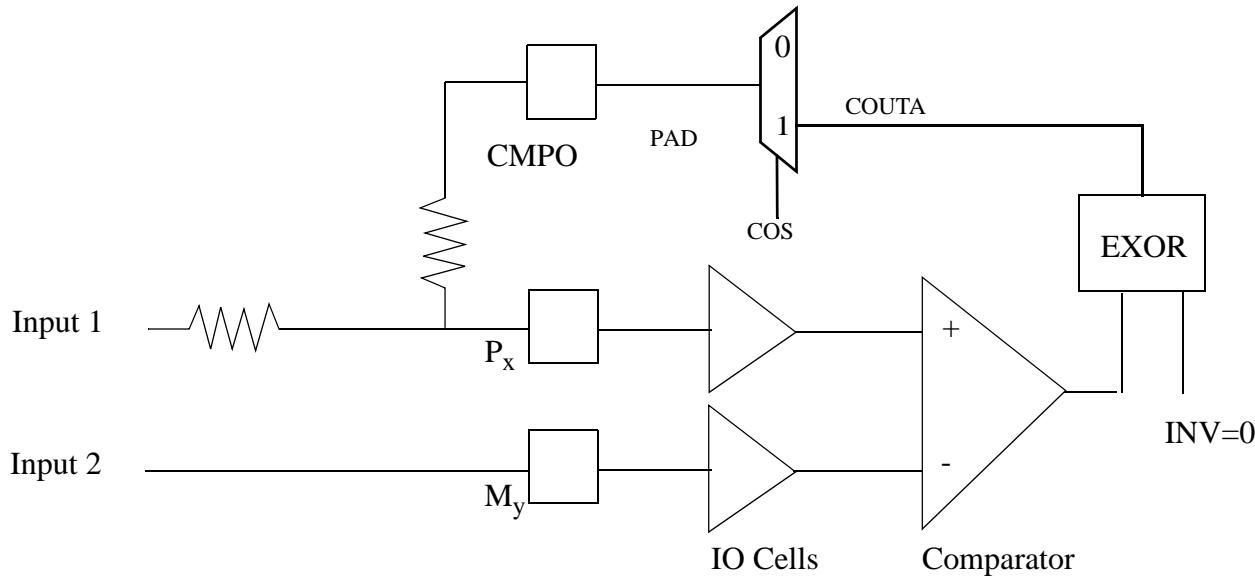
### 13.6.2 Hysteresis

The following diagram illustrates implementation of an external hysteresis resistor bridge between the asynchronous comparator output and the positive (+) input of the Comparator. Since positive feedback is required, INV must be set to 0 when the hysteresis resistor bridge is added to the positive (+) input of the Comparator. INV must be set to 1 when the hysteresis resistor bridge is added to the negative (-) input of the Comparator.

The option of adding an external resistor bridge for the purpose of adding hysteresis to the Comparator and the amount of hysteresis depends on the user's individual requirements. Hysteresis can be important in some system designs. In the absence of hysteresis, the continuous comparison of nearly identical analog inputs may add noise and waste power by generating high-frequency oscillations at CMPO.

If external hysteresis is added to the comparator, the bridge must be designed to consider other issues than simply how much hysteresis is needed. The resistor values must be sufficiently high so that they do not cause the drive strength of the digital output driver in the CMPO IO cell to be exceeded. Also, if any digital

function other than CMPO must operate on the CMPO pad in the presence of such a bridge, the resistor values must be sufficiently high so that the IO cell can function appropriately in its digital role.



**Figure 13-15. External Hysteresis Circuit**

### 13.6.3 Startup and Operation

A typical startup sequence is as follows.

The time required to stabilize COUT will be the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function and filter. Power on delay of the comparators are available from data sheets. The windowing function has a maximum of 1 peripheral bus clock period delay. Filter delay is specified in [Section 13.6.4, “Low Pass Filter”](#).

Interrupts should be disabled during power up, recovery from partial-power-down, and while re-enabling the module. Failure to do so could result in spurious interrupts.

During operation, the propagation delay of the selected data paths must always be considered. It can take many peripheral bus clock cycles for COUT and the CFR/CFF status bits to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT will initially be equal to zero until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic one.

### 13.6.4 Low Pass Filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by HSCMPxFPR[FILT\_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high frequency oscillations can be generated at COUTA when the selected N and P input voltages differ by less than the offset voltage of the differential comparator.

### 13.6.4.1 Enabling Filter Modes

Filter modes are enabled by setting HSCMPxCR0[FILTER\_CNT] greater than 0x1 and (setting HSCMPxFPR[FILT\_PER] to a non-zero value OR setting SE=1). If using the divided peripheral clock to drive the filter, it will take samples of COUTA every FILT\_PER peripheral bus cycles.

The filter output will be at logic zero when first initialized, and will subsequently change when FILT\_CNT consecutive samples all agree that the output value has changed. Said another way, COUT will be zero for some initial period, even when COUTA is at logic one.

Setting both SE and FILT\_PER to 0 disables the filter and eliminates switching current associated with the filtering process.

#### NOTE

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching FILTER\_CNT on the fly without this intermediate step can result in unexpected behavior.

If SE=1, the filter takes samples of COUTA on each positive transition of the SAMPLE input. The output state of the filter changes when FILT\_CNT consecutive samples all agree that the output value has changed.

### 13.6.4.2 Latency Issues

The FILT\_PER value (or SAMPLE period) should be set such that the sampling period is just larger the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT\_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT\_CNT power.

[Table 13-5](#) summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency is restarted each time an actual output transition is masked by noise.

The values of FILT\_PER (or SAMPLE period) and FILT\_CNT must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an

actual output change within the nominal latency is the probability of a correct sample raised to the FILT\_CNT power.

**Table 13-5. Comparator Sample/Filter Maximum Latencies**

<b>Mode #</b>	<b>EN</b>	<b>WE</b>	<b>S E</b>	<b>FILT_CNT</b>	<b>FILT_PER</b>	<b>Operation</b>	<b>Maximum Latency<sup>1</sup></b>
1	0	X	X	X	X	Disabled	N/A
2A	1	0	0	0x0	X	<b>Continuous Mode</b>	$T_{PD}$
2B	1	0	0	X	0x00		
3A	1	0	1	0x1	X	<b>Sampled, Non-Filtered mode</b>	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x1	> 0x0		$T_{PD} + (FILT\_PER \times T_{per}) + T_{per}$
4A	1	0	1	> 0x1	X	<b>Sampled, Filtered mode</b>	$T_{PD} + (FILT\_CNT \times T_{SAMPLE}) + T_{per}$
4B	1	0	0	> 0x1	> 0x0		$T_{PD} + (FILT\_CNT \times FILT\_PER \times T_{per}) + T_{per}$
5A	1	1	0	0x0	X	<b>Windowed mode</b>	$T_{PD} + T_{per}$
5B	1	1	0	X	0x00		$T_{PD} + T_{per}$
6	1	1	0	0x1	0x01 - 0xFF	<b>Windowed / Resampled mode</b>	$T_{PD} + (FILT\_PER \times T_{per}) + 2T_{per}$
7	1	1	0	> 0x1	0x01 - 0xFF	<b>Windowed / Filtered mode</b>	$T_{PD} + (FILT\_CNT \times FILT\_PER \times T_{per}) + 2T_{per}$

<sup>1</sup>  $T_{PD}$  represents the intrinsic delay of the analog component plus the polarity select logic.  $T_{SAMPLE}$  is the clock period of the external sample clock.  $T_{per}$  is the period of the peripheral bus clock.

## 13.7 Interrupts

The HSCMP module is capable of generating an interrupt on either the rising or falling edge of the comparator output (or both). The interrupt request is asserted when both HSCMPxSCR[IER] bit and HSCMPxSCR[CFR] are set. It is also asserted when both HSCMPxSCR[IEF] bit and HSCMPxSCR[CFF] are set. The interrupt is de-asserted by clearing either HSCMPxSCR[IER] or HSCMPxSCR[CFR] for a rising edge interrupt, or HSCMPxSCR[IEF] and HSCMPxSCR[CFF] for a falling edge interrupt.

## 13.8 Memory Map & Register Definitions

**Table 13-6. Module Memory Map**

<b>Address</b>	<b>Use</b>	<b>Access</b>
Base + 0x00	HSCMP control register 0 (HSCMPxCR0)	Read/Write
Base + 0x01	HSCMP control register 1 (HSCMPxCR1)	Read/Write
Base + 0x02	HSCMP filter period register (HSCMPxFPR)	Read/Write
Base + 0x03	HSCMP status & control register (HSCMPxSCR)	Read/Write
Base + 0x03	HSCMP pin control register (HCSMPxPCR)	Read/Write

### 13.8.1 Control Register 0 (HSCMPxCR0)

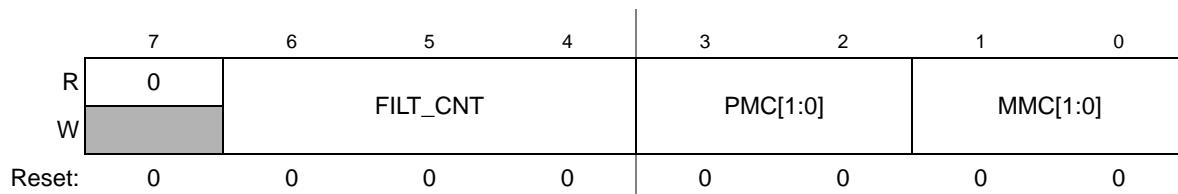


Figure 13-16. HSCMP Control Register 0 (HSCMPxCR0)

Table 13-7. HSCMPxCR0 Field Descriptions

Field	Description
7	Reserved, must be cleared.
6–4 FILT_CNT	Filter sample count. These bits represent the number of consecutive samples that must agree prior to the comparator output filter accepting a new output state. Filter programming and latency details are described in <a href="#">Section 13.6.4, "Low Pass Filter"</a> . 000 Filter is disabled. If SE = 1, COUT is cleared (this is not a legal state in <a href="#">Table 13-4</a> , and is not recommended). If SE = 0, COUT = COUTA. 001 1 consecutive samples must agree (comparator output is simply sampled) 010 2 consecutive samples must agree 011 3 consecutive samples must agree 100 4 consecutive samples must agree 101 5 consecutive samples must agree 110 6 consecutive samples must agree 111 7 consecutive samples must agree
3–2 PMC	Positive input mux control. Determines which input is selected for the positive input of the comparator. 00 P1 01 P2 10 P3 11 P4
1–0 MMC	Minus input mux control. Determines which input is selected for the minus input of the comparator. 00 M1 01 M2 10 M3 11 M4

### 13.8.2 Control Register 1 (HSCMPxCR1)

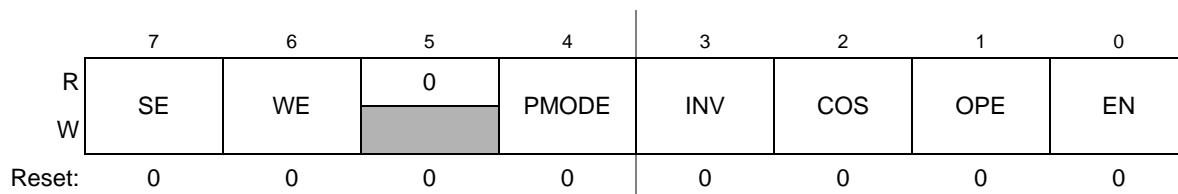
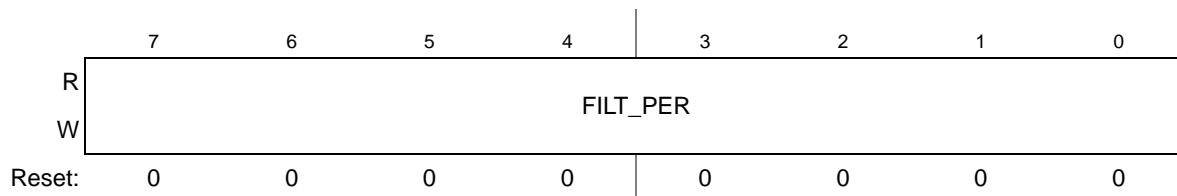


Figure 13-17. HSCMP Control Register 1 (HSCMPxCR1)

**Table 13-8. HSCMPxCR1 Field Descriptions**

Field	Description
7 SE	Sample enable 0 Sampling mode not selected 1 Sampling mode selected At most, one of SE and WE can be set at a time. If a write to this register attempts to set both, then SE will be set and WE will be cleared. However, writing ones to both bit locations should be avoided, as the “11” case is reserved and may change in future implementations.
6 WE	Windowing enable 0 Windowing mode not selected 1 Windowing mode selected At most, one of SE and WE can be set at a time. If a write to this register attempts to set both, then SE will be set and WE will be cleared. However, writing ones to both bit locations should be avoided, as the “11” case is reserved and may change in future implementations.
5	Reserved, must be cleared.
4 PMODE	Power mode select 0 Power savings mode 1 High speed comparison mode
3 INV	Comparator invert. This bit allows you to select the polarity of the comparator function. It is also driven to the COUT output (on both the device pin and as HSCMPxSCR[COUT]) when OPE=0. 0 Do NOT invert the comparator output 1 Invert the output of the analog comparator
2 COS	Comparator output select 0 Set CMPO to equal COUT (filtered comparator output) 1 Set CMPO to equal COUTA (unfiltered comparator output)
1 OPE	Comparator output pin enable. OPE is used to enable the comparator output to be placed onto the external pin, CMPO. 0 The comparator output (CMPO) is not available on the associated CMPO output pin. The pin is available for use by other on-chip functions. 1 The comparator output (CMPO) is driven out on associated CMPO output pin
0 EN	Comparator module enable. The EN bit enables the Analog Comparator Module. When the module is not enabled, it remains in the off state, and consumes no power. 0 Analog comparator disabled 1 Analog comparator enabled

### 13.8.3 Filter Period Register (HSCMPxFPR)

**Figure 13-18. HSCMP Filter Period Register (HSCMPxFPR)**

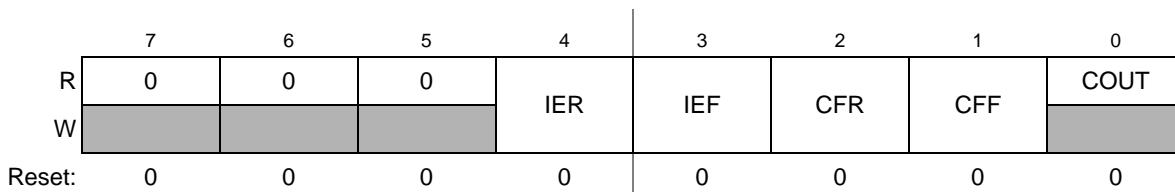
**Table 13-9. HSCMPxFPR Field Descriptions**

Field	Description
7–0 FILT_PER	Filter sample period. When HSCMPxCR1[SE] is cleared, this field specifies the sampling period, in peripheral clock cycles, of the comparator output filter. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details are described in <a href="#">Section 13.6.4, “Low Pass Filter”</a> . This field has no affect when HSCMPxCR1[SE] is equal to one. In that case, the external SAMPLE signal is used to determine the sampling period.

### 13.8.4 Status & Control Register (HSCMPxSCR)

#### NOTE

Interrupts must be disabled during power up, recovery from partial-power-down, and module enable sequences. The comparator output is not guaranteed to be stable until the module has had time to reach its stable operating point. See  $t_{ONEN}$ ,  $t_{ONPOR}$  and  $t_{ONPPD}$  in device data sheet.

**Figure 13-19. HSCMP Status and Control Register (HSCMPxSCR)****Table 13-10. HSCMPxSCR Field Descriptions**

Field	Description
7–5	Reserved, must be cleared.
4 IER	Comparator interrupt enable rising. The IER bit enables the rising edge detected interrupt from the ACM. When this bit is set, an interrupt will be asserted when the CFR bit is set. 0 Interrupt disabled 1 Interrupt enabled
3 IEF	Comparator interrupt enable falling. The IEF bit enables the rising edge detected interrupt from the ACM. When this bit is set, an interrupt will be asserted when the CFF bit is set. 0 Interrupt disabled 1 Interrupt enabled
2 CFR	Analog comparator flag rising. The CFR bit is set when a rising edge on COUT has been detected. The CFR bit is cleared by writing a logic one to the bit. 0 Rising edge on COMPO has not been detected 1 Rising edge on COUT has occurred
1 CFF	Analog comparator flag falling. The CFF bit is set when a falling edge on COUT has been detected. The CFF bit is cleared by writing a logic one to the bit. 0 A Falling edge on COUT has not been detected. 1 Falling edge on COUT has occurred.
0 COUT	Analog comparator output. Reading the COUT bit will return the current value of the analog comparator output. The register bit is reset to zero and will read as HSCMPxCR1[INV] when the Analog Comparator module is disabled (HSCMPxCR1[EN] = 0). Writes to this bit are ignored.

### 13.8.5 Pin Control Register (HSCMPxPCR)

The HSCMPxPCR is used to request static ownership of a given package pin by the HSCMP. The HSCMPxPCR must be programmed to enable HSCMP ownership of all input pins which may be required by an application. These fields are in addition to HSCMPxCR0[PMC] and HSCMPxCR0[MMC], which control on-the-fly switching between inputs.

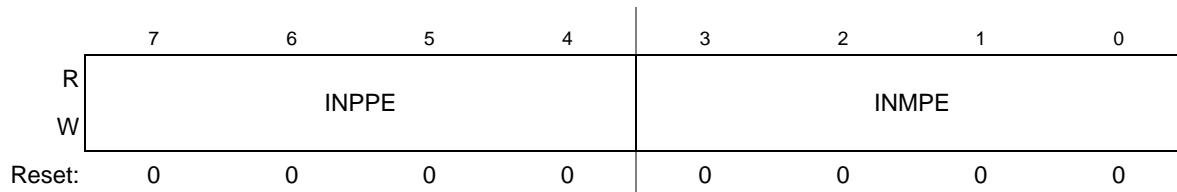


Figure 13-20. HSCMP Pin Control Register (HSCMPxPCR)

Table 13-11. HSCMPxPCR Field Descriptions

Field	Description
7–4 INPPE	Positive input pin enable xxx0 P1 is not required by the HSCMP xxx1 P1 is required by the HSCMP xx0x P2 is not required by the HSCMP xx1x P2 is required by the HSCMP x0xx P3 is not required by the HSCMP x1xx P3 is required by the HSCMP 0xxx P4 is not required by the HSCMP 1xxx P4 is required by the HSCMP
3–0 INMPE	Minus input pin enable xxx0 M1 is not required by the HSCMP xxx1 M1 is required by the HSCMP xx0x M2 is not required by the HSCMP xx1x M2 is required by the HSCMP x0xx M3 is not required by the HSCMP x1xx M3 is required by the HSCMP 0xxx M4 is not required by the HSCMP 1xxx M4 is required by the HSCMP

For example, if an application would like to cycle through P1–P4, comparing each in turn to M1. In this case, M2, M3, and M4 would be unused. Set HSCMPxPCR to 0xF1, HSCMPxCR0[MMC] to zero, and HSCMPxCR0[PMC] cycled through the values 00, 01, 10 and 11 via software.



# Chapter 14

## Internal Clock Source (S08ICSV3)

### 14.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock.

The ICSTRM and FTRIM bits are normally reset to the factory trim values on any reset. However, any reset that puts the device into BDM (a POR with the BKGD pin held low or a development tool setting SBDFR[BDFR]) results in the ICSTRM and FTRIM bits being set to values of 0x80 and 0. When debugging the MCU, the factory trim value can be used by copying the trim values from the Flash locations shown in [Table 4-4](#).

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived. The bus frequency will be one-half of the ICSOUT frequency.

[Figure 14-1](#) shows the MC9S08MP16 block diagram with the ICS highlighted.

### 14.2 Module Configuration

#### 14.2.1 System Clock Distribution

Refer to [Section 1.4, “System Clock Distribution”](#) for a detailed view of the distribution of clock sources throughout the MCU.

#### 14.2.2 External Oscillator

The ICS also controls a low power oscillator (XOSC) module to allow the use of an external clock or crystal/resonator as the external reference clock. This output, OSCOUT, can also be used as the real-time counter module (RTC) clock source.

#### 14.2.3 Stop3 Operation

When the internal reference is enabled in stop3 mode (IREFSTEN = 1), the voltage regulator must also be enabled in stop mode by setting the LVDE and LVDSE bits in the SPMSC1 register.

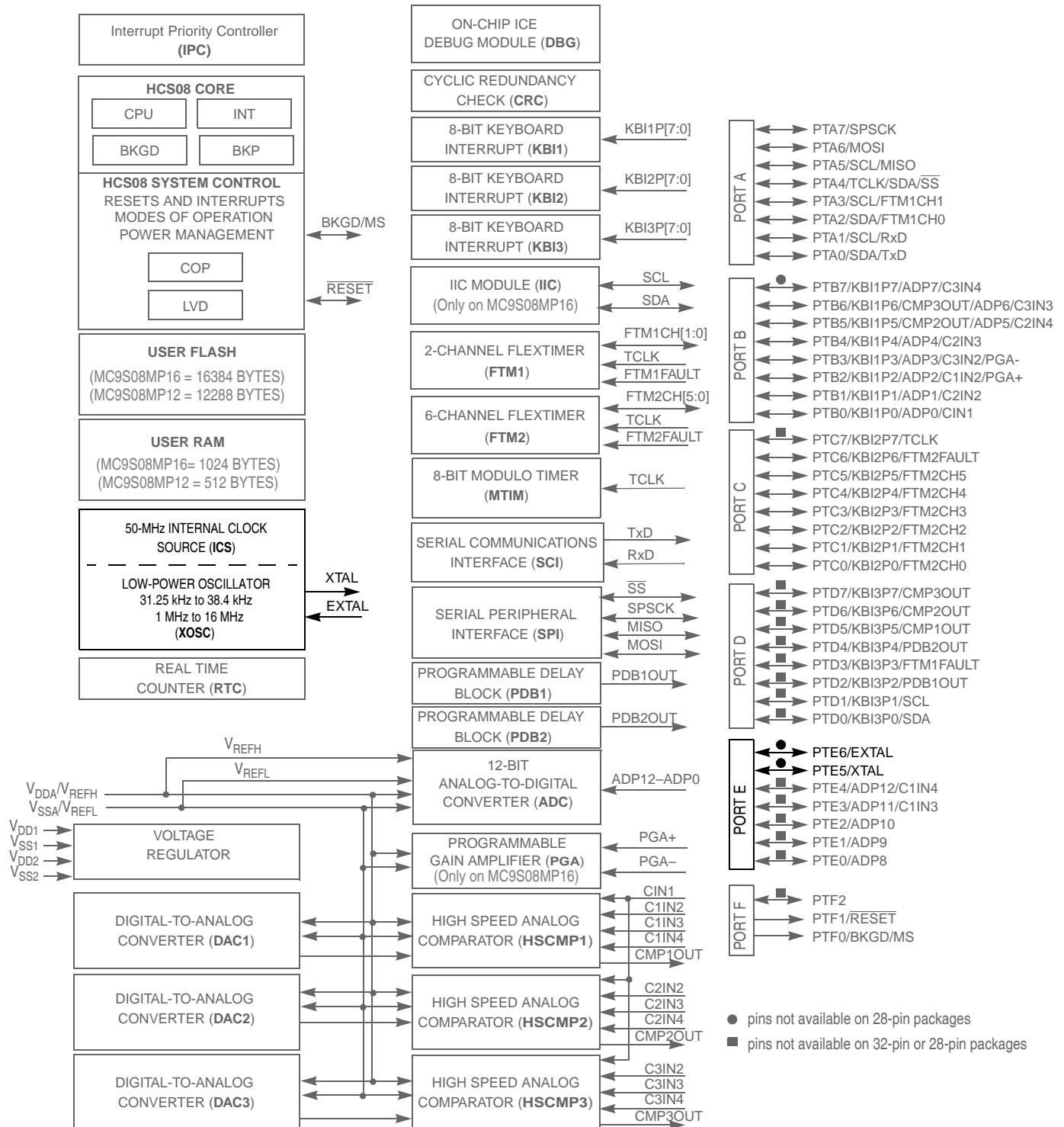


Figure 14-1. MC9S08MP16 Series Block Diagram Highlighting ICS Block and Pins

#### 14.2.4 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
- Internal or external reference clocks can be used to control the FLL
- Reference divider is provided for external clock
- Internal reference clock has 9 trim bits available
- Internal or external reference clocks can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  - 2-bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8
- Control signals for a low power oscillator clock generator (OSCOUT) as the ICS external reference clock are provided
  - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL Engaged Internal mode is automatically selected out of reset
- BDC clock is provided as a constant divide by 2 of the low range DCO output
- Three selectable digitally-controlled oscillators (DCO) optimized for different frequency ranges.
- Option to maximize output frequency for a 32768 Hz external reference clock source.

#### 14.2.5 Block Diagram

Figure 14-2 is the ICS block diagram.

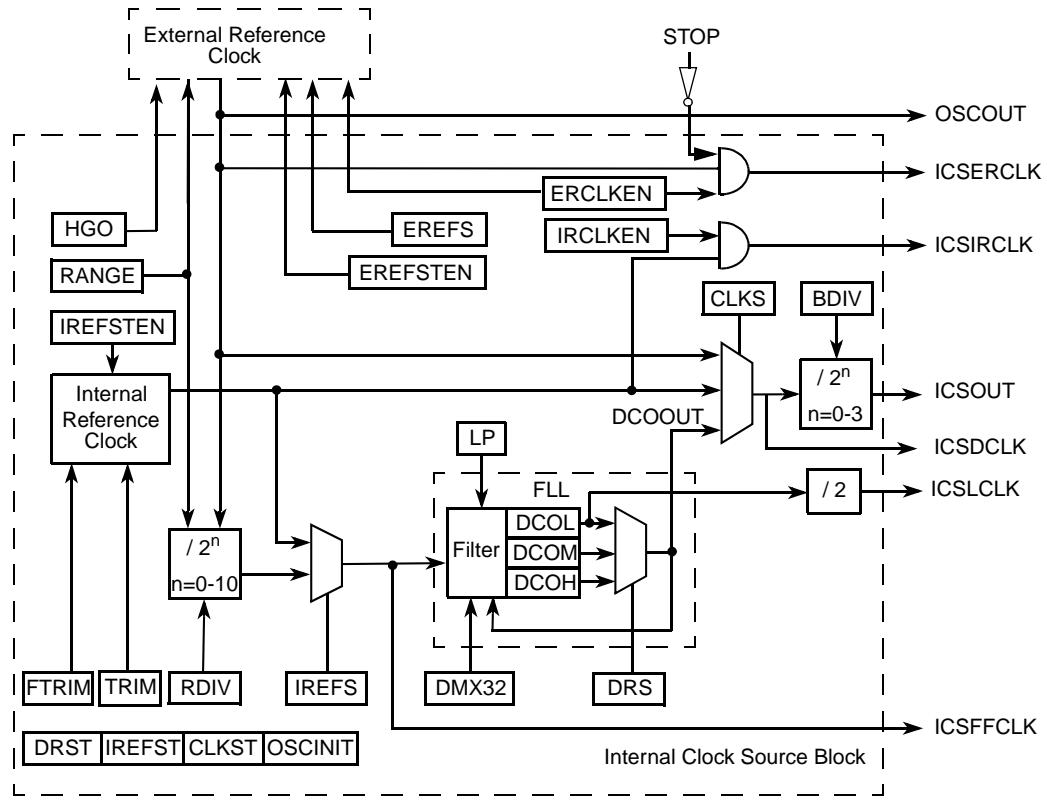


Figure 14-2. Internal Clock Source (ICS) Block Diagram

## 14.2.6 Modes of Operation

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

### 14.2.6.1 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.

### 14.2.6.2 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 source. The BDC clock is supplied from the FLL.

### 14.2.6.3 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.

#### 14.2.6.4 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.

#### 14.2.6.5 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 source. The BDC clock is supplied from the FLL.

#### 14.2.6.6 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 BDC clock is not available.

#### 14.2.6.7 Stop (STOP)

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

##### NOTE

The DCO frequency changes from the pre-stop value to its reset value and the FLL will need to re-acquire the lock before the frequency is stable. Timing sensitive operations should wait for the FLL acquisition time, tAcquire, before executing.

### 14.3 External Signal Description

There are no ICS signals that connect off chip.

### 14.4 Register Definition

[Figure 14-1](#) is a summary of ICS registers.

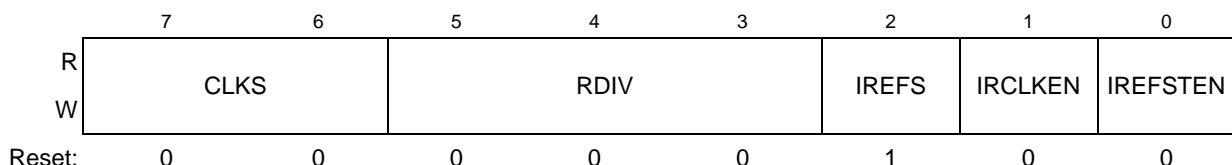
**Table 14-1. ICS Register Summary**

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

**Table 14-1. ICS Register Summary (continued)**

Name		7	6	5	4	3	2	1	0
ICSSC	R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
	W	DRS							

### 14.4.1 ICS Control Register 1 (ICSC1)

**Figure 14-3. ICS Control Register 1 (ICSC1)****Table 14-2. ICS Control Register 1 Field Descriptions**

Field	Description
7:6 CLKS	<b>Clock Source Select</b> — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 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	<b>Reference Divider</b> — Selects the amount to divide down the external reference clock. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. See <a href="#">Table 14-3</a> for the divide-by factors.
2 IREFS	<b>Internal Reference Select</b> — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected. 0 External reference clock selected.
1 IRCLKEN	<b>Internal Reference Clock Enable</b> — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active. 0 ICSIRCLK inactive.
0 IREFSTEN	<b>Internal Reference Stop Enable</b> — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set before entering stop. 0 Internal reference clock is disabled in stop.

**Table 14-3. Reference Divide Factor**

RDIV	RANGE=0	RANGE=1
0	1 <sup>1</sup>	32
1	2	64
2	4	128
3	8	256

**Table 14-3. Reference Divide Factor**

RDIV	RANGE=0	RANGE=1
4	16	512
5	32	1024
6	64	Reserved
7	128	Reserved

<sup>1</sup> Reset default

## 14.4.2 ICS Control Register 2 (ICSC2)

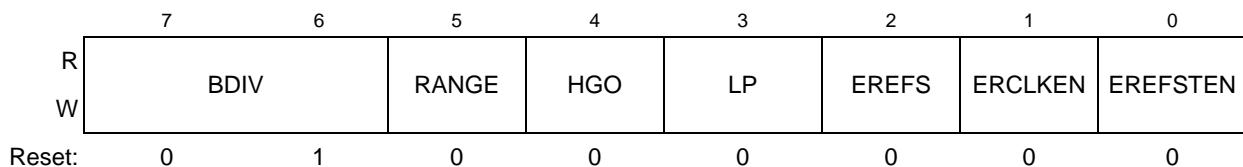
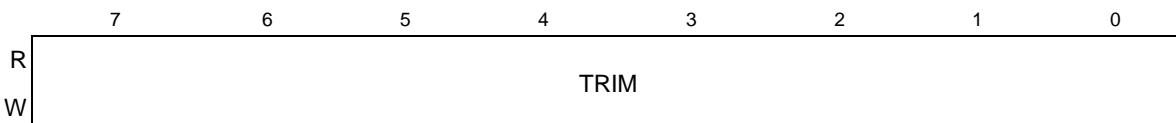


Figure 14-4. ICS Control Register 2 (ICSC2)

Table 14-4. ICS Control Register 2 Field Descriptions

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

## 14.4.3 ICS Trim Register (ICSTRM)



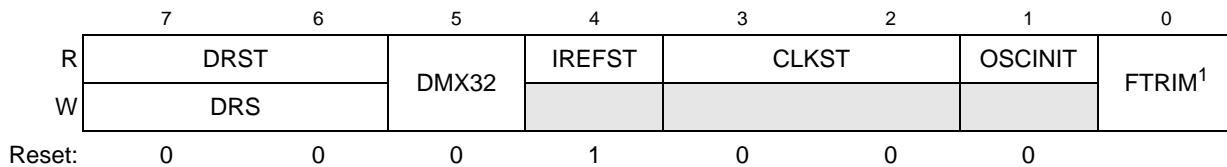
Reset: Note: TRIM is loaded during reset from a factory programmed location when not in BDM mode. If in a BDM mode, a default value of 0x80 is loaded.

Figure 14-5. ICS Trim Register (ICSTRM)

**Table 14-5. ICS Trim Register Field Descriptions**

Field	Description
7:0 TRIM	<b>ICS Trim Setting</b> — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (in other words, bit 1 adjusts 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.

#### 14.4.4 ICS Status and Control (ICSSC)

**Figure 14-6. ICS Status and Control Register (ICSSC)**

<sup>1</sup> FTRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, FTRIM gets loaded with a value of 1'b0.

**Table 14-6. ICS Status and Control Register Field Descriptions**

Field	Description
7-6 DRST DRS	<b>DCO Range Status</b> — The DRST read field indicates the current frequency range for the FLL output, DCOOUT. See <a href="#">Table 14-7</a> . The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. Writing the DRS bits to 2'b11 is ignored and the DRST bits remain with the current setting.  <b>DCO Range Select</b> — The DRS field selects the frequency range for the FLL output, DCOOUT. Writes to the DRS field while the LP bit is set are ignored. 00 Low range. 01 Mid range. 10 High range. 11 Reserved.
5 DMX32	<b>DCO Maximum frequency with 32.768 kHz reference</b> — The DMX32 bit controls whether or not the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. See <a href="#">Table 14-7</a> . 0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.
4 IREFST	<b>Internal Reference Status</b> — The IREFST bit indicates the current source for the reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains. 0 Source of reference clock is external clock. 1 Source of reference clock is internal clock.

**Table 14-6. ICS Status and Control Register Field Descriptions (continued)**

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

**Table 14-7. DCO frequency range<sup>1</sup>**

DRS	DMX32	Reference range	FLL factor	DCO range
00	0	31.25 - 39.0625 kHz	512	16 - 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	0	31.25 - 39.0625 kHz	1024	32 - 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	0	31.25 - 39.0625 kHz	1536	48 - 60 MHz
	1	32.768 kHz	1824	59.77 MHz
11		Reserved		

<sup>1</sup> The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.

## 14.5 Functional Description

### 14.5.1 Operational Modes

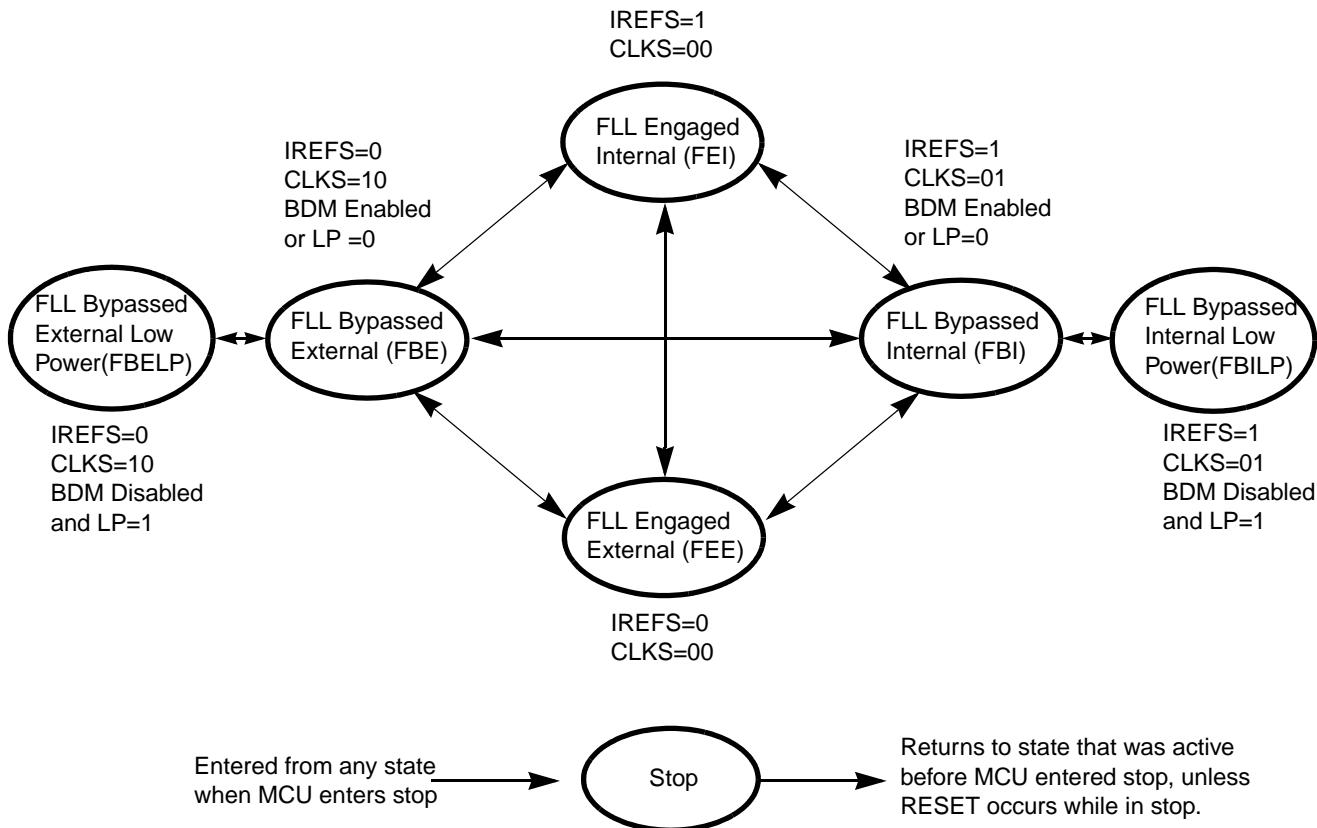


Figure 14-7. Clock Switching Modes

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

#### 14.5.1.1 FLL Engaged Internal (FEI)

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

- CLKS bits are written to 00.
- IREFS bit is written to 1.

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 locks the frequency to the FLL factor times the internal reference frequency. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

### 14.5.1.2 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 external 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 source. The FLL loop locks the frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

### 14.5.1.3 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 locks the FLL frequency to the FLL factor times the internal reference frequency. The ICSLCLK will be available for BDC communications, and the internal reference clock is enabled.

### 14.5.1.4 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.

### 14.5.1.5 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.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.
- 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 source. The FLL clock is controlled by the external reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits, so that the ICSLCLK will be available for BDC communications, and the external reference clock is enabled.

#### 14.5.1.6 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 source and the FLL is disabled. The ICSLCLK will not be available for BDC communications. The external reference clock source is enabled.

#### 14.5.1.7 Stop

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

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

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

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

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

#### 14.5.2 Mode Switching

The IREF bit can be changed at anytime, but the actual switch to the newly selected clock is shown by the IREFST bit. When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes, the FLL begins locking again after the switch is completed.

The CLKS bits can also be changed at anytime, but the actual switch to the newly selected clock is shown by the CLKST bits. If the newly selected clock is not available, the previous clock remains selected.

The DRS bits can be changed at anytime except when LP bit is 1. If the DRS bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the bus clock remains at the previous DCO range until the new DCO starts. When the new DCO starts the bus clock switches to it. After switching to the new DCO the FLL remains unlocked for several reference cycles. Once the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the DRST bits.

### 14.5.3 Bus Frequency Divider

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

### 14.5.4 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. The DRS bits can not be written while LP bit is 1.

However, in some applications it may be desirable to allow the FLL to be enabled and to lock for maximum accuracy before switching to an FLL engaged mode. To do this, write the LP bit to 0.

### 14.5.5 DCO Maximum Frequency with 32.768 kHz Oscillator

The FLL has an option to change the clock multiplier for the selected DCO range such that it results in the maximum bus frequency with a common 32.768 kHz crystal reference clock.

### 14.5.6 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal is presented as ICSIRCLK, which can be used as an additional clock source. To re-target the ICSIRCLK frequency, write a new value to the TRIM bits in the ICSTRM register to trim the period of the internal reference clock:

- Writing a larger value slows down the ICSIRCLK frequency.
- Writing a smaller value to the ICSTRM register speeds up the ICSIRCLK frequency.

The TRIM bits 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.

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 keeps 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 is uploaded to the ICSTRM register and ICS FTRIM register during any reset initialization. For finer precision, trim the internal oscillator in the application and set the FTRIM bit accordingly.

### 14.5.7 External Reference Clock

The ICS module supports an external reference clock with frequencies between 31.25 kHz to 40 MHz in all modes. When the ERCLKEN is set, the external reference clock signal is presented as ICSERCLK, which can be used as an additional clock source in run mode. When IREFS = 1, the external reference clock is not 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 support (see the [Device Overview](#) chapter).

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

### 14.5.8 Fixed Frequency Clock

The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source. ICSFFCLK frequency must be no more than 1/4 of the ICSOUT frequency to be valid.

### 14.5.9 Local Clock

The ICS presents the low range DCO output clock divided by two as ICSLCLK for use as a clock source for BDC communications. ICSLCLK is not available in FLL bypassed internal low power (FBILP) and FLL bypassed external low power (FBELP) modes.



# Chapter 15

## Inter-Integrated Circuit (S08IICV3)

### 15.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at up to 400 kbps with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

Figure 15-1 shows the MC9S08MP16 Series block diagram with the IIC module highlighted.

#### NOTE

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

### 15.2 Module Configuration

This section provides device-specific information for configuring the IIC modules on MC9S08MP16 Series.

#### 15.2.1 IIC Clock Gating

The bus clock to the IIC can be gated on and off using the IIC bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the IIC bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 15.2.2 Pin Repositioning

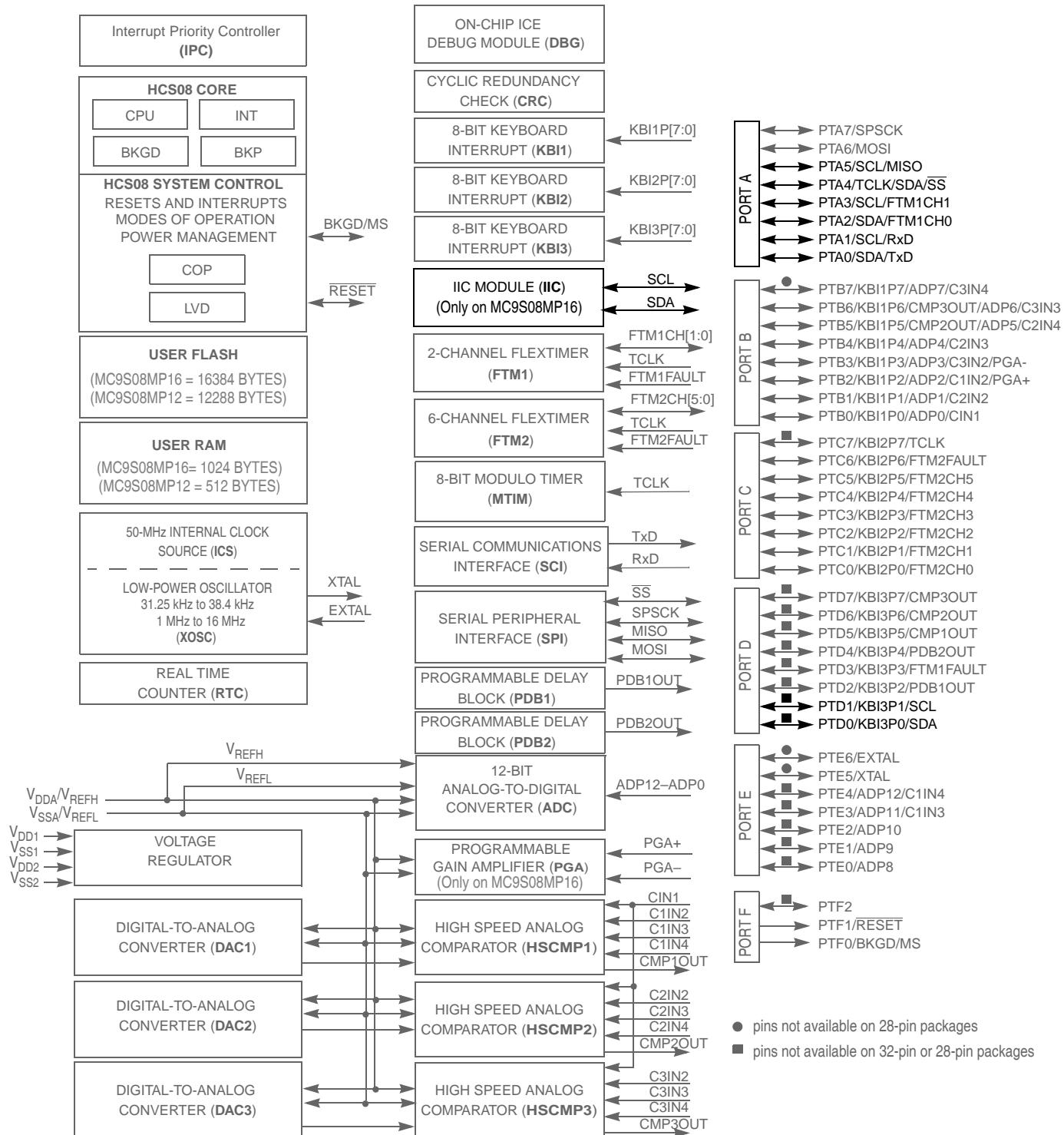
The IIC module pins, SDA and SCL can be repositioned under software control using IICPS in SOPT2 as shown in [Table 15-1](#). IICPS in SOPT2 selects which general-purpose I/O ports are associated with IIC operation.

**Table 15-1. IIC Position Options**

IICPS in SOPT2	Port Pin for SDA	Port Pin for SCL
00 (default)	PTD0	PTD1
01	PTA0	PTA1
10	PTA2	PTA3
11	PTA4	PTA5

### 15.2.3 IIC Pin Filtering

The IIC pin filtering on MC9S08MP16 Series devices is controlled by three bits (FLT[2:0]) in IICFLT. The clock source to the digital filter is at the ICSOUT frequency, providing a resolution for IIC pin filtering of 1/2 bus cycle.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

**Figure 15-1. MC9S08MP16 Series Block Diagram Highlighting IIC Block and Pins**

**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: IICC
  - 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](#)

**Module Initialization (Master)**

1. Write: IICF
  - to set the IIC baud rate (example provided in this chapter)
2. Write: IICC
  - 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](#)
5. Write: IICC
  - to enable TX
6. Write: IICC
  - to enable MST (master mode)
7. Write: IICD
  - with the address of the target slave. (The LSB of this byte will determine whether the communication is master receive or transmit.)

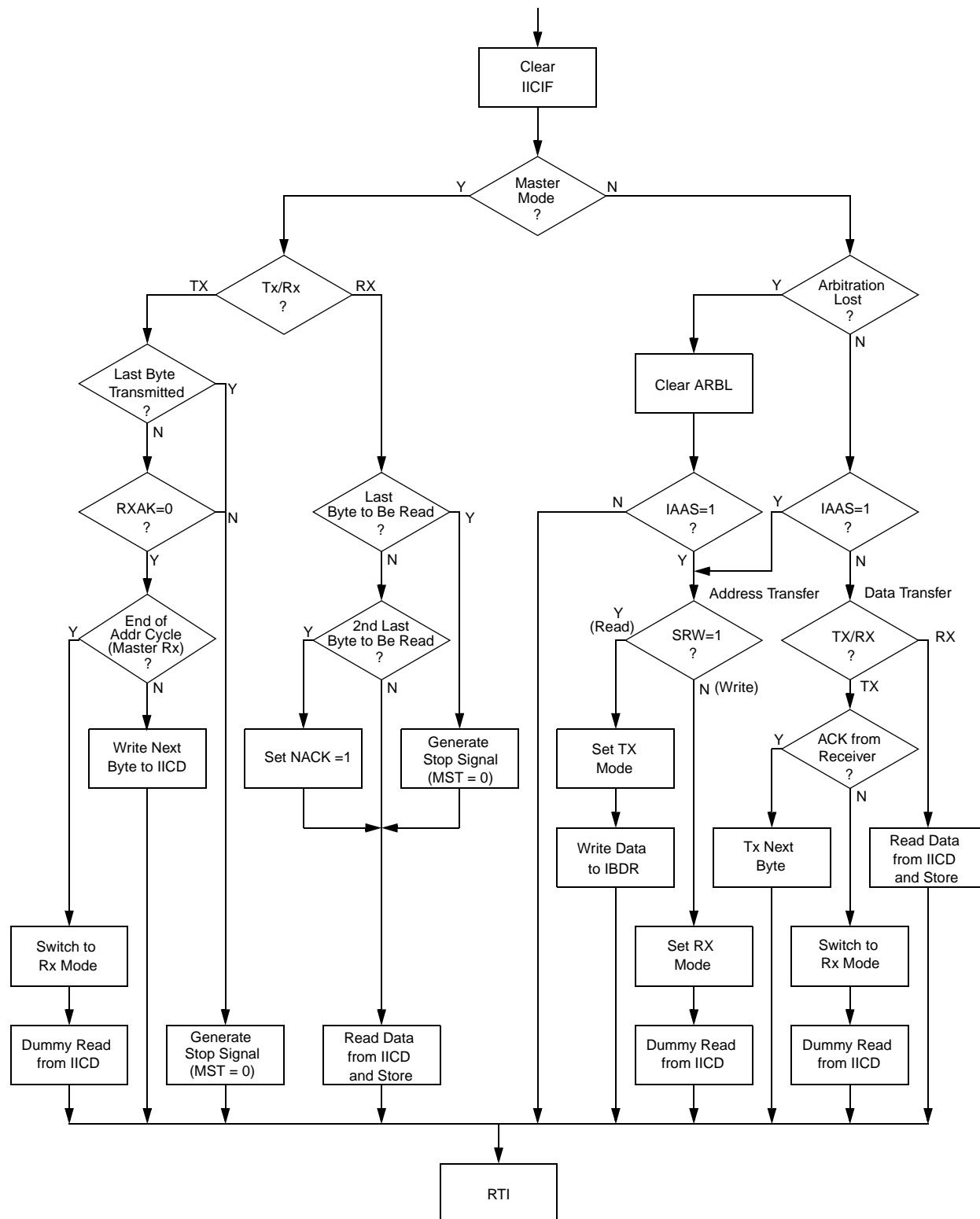
**Module Use**

The routine shown in [Figure](#) can handle both master and slave IIC operations. For slave operation, an incoming IIC message that contains the proper address will begin IIC communication. For master operation, communication must be initiated by writing to the IICD register.

**Register Model**

IICA	AD[7:1] 0								
Address to which the module will respond when addressed as a slave (in slave mode)									
IICF	MULT	ICR							
Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER))									
IICC	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 15-2. IIC Module Quick Start**



### Typical IIC Interrupt Routine

## 15.2.4 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/detection
- Acknowledge bit generation/detection
- Bus busy detection
- General call recognition
- 10-bit address extension
- Support System Management Bus Specification(SMBus), version2
- Programmable glitch input filter

## 15.2.5 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 will continue 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 will reset the register contents.

## 15.2.6 Block Diagram

Figure 15-3 is a block diagram of the IIC.

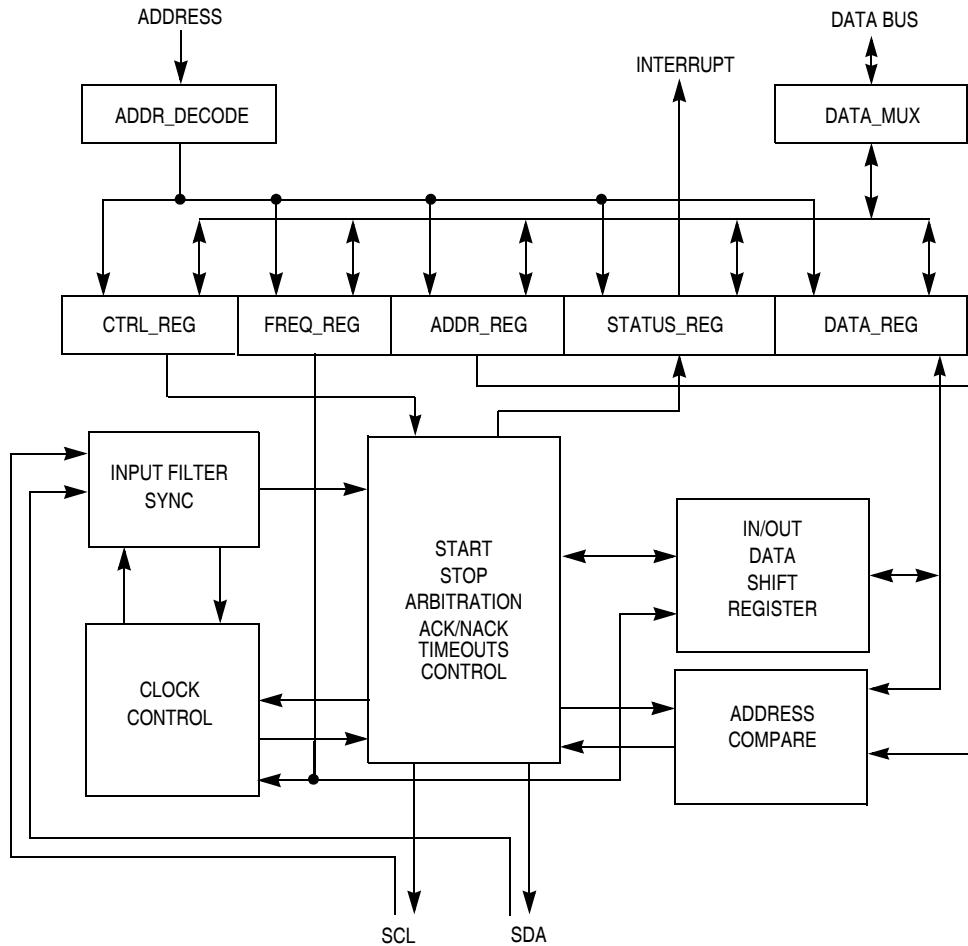


Figure 15-3. IIC Functional Block Diagram

## 15.3 External Signal Description

This section describes each user-accessible pin signal.

### 15.3.1 SCL — Serial Clock Line

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

### 15.3.2 SDA — Serial Data Line

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

## 15.4 Register Definition

### 15.4.1 Module Memory Map

The IIC has ten 8-bit registers. The base address of the module is hardware programmable. The IIC register map is fixed and begins at the module's base address. [Table 15-2](#) summarizes the IIC module's address space. The following section describes the bit-level arrangement and functionality of each register.

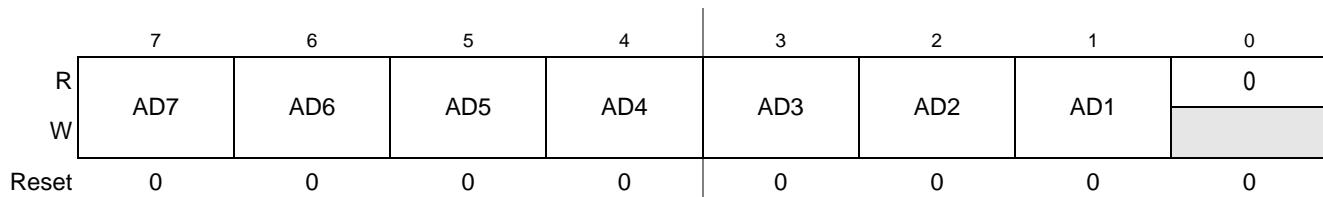
**Table 15-2. Module Memory Map**

Address	Use	Access
Base + \$0000	IIC Address Register 1 (IICA1)	read/write
Base + \$0001	IIC Frequency Divider Register (IICF)	read/write
Base + \$0002	IIC Control Register 1 (IICC1)	read/write
Base + \$0003	IIC Status Register (IICS)	read
Base + \$0004	IIC Data IO Register (IICD)	read/write
Base + \$0005	IIC Control Register 2 (IICC2)	read/write
Base + \$0006	SMBUS IIC Control and Status Register (IICSMB)	read/write
Base + \$0007	IIC Address Register 2 (IICA2)	read/write
Base + \$0008	IIC SCL Low Time Out Register High (IICSLTH)	read/write
Base + \$0009	IIC SCL Low Time Out Register Low (IICSLTL)	read/write
Base + \$000A	IIC input programmable filter (IICFLT)	read/write

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

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet 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.

### 15.4.2 IIC Address Register 1 (IICA1)



**Figure 15-4. IIC Address Register 1 (IICA1)**

**Table 15-3. IICA1 Field Descriptions**

Field	Description
7:1 AD[7:1]	<b>Slave Address 1</b> —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.

### 15.4.3 IIC Frequency Divider Register (IICF)

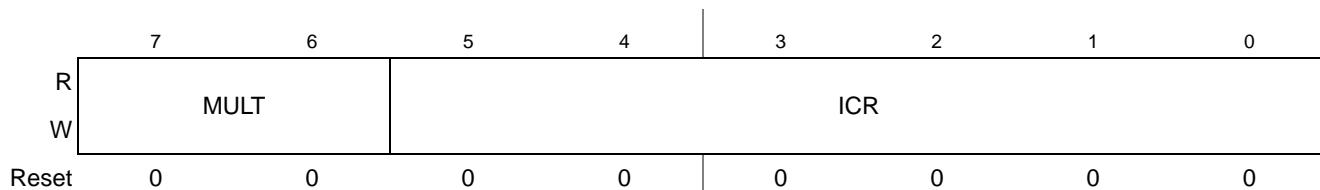


Figure 15-5. IIC Frequency Divider Register (IICF)

Table 15-4. IICF Field Descriptions

Field	Description
7:6 MULT	<b>IIC Multiplier Factor</b> — The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate 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	<b>IIC Clock Rate</b> — The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits are used to determine the IIC baud rate, the SDA hold time, the SCL Start hold time and the SCL Stop hold time. <a href="#">Table 15-5</a> provides the SCL divider and hold values for corresponding values of the ICR.  The SCL divider multiplied by multiplier factor mul is used to generate IIC baud rate.  $\text{IIC baud rate} = \text{bus speed (Hz)} / (\text{mul} * \text{SCL divider}) \quad \text{Eqn. 15-1}$ SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data).  $\text{SDA hold time} = \text{bus period (s)} * \text{mul} * \text{SDA hold value} \quad \text{Eqn. 15-2}$ 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).  $\text{SCL Start hold time} = \text{bus period (s)} * \text{mul} * \text{SCL Start hold value} \quad \text{Eqn. 15-3}$ SCL Stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA (IIC data) while SCL is high (Stop condition).  $\text{SCL Stop hold time} = \text{bus period (s)} * \text{mul} * \text{SCL Stop hold value} \quad \text{Eqn. 15-4}$

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

MULT	ICR	Hold times ( $\mu\text{s}$ )		
		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 15-5. IIC Divider and Hold Values**

<b>ICR (hex)</b>	<b>SCL Divider</b>	<b>SDA Hold Value</b>	<b>SCL Hold (Start) Value</b>	<b>SDA Hold (Stop) Value</b>
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

<b>ICR (hex)</b>	<b>SCL Divider</b>	<b>SDA Hold Value</b>	<b>SCL Hold (Start) Value</b>	<b>SCL Hold (Stop) Value</b>
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

### 15.4.4 IIC Control Register (IICC1)

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

Figure 15-6. IIC Control Register (IICC1)

Table 15-6. IICC1 Field Descriptions

Field	Description
7 IICEN	<b>IIC Enable</b> — The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled. 1 IIC is enabled.
6 IICIE	<b>IIC Interrupt Enable</b> — The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled. 1 IIC interrupt request enabled.
5 MST	<b>Master Mode Select</b> — When the MST bit is changed from a 0 to a 1, 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	<b>Transmit Mode Select</b> — 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 will always be 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	<b>Transmit Acknowledge Enable</b> — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. There are two conditions will effect NAK/ACK generation. If FACK (fast NACK/ACK) is cleared, 0 An acknowledge signal will be sent out to the bus on the following receiving data byte. 1 No acknowledge signal response is sent to the bus on the following receiving data byte. If FASK bit is set. no ACK or NACK is sent out after receiving one data byte until this TXAK bit is written 0 An acknowledge signal will be sent out to the bus on the current receiving data byte 1 No acknowledge signal response is sent to the bus on the current receiving data byte Note: SCL is held to low until TXAK is written.
2 RSTA (Write Only read always 0)	<b>Repeat START</b> — Writing a 1 to this bit will generate a repeated START condition provided it is the current master. Attempting a repeat at the wrong time will result in loss of arbitration. 0 No repeat start detected in bus operation. 1 Repeat start generated.

## 15.4.5 IIC Status Register (IICS)

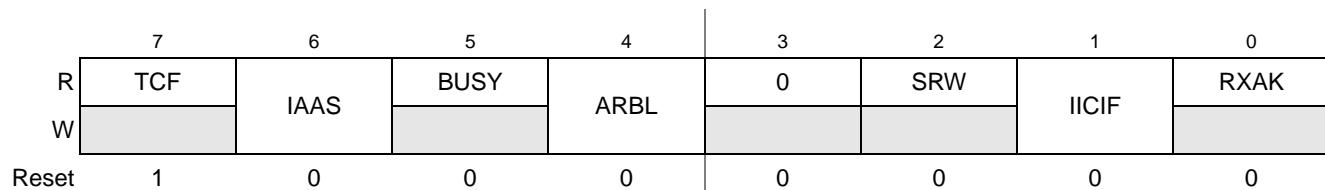


Figure 15-7. IIC Status Register (IICS)

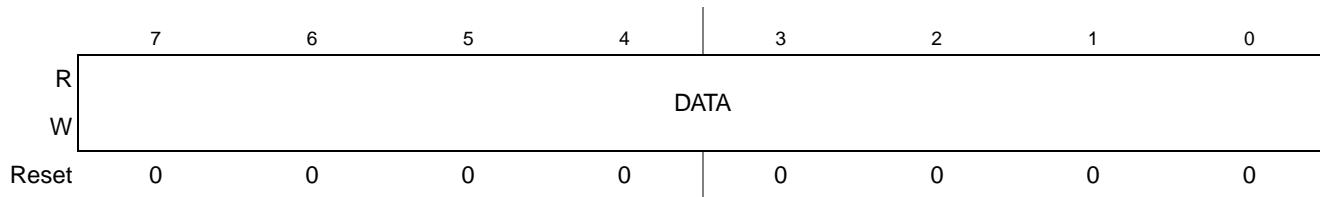
Table 15-7. IICS Field Descriptions

Field	Description
7 TCF	<b>Transfer Complete Flag</b> — This bit is set on the completion of a byte and acknowledge bit transfer. Note that 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	<b>Addressed as a Slave</b> — The IAAS bit is set when one of the following conditions is met 1) When the calling address matches the programmed slave address, 2) If the GCAEN bit is set and a general call is received. 3) If SIICAEN bit is set, when the calling address matches the 2nd programmed slave address 4) If ALERTEN bit is set and SMBus alert response address is received  This bit is set before ACK bit. The CPU needs to check the SRW bit and set TX/RX bit accordingly. Writing the IICC1 register with any value clears this bit. 0 Not addressed. 1 Addressed as a slave.
5 BUSY	<b>Bus Busy</b> — 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	<b>Arbitration Lost</b> — 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	<b>Slave Read/Write</b> — 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.

**Table 15-7. IICS Field Descriptions (continued)**

Field	Description
1 IICIF	<b>IIC Interrupt Flag</b> — 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: <ul style="list-style-type: none"> <li>• One byte transfer including ACK/NACK bit completes if FACK = 0</li> <li>• One byte transfer including ACK/NACK bit completes if FACK = 1 and this byte is an address byte</li> <li>• One byte transfer excluding ACK/NACK bit completes if FACK = 1 and this byte is a data byte. an ACK or NACK is sent out on the bus by writing 0 or 1 to TXAK after this bit is set.</li> <li>• Match of slave addresses to calling address (Primary Slave address, General Call address, Alert Response address, and Second Slave address) (Received address is stored in data register)</li> <li>• Arbitration lost</li> <li>• Timeouts in SMBus mode except high timeout</li> </ul> 0 No interrupt pending. 1 Interrupt pending.
0 RXAK	<b>Receive Acknowledge</b> — 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.

### 15.4.6 IIC Data I/O Register (IICD)

**Figure 15-8. IIC Data I/O Register (IICD)****Table 15-8. IICD Field Descriptions**

Field	Description
7:0 DATA	<b>Data</b> — 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.

Note that 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, then reading the IICD will not initiate the receive.

Reading the IICD will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IICD does not reflect every byte that is 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 (Start bit) or assertion of RSTA bit (repeated Start ) is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/W bit (in position bit 0).

### 15.4.7 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

Figure 15-9. IIC Control Register (IICC2)

Table 15-9. IICC2 Field Descriptions

Field	Description
7 GCAEN	<b>General Call Address Enable</b> — The GCAEN bit enables or disables general call address. 0 General call address is disabled 1 General call address is enabled.
6 ADEXT	<b>Address Extension</b> — 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]	<b>Slave Address</b> — 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.

## 15.4.8 IIC SMBus Control and Status Register (IICSMB)

	7	6	5	4		3	2	1	0
R W	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF	0	0	0
Reset	0	0	0	0	0	0	0	0	0

NOTE Table 15-10. IICSMB Field Descriptions

Field	Description
7 FACK	<b>Fast NACK/ACK enable</b> — For SMBus Packet Error Checking, CPU should be able to issue an ACK or NACK according to the result of receiving data byte. 0 ACK or NACK will be sent out on the following receiving data byte. 1 Writing an 0 to TXAK after receiving data byte will generate an ACK; Writing an 1 to TXAK after receiving data byte will generate a NACK
6 ALERTEN reserved	<b>SMBus Alert Response Address Enable</b> — The ALERTEN bit enables or disable SMBus alert response address. 0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled.
5 SIICAEN	<b>Second IIC Address Enable</b> — The SIICAEN bit enables or disable SMBus device default address. 0 IIC Address Register 2 matching is disabled. 1 IIC Address Register 2 matching is enabled.
4 TCKSEL	Time Out Counter Clock Select— This bit selects the clock sources of Time Out Counter 0 Time Out Counter counts at bus/64 frequency 1 Time Out Counter counts at the bus frequency
3 SLTF	SCL Low Timeout Flag — This read-only bit is set to logic 1 when IICSLT loaded non zero value (LoValue) and a SCL Low Time Out occurs. This bit is cleared by software, by writing a logic 1 to it 0 No LOW TIME OUT occurs. 1 A LOW TIME OUT occurs. Note: LOW TIME OUT function is disabled when IIC SCL LOW TIMER OUT register is set to zero
2 SHTF	SCL High Timeout Flag — This read-only bit is set to logic 1 when SCL and SDA are held high more than clock * LoValue/512, which indicates the Bus Free. This bit is cleared automatically. 0 No HIGH TIMEOUT occurs. 1 An HIGH TIMEOUT occurs.

### NOTE

A master can assume that the bus is free if it detects that the clock and data signals have been high for greater than THIGH,MAX, however, the SHTF will rise in bus transmission process but bus idle state.

When TCKSEL=1 there is no meaning to monitor SHTF since the bus speed is too high to match the protocol of SMBus.

### 15.4.9 IIC Address Register 2 (IICA2)

	7	6	5	4		3	2	1	0
R W	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0	
Reset	1	1	0	0	0	0	1	0	

Field	Description
7:1 SAD[7:1]	<b>SMBUs Address</b> — The AD[7:1] field contains the slave address to be used by the SMBus. This field is used on the device default address or other related address

### 15.4.10 IIC SCL Low Time Out Register High (IICSLTH)

	7	6	5	4		3	2	1	0
R W	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8	
Reset	0	0	0	0	0	0	0	0	

Field	Description
7:0 SSLT[15:8]	The value in this register is the most significant byte of SCL low time out value that determines the time-out period of SCL low.

### 15.4.11 IIC SCL LowTime Out register Low (IICSLTL)

	7	6	5	4		3	2	1	0
R W	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0	
Reset	0	0	0	0	0	0	0	0	

Field	Description
7:0 SSLT[7:0]	The value in this register is the least significant byte of SCL low time out value that determines the time-out period of SCL low..

### 15.4.12 IIC Programmable Input Glitch Filter (IICFLT)

	7	6	5	4		3	2	1	0
R W	0	0	0	0	FLT3	FLT2	FLT1	FLT0	
Reset	0	0	0	0	0	0	0	0	

**Table 15-11. IICFLT Field Descriptions**

Field	Description
3:0 FLT	<p>IIC programmable filter factor. Contains the programming controls for the width of glitch (in terms of bus clock cycles) the filter should absorb; in other words, the filter does not let glitches less than or equal to this width setting pass.</p> <p>The width of this field is device-specific. See the <a href="#">Introduction</a> section of this chapter for the width available for this device.</p> <ul style="list-style-type: none"> <li>0000 No filter/bypass</li> <li>0001 Filter glitches up to width of 1 (half) IPBUS clock cycle</li> <li>0010 Filter glitches up to width of 2 (half) IPBUS clock cycles</li> <li>0011 Filter glitches up to width of 3 (half) IPBUS clock cycles</li> <li>0100 Filter glitches up to width of 4 (half) IPBUS clock cycles</li> <li>0101 Filter glitches up to width of 5 (half) IPBUS clock cycles</li> <li>0110 Filter glitches up to width of 6 (half) IPBUS clock cycles</li> <li>0111 Filter glitches up to width of 7 (half) IPBUS clock cycles</li> <li>1000 Filter glitches up to width of 8 (half) IPBUS clock cycles</li> <li>1001 Filter glitches up to width of 9 (half) IPBUS clock cycles</li> <li>1010 Filter glitches up to width of 10 (half) IPBUS clock cycles</li> <li>1011 Filter glitches up to width of 11 (half) IPBUS clock cycles</li> <li>1100 Filter glitches up to width of 12 (half) IPBUS clock cycles</li> <li>1101 Filter glitches up to width of 13 (half) IPBUS clock cycles</li> <li>1110 Filter glitches up to width of 14 (half) IPBUS clock cycles</li> <li>1111 Filter glitches up to width of 15 (half) IPBUS clock cycles</li> </ul>

## 15.5 Functional Description

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

### 15.5.1 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 15-10](#).

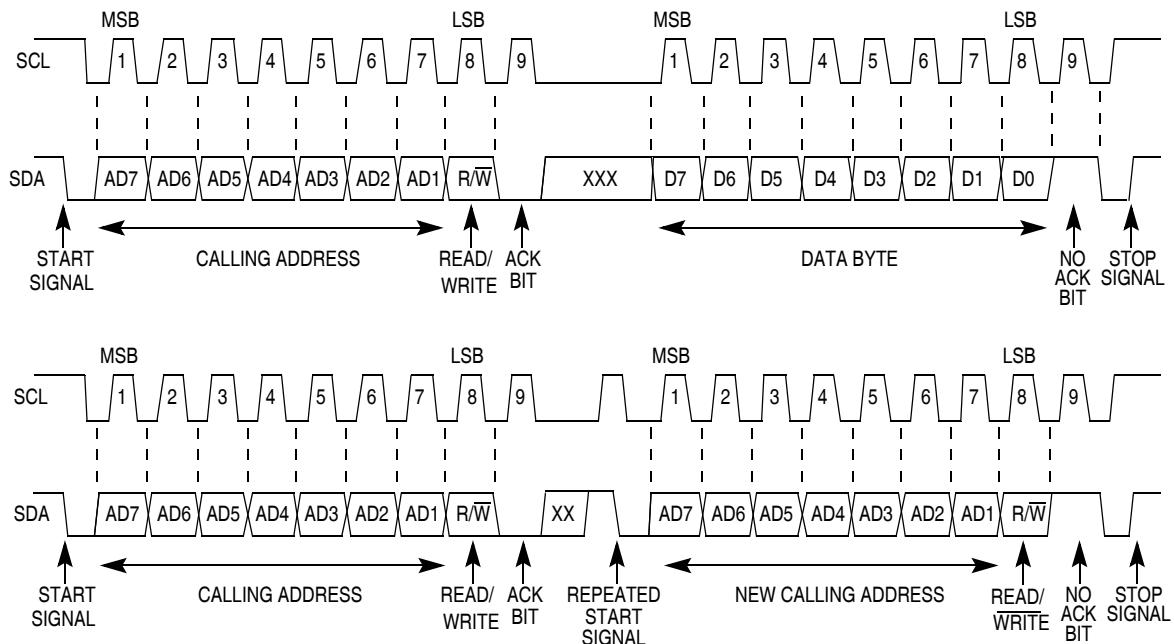


Figure 15-10. IIC Bus Transmission Signals

### 15.5.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 15-10, 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.

### 15.5.1.2 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/W bit. The R/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 will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 15-10).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is 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 will revert to slave mode and operate correctly even if it is being addressed by another master.

### 15.5.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/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 15-10](#). 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 9th 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.

### 15.5.1.4 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 15-10](#)).

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

### 15.5.1.5 Repeated START Signal

As shown in [Figure 15-10](#), 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.

### 15.5.1.6 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.

### 15.5.1.7 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 15-11). 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.

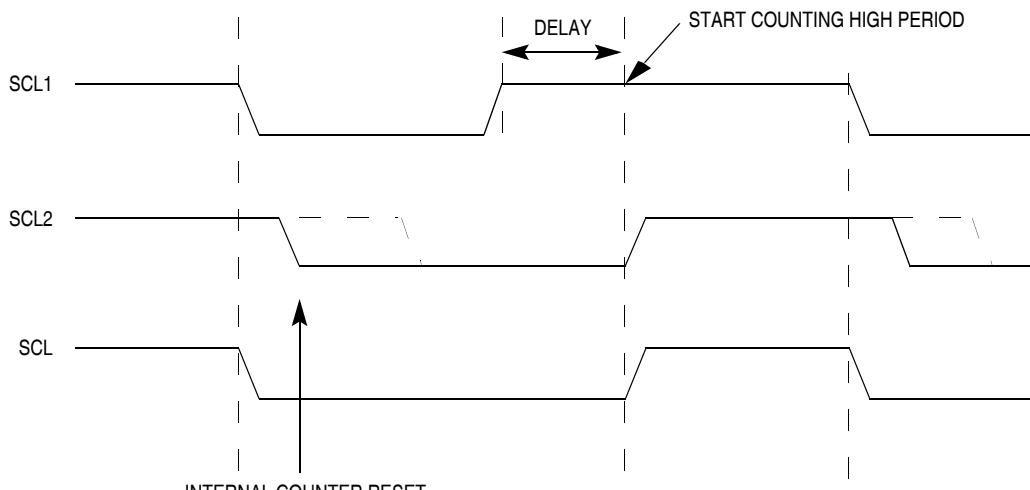


Figure 15-11. IIC Clock Synchronization

### 15.5.1.8 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 case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

### 15.5.1.9 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.

## 15.5.2 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.

### 15.5.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see [Table 15-12](#)). 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/W direction bit) is 0. It is possible that more than one device will find a match and generate an acknowledge (A1). Each slave that finds a match will compare the eight bits of the second byte of the slave address with its own address, but only one slave will find a match and generate an acknowledge (A2). The matching slave will remain 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 15-12. 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 will see an IIC interrupt. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as valid data.

### 15.5.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/W bit (see [Table 15-13](#)). 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/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 will also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/W) bit. However, none of them will be addressed because R/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 15-13. 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 will see an IIC interrupt. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as valid data.

### 15.5.3 Address Matching

All received Addresses can be requested in 7-bit or 10-bit address. IIC Address Register 1, which contains IIC primary slave address, always participates the address matching process. If the GCAEN bit is set, general call will participate the address matching process. If the ALERTEN bit is set, alert response will participate the address matching process. If SIICAEN bit is set, the IIC Address Register 2 will participate the address matching process.

When the IIC responds to one of above mentioned address, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software need to read the IICD register after the first byte transfer to determine which the address is matched.

### 15.5.4 System Management Bus Specification

SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of tripping individual control lines. Removing the individual control lines reduces pin count. Accepting messages ensures future expandability. With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

#### 15.5.4.1 Timeouts

The TTIMEOUT,MIN parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. It is highly recommended that a slave device release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than TTIMEOUT,MIN. Devices that have detected this condition should reset their communication and be able to receive a new START condition in no later than TTIMEOUT,MAX.

SMBus defines a clock low time-out, TTIMEOUT of 35 ms and specifies TLOW:SEXT as the cumulative clock low extend time for a slave device and specifies TLOW:MEXT as the cumulative clock low extend time for a master device.

##### 15.5.4.1.1 SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a “timeout” value condition. Devices that have detected the timeout condition must reset the communication. When active master, if the IIC detects that SMBCLK low has exceeded the value of TTIMEOUT,MIN it must generate a stop condition within or after the current data byte in the transfer process. When slave, upon detection of the TTIMEOUT,MIN condition, the IIC shall reset its communication and be able to receive a new START condition.

### 15.5.4.1.2 SCL High (SMBus Free) Timeout

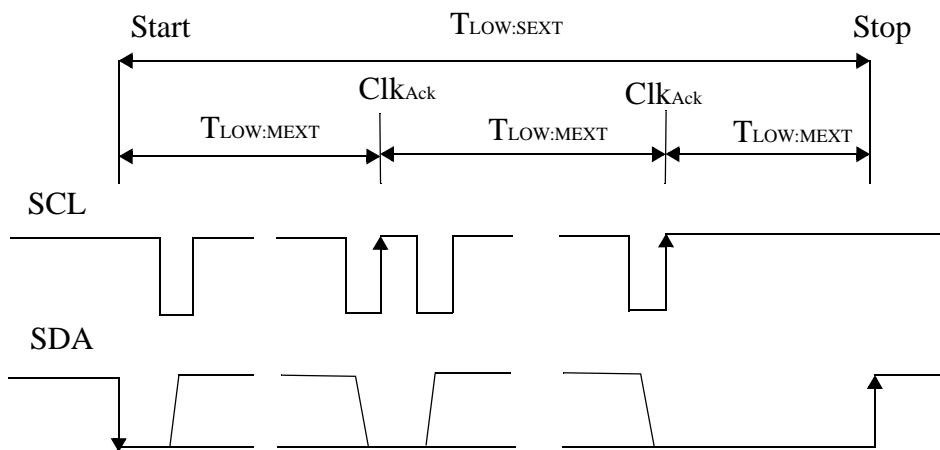
The IIC shall assume that the bus is idle, when it has determined that the SMBCLK and SMBDAT signals have been high for at least THIGH:MAX.HIGH timeout can occur in two ways: 1) HIGH timeout detected after a STOP condition appears on the bus; 2) HIGH timeout detected after a START condition, but before a STOP condition appears on the bus. Any master detecting either scenario can assume the bus is free then SHTF rises. HIGH timeout occurred in scenario 2 if it ever detects that both the following is true: BUSY bit is high and SHTF is high.

### 15.5.4.1.3 CSMBCLK TIMEOUT MEXT

Figure1-10: Timeout measurement intervals illustrates the definition of the timeout intervals, TLOW:SEXT and TLOW:MEXT. When master mode, the I2C must not cumulatively extend its clock cycles for a period greater than TLOW:MEXT within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs SMBus MEXT will rise and also trigger the SLTF.

### 15.5.4.1.4 CSMBCLK TIMEOUT SEXT

A Master is allowed to abort the transaction in progress to any slave that violates the TLOW:SEXT or TTIMEOUT,MIN specifications. This can be accomplished by the Master issuing a STOP condition at the conclusion of the byte transfer in progress. When slave, the I2C must not cumulatively extend its clock cycles for a period greater than TLOW:SEXT during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs SEXT will rise and also trigger SLTF.



**Figure 15-12. Timeout measurement intervals**

Note: CSMBCLK TIMEOUT SEXT and MEXT are optional functions which will be implemented in second step.

#### 15.5.4.2 FAST ACK and NACK

To improve reliability and communication robustness, implementation of Packet Error Checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the Address Resolution Protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by receiver. Otherwise an ACK will be issued. In order to calculate the CRC-8 by software, this module can hold SCL line to low after receiving eighth SCL (bit 8th) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent out to the bus by setting or clearing TXAK bit if FASK (fast ACK/NACK enable bit) is enabled.

SMBus requires devices to acknowledge their own address always, as a mechanism to detect a removable devices presence on the bus (battery, docking station, etc.) Besides to indicate a slave device busy condition, SMBus is using the NACK mechanism also to indicate the reception of an invalid command or data. Since such a condition may occur on the last byte of the transfer, it is required that SMBus devices have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

#### NOTE

In the last byte of master receive slave transmit mode, the master should send NACK to bus so FACK should be switched off before the last byte transmit.

### 15.6 Resets

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

### 15.7 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in [Table 15-14](#) 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. The user can determine the interrupt type by reading the status register. For SMBus timeouts interrupt, the interrupt is driven by SLTF and masked with bit IICIE. The SLTF bit must be cleared by software by writing a 1 to it in the interrupt routine. The user can determine the interrupt type by reading the status register.

Note: In Master receive mode the FACK should be set zero before the last byte transfer.

**Table 15-14. 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
SMBus Timeout Interrupt Flag	SLTF	IICIF	IICIE

### 15.7.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

### 15.7.2 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.

### 15.7.3 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 by writing a 1 to it.

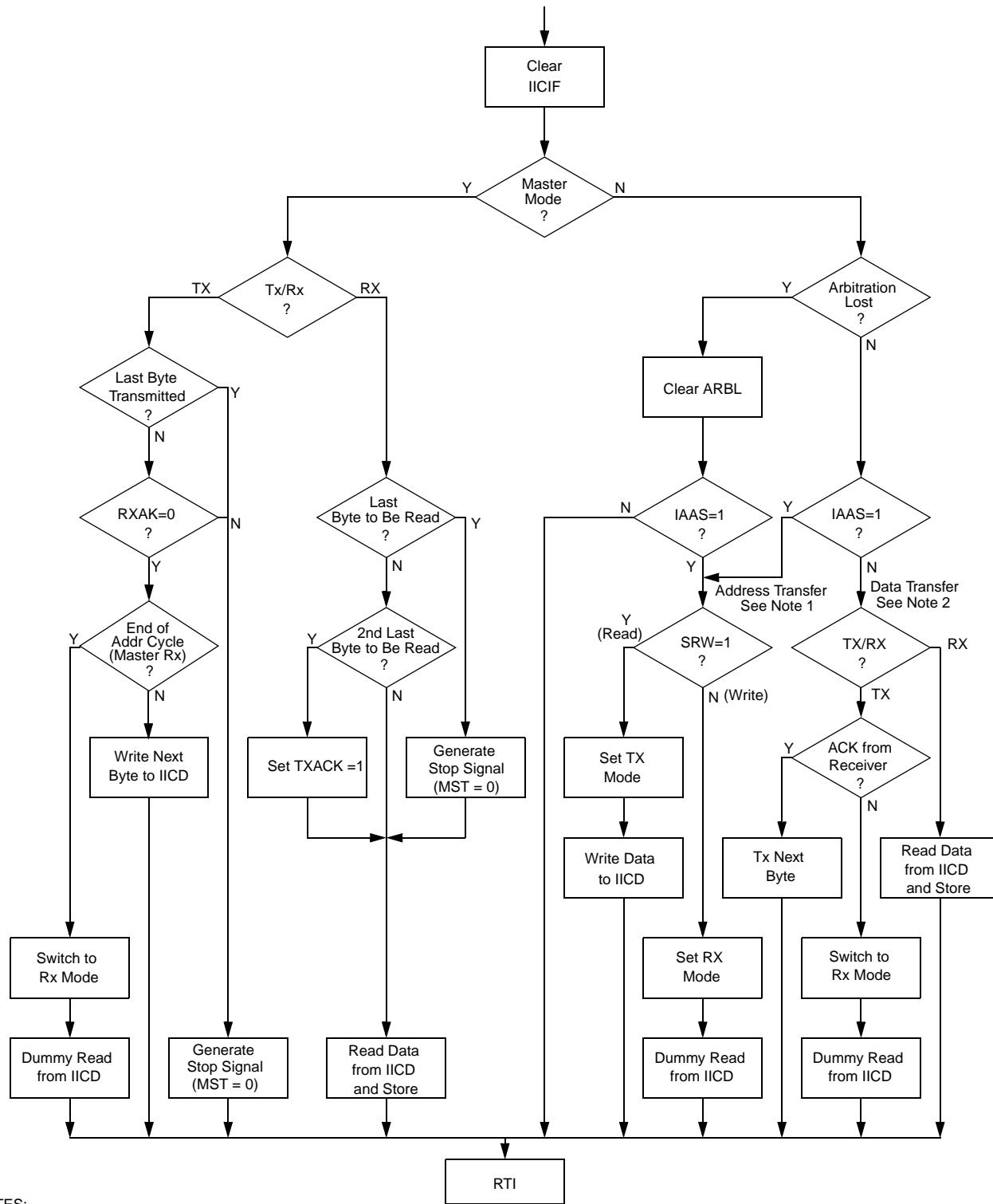
### 15.7.4 Timeouts Interrupt in SMBus

When IICIE is set, the IIC asserts a timeout interrupt output SLTF upon detection of any of the mentioned timeout conditions, with one exception. The HIGH TIMEOUT mechanism shall not be used to influence the timeout interrupt output, because the HIGH TIMEOUT indicates an idle condition on the bus.

### 15.7.5 Programmable input glitch filter

An IIC glitch filter has been added outside the IIC legacy modules, but within the IIC package. This filter can absorb glitches on the IIC clock and data lines for I2C module. The width of the glitch to absorb can be specified in terms of number of half bus clock cycles. A single glitch filter control register is provided as IICFLT. Effectively, any down-up-down or up-down-up transition on the data line that occurs within the number of clock cycles programmed here is ignored by the IIC. the programmer only needs to specify the size of glitch (in terms of bus clock cycles) for the filter to absorb and not pass.

## 15.8 Initialization/Application Information



## NOTES:

- 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.
- When 10-bit addressing is used to address a slave, the slave will see 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 15-13. Typical IIC Interrupt Routine



# Chapter 16

## Interrupt Priority Controller (S08IPCV1)

### 16.1 Introduction

The standard S08 CPU has a fixed priority for the system interrupts. The interrupt priority controller (IPC) allows modification to the default priority scheme by raising or lowering individual interrupt levels. The IPC supports 4 level of priority interrupts.

The interrupt priority controller (IPC) provides a flexible four-level interrupt prioritization mechanism for the HCS08 family of microcontrollers. The IPC resides between the interrupt source and the CPU. The HCS08 CPU interrupt functions are not altered and the I bit (interrupt mask bit) can allow or disallow interrupt requests to the CPU.

The interrupt priority controller (IPC) allows the user to enable preemptive interrupts built on the existing HCS08 CPU architecture with minimal software overhead.

The interrupt in (INTIN2–INTIN32) assignment is equal to the vector number which is specified in the resets and interrupts chapter. [Table 16-1](#) is the interrupt sources configuration in MC9S08MP16 Series.

**Table 16-1. Interrupt Sources Configuration in MC9S08MP16 Series**

Vector Number	IPC Interrupt Input	Interrupt Source	Address (High:Low)
31	INTIN31	0	0xFFC0:FFC1
30	INTIN30	KBI3	0xFFC2:FFC3
29	INTIN29	KBI2	0xFFC4:FFC5
28	INTIN28	KBI1	0xFFC6:FFC7
27	INTIN27	HSCMP3	0xFFC8:FFC9
26	INTIN26	HSCMP2	0xFFCA:FFCB
25	INTIN25	HSCMP1	0xFFCC:FFCD
24	INTIN24	RTC	0xFFCE:FFCF
23	INTIN23	IIC	0xFFD0:FFD1
22	INTIN22	SCI Transmit	0xFFD2:FFD3
21	INTIN21	SCI Receive	0xFFD4:FFD5
20	INTIN20	SCI Error	0xFFD6:FFD7
19	INTIN19	SPI	0xFFD8:FFD9
18	INTIN18	ADC Conversion	0xFFDA:FFDB

**Table 16-1. Interrupt Sources Configuration in MC9S08MP16 Series (continued)**

<b>Vector Number</b>	<b>IPC Interrupt Input</b>	<b>Interrupt Source</b>	<b>Address (High:Low)</b>
17	INTIN17	PDB2	0xFFDC:FFDD
16	INTIN16	PDB1	0xFFDE:FFDF
15	INTIN15	MTIM	0xFFE0:FFE1
14	INTIN14	FTM2 Overflow	0xFFE2:FFE3
13	INTIN13	FTM2 Channel 5	0xFFE4:FFE5
12	INTIN12	FTM2 Channel 4	0xFFE6:FFE7
11	INTIN11	FTM2 Channel 3	0xFFE8:FFE9
10	INTIN10	FTM2 Channel 2	0xFFEA:FFEB
9	INTIN9	FTM2 Channel 1	0xFFEC:FFED
8	INTIN8	FTM2 Channel 0	0xFFEE:FFEF
7	INTIN7	FTM1 Overflow	0xFFF0:FFF1
6	INTIN6	FTM1 Channel 1	0xFFF2:FFF3
5	INTIN5	FTM1 Channel 0	0xFFF4:FFF5
4	INTIN4	FTM2 Fault	0xFFF6:FFF7
3	INTIN3	FTM1 Fault	0xFFF8:FFF9
2	INTIN2	Low Voltage Warning	0xFFFFA:FFFFB
1	INTIN1	0	0xFFFFC:FFFFD
0	INTIN0	0	0xFFFFE:FFFFF

### 16.1.1 Stop3 Operation

MCU wakeups from stop3 are through an asynchronous path. IPC must be configured to allow all enabled asynchronous modules to an interrupt priority level (ILRxx) equal to or greater than the interrupt priority mask level (IPM). If not configured properly, system clocks may begin operation without CPU waking from stop.

## 16.1.2 Features

The interrupt priority controller (IPC) includes the following features:

- Four-level programmable interrupt priority for each interrupt source
- Support for prioritized preemptive interrupt service routines
  - Lower priority interrupt requests are blocked when higher priority interrupts are being serviced
  - Higher or equal priority level interrupt requests can preempt lower priority interrupts being serviced
- Automatic update of interrupt priority mask with being serviced interrupt source priority level when the interrupt vector is being fetched
- Interrupt priority mask can be modified during main flow or interrupt service execution
- Previous interrupt mask level is automatically stored when interrupt vector is fetched(four levels of previous values accommodated)

## 16.1.3 Modes of Operation

### 16.1.3.1 Run Mode

In run mode, if the IPC is enabled, interrupt requests are qualified against interrupt mask register and unique interrupt level register before being sent to the CPU. If the IPC is disabled, the module is inactive and is transparently allowing interrupt requests to pass to HCS08 CPU, no programmable priority or priority preemptive interrupt is supported.

### 16.1.3.2 Wait Mode

In wait mode, the IPC module acts as it does in run mode.

### 16.1.3.3 Stop Mode

In stop3 mode, the interrupt mask is set to 0 and the IPC module is bypassed. The IPC interrupt mask value upon the stop3 entry is automatically restored when exiting stop3. This ensures that asynchronous interrupt can still wake up CPU from stop3 mode.

If the stop3 exits with an interrupt, the IPC continues working with the previous setting; If the stop3 exits with a reset, the IPC will return to its reset state.

In stop2 and stop1 mode, the IPC module is powered off, the MCU works as the module is not there . Upon the exiting of stop2 and stop1, the IPC module is reset.

## 16.1.4 Block Diagram

Figure 16-1 is the block diagram of the interrupt priority controller module (IPC).

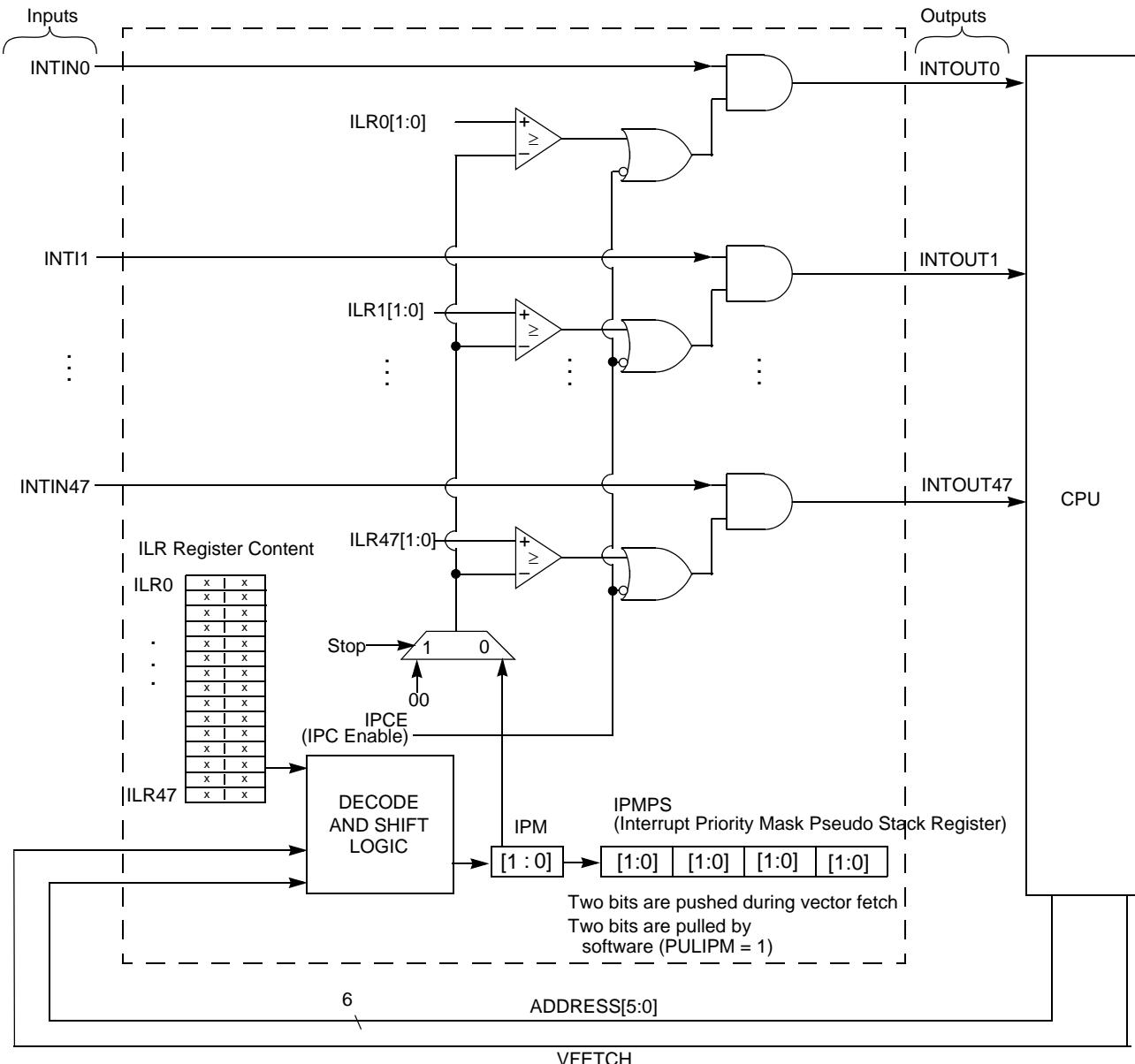


Figure 16-1. Interrupt Priority Controller (IPC) Block Diagram

## 16.2 External Signal Description

Table 16-2. Signal Properties

Name	Port	Function	Reset State	Pull Up
INTIN[47:2]	N/A	Interrupt source interrupt request input	Input	N/A
VFETCH	N/A	Vector fetch indicator from HCS08 CPU	Input	N/A
IADB[5:0]	N/A	Address bus input from HCS08 CPU	Input	N/A
INTOUT[47:2]	N/A	Interrupt request to HCS08 CPU	Output	N/A

### 16.2.1 INTIN[47:0] — Interrupt Source Interrupt Request Input

Input from interrupt sources..

### 16.2.2 VFETCH — Vector Fetch Indicator from HCS08 CPU

Vector fetch signal generated from HCS08 CPU.

### 16.2.3 IADB[5:0] — Address Bus Input from HCS08 CPU

Internal address bus used to decode the IPC registers.

### 16.2.4 INTOUT[47:0] — Interrupt Request to HCS08 CPU

Interrupt output signals to HCS08 CPU.

## 16.3 Register Definition

[Table 16-3](#) is the memory map of the IPC.

**Table 16-3. Module Memory Map**

Address	Use	Access
Base1 + \$00	IPC Status and Control Register (IPCSC)	Read/Write
Base1 + \$01	Interrupt Priority Mask Pseudo Stack Register (IPMPS)	Read
Base2 + \$00	Interrupt Level Register Set 0 (ILRS0)	Read/Write
Base2 + \$01	Interrupt Level Register Set 1 (ILRS1)	Read/Write
Base2 + \$02	Interrupt Level Register Set 2 (ILRS2)	Read/Write
Base2 + \$03	Interrupt Level Register Set 3 (ILRS3)	Read/Write
Base2 + \$04	Interrupt Level Register Set 4 (ILRS4)	Read/Write
Base2 + \$05	Interrupt Level Register Set 5 (ILRS5)	Read/Write
Base2 + \$06	Interrupt Level Register Set 6 (ILRS6)	Read/Write
Base2 + \$07	Interrupt Level Register Set 7 (ILRS7)	Read/Write
Base2 + \$08	Interrupt Level Register Set 8 (ILRS8)	Read/Write
Base2 + \$09	Interrupt Level Register Set 9 (ILRS9)	Read/Write
Base2 + \$0A	Interrupt Level Register Set 10 (ILRS10)	Read/Write
Base2 + \$0B	Interrupt Level Register Set 11 (ILRS11)	Read/Write

### 16.3.1 IPC Status and Control Register (IPCSC)

This register contains status and control bits for the IPC.

	7	6	5	4	3	2	1	0
R	IPCE	0	PSE	PSF	0	0		IPM
W					PULIPM			
Reset	0	0	1	0	0	0	0	0
= Unimplemented or Reserved								

Figure 16-2. IPC Status and Control Register (IPCSC)

Table 16-4. IPCSC Field Descriptions

Field	Description
7 IPCE	<b>Interrupt Priority Controller Enable</b> — This bit enables/disables the interrupt priority controller module. 0 Disables IPCE. Interrupt generated from the interrupt source is passed directly to CPU without processing. (Bypass mode) The IPMPS register is not updated when the module is disabled. 1 Enables IPCE and interrupt generated from the interrupt source is processed by IPC before passing to CPU.
5 PSE	<b>Pseudo Stack Empty</b> — This bit indicates that the pseudo stack has no valid information. This bit is automatically updated after each IPMPS register push or pull operation.
4 PSF	<b>Pseudo Stack Full</b> — This bit indicates that the pseudo stack register IPMPS register is full. It is automatically updated after each IPMPS register push or pull operation. If additional interrupt is nested after this bit is set, the earliest interrupt mask value(IPM0[1:0]) stacked in IPMPS will be lost. 0 IPMPS register is not full. 1 IPMPS register is full.
3 PULIPM	<b>Pull IPM from IPMPS</b> — This bit pulls stacked IPM value from IPMPS register to IPM bits of IPCSC. Zeros are shifted into bit positions 1 and 0 of IPMPS. 0 No operation. 1 Writing 1 to this bit causes a 2-bit value from the interrupt priority mask pseudo stack register to be pulled to the IPM bits of IPCSC to restore the previous IPM value.
1:0 IPM	<b>Interrupt Priority Mask</b> — This field sets the mask for the interrupt priority control. If the interrupt priority controller is enabled, the interrupt source with interrupt level (ILRxx) value which is greater than or equal to the value of IPM will be presented to the CPU. Writes to this field are allowed, but doing this will not push information to the IPMPS register. Writing IPM with PULIPM setting when IPCE is already set, the IPM will restore the value pulled from the IPMPS register, not the value written to the IPM register

### 16.3.2 Interrupt Priority Mask Pseudo Stack Register (IPMPS)

This register is used to store the previous interrupt priority mask level temporarily while the currently active interrupt is executed.

	7	6	5	4	3	2	1	0
R	IPM3		IPM2		IPM1		IPM0	
W								
Reset	0	0	0	0	0	0	0	0
= Unimplemented or Reserved								

Figure 16-3. Interrupt Priority Mask Pseudo Stack Register (IPMR)

**Table 16-5. IPMPS Positions 0–3 Field Descriptions**

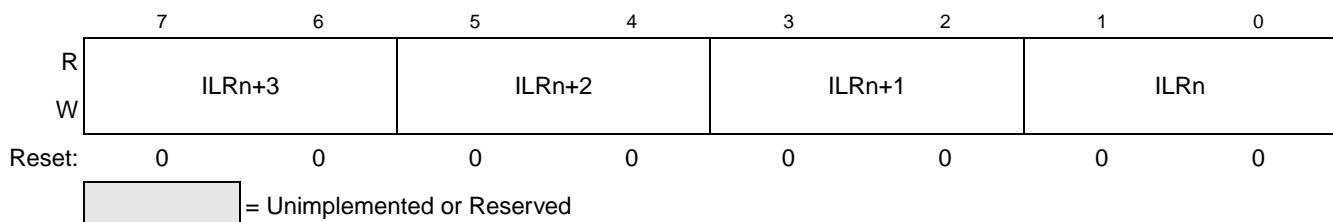
Field	Description
7:6 IPM3	<b>Interrupt Priority Mask pseudo stack position 3</b> — This field is the pseudo stack register for IPM3. The most recent information is stored in IPM3.
5:4 IPM2	<b>Interrupt Priority Mask pseudo stack position 2</b> — This field is the pseudo stack register for IPM2.
3:2 IPM1	<b>Interrupt Priority Mask pseudo stack position 1</b> — This field is the pseudo stack register for IPM1.
1:0 IPM0	<b>Interrupt Priority Mask pseudo stack position 0</b> — This field is the pseudo stack register for IPM0.

### 16.3.3 Interrupt Level Setting Registers (ILRS0–ILRS11)

This set of registers (ILRS0–ILRS11) contains the user specified interrupt level for each interrupt source. In [Figure 16-4](#), x indicates the number of the register (ILRSx is ILRS0 through ILRS11). Also, n is the field number (ILRn is ILR0 through ILR47). Refer to [Table 16-6](#).

**Table 16-6. Interrupt Level Register Fields**

	7	6	5	4	3	2	1	0
ILRS0	ILR3		ILR2		ILR1		ILR0	
ILRS1	ILR7		ILR6		ILR5		ILR4	
ILRS2	ILR11		ILR10		ILR9		ILR8	
ILRS3	ILR15		ILR14		ILR13		ILR12	
ILRS4	ILR19		ILR18		ILR17		ILR16	
ILRS5	ILR23		ILR22		ILR21		ILR20	
ILRS6	ILR27		ILR26		ILR25		ILR24	
ILRS7	ILR31		ILR30		ILR29		ILR28	
ILRS8	ILR34		ILR34		ILR33		ILR32	
ILRS9	ILR39		ILR38		ILR37		ILR36	
ILRS10	ILR43		ILR42		ILR41		ILR40	
ILRS11	ILR47		ILR46		ILR45		ILR44	

**Figure 16-4. Interrupt Level Register Set ILRx (ILRS0–ILRS11)**

**Table 16-7. Interrupt Level Registers**

Field	Description
7:6 ILR <sub>n+3</sub>	<b>Interrupt Level Register for Source n+3</b> — This field sets the interrupt level for interrupt source n+3.
5:4 ILR <sub>n+2</sub>	<b>Interrupt Level Register for Source n+2</b> — This field sets the interrupt level for interrupt source n+2.
3:2 ILR <sub>n+1</sub>	<b>Interrupt Level Register for Source n+1</b> — This field sets the interrupt level for interrupt source n+1.
1:0 ILR <sub>n</sub>	<b>Interrupt Level Register for Source n</b> — This field sets the interrupt level for interrupt source n.

The number of ILRS registers is parameterized in the design, the number can be 4, 6, 8, 10 and 12 based on the actual interrupt number in the design. The corresponding interrupt number is 16, 24, 32, 40 and 48 separately.

## 16.4 Functional Description

The IPC works with the existing HCS08 interrupt mechanism to allow nestable interrupts with programmable priority levels. This module also allows implementation of preemptive interrupt according to the programmed interrupt priority with minimal software overhead. The IPC consists of three major functional blocks.

- The interrupt priority level registers
- The interrupt priority level comparator set
- The interrupt mask register update and restore mechanism

### 16.4.1 Interrupt Priority Level Register

This set of registers is associated with the interrupt sources to the HCS08 CPU. Each interrupt priority level is a 2-bit value such that a user can program the interrupt priority level of each source to priority 0, 1, 2, or 3. Level 3 has the highest priority while level 0 has the lowest. Software can read or write to these registers at any time. The interrupt priority level comparator set, interrupt mask register update, and restore mechanism use this information.

### 16.4.2 Interrupt Priority Level Comparator Set

When the module is enabled, an active interrupt request forces a comparison between the corresponding ILR and the 2-bit interrupt mask IPM[1:0](in stop3 mode, the IPM[1:0] is substituted by value 0x00).. If the ILR value is greater than or equal to the value of the interrupt priority mask (IPM bits in IPCSC), the corresponding interrupt out (INTOUT) signal will be asserted and will signal an interrupt request to the HCS08 CPU.

When the module is disabled, the interrupt request signal from the source is directly passed to the CPU.

Because the IPC is an external module, the interrupt priority level programmed in the interrupt priority register will not affect the inherent interrupt priority arbitration as defined by the HCS08 CPU. Therefore, if two (or more) interrupts are present in the HCS08 CPU at the same time, the inherent priority in HCS08 CPU will perform arbitration by the inherent interrupt priority.

### 16.4.3 Interrupt Priority Mask Update and Restore Mechanism

The interrupt priority mask (IPM) is 2-bits located in the least significant end of IPCSC register. This two bits controls which interrupt is allowed to be presented to the HCS08 CPU. During vector fetch, the interrupt priority mask is updated automatically with the value of the ILR corresponding to that interrupt source. The original value of the IPM will be saved onto IPMPS for restoration after the interrupt service routine completes execution. When the interrupt service routine completes execution, the user restore the original value of IPM by writing 1 to the PULIPM bit. In both cases, the IPMPS is a shift register functioning as a pseudo stack register for storing the IPM. When the IPM is updated, the original value is shifted into IPMPS. The IPMPS can store four levels of IPM.. If the last position of IPMPS is written, the PSF flag indicates that the IPMPS is full. If all the values in the IPMPS were read, the PSE flag indicates that the IPMPS is empty.

### 16.4.4 The Integration and Application of the IPC

All the interrupt inputs coming from peripheral modules are synchronous signals. None of asynchronous signals of the interrupts are routed to IPC. The asynchronous signals of the interrupts are routed directly to SIM module to wake up system clocks in stop3 mode.

Additional care should be exercised when IRQ is re-prioritized by IPC. CPU instructions BIL and BIH need input from IRQ pin. If IRQ interrupt is masked, BIL and BIH still work but the IRQ interrupt will not occur.

## 16.5 Application Examples

[Figure 16-5](#) and [Figure 16-6](#) are the examples of the IPC operation at interrupt entry and exiting.

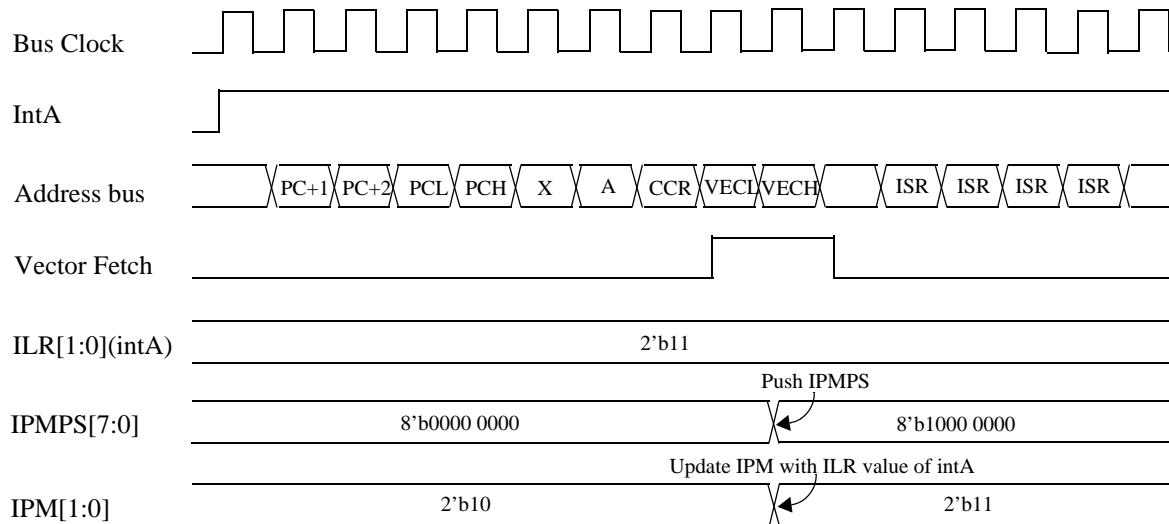


Figure 16-5. IPC Operation at Interrupt Entry

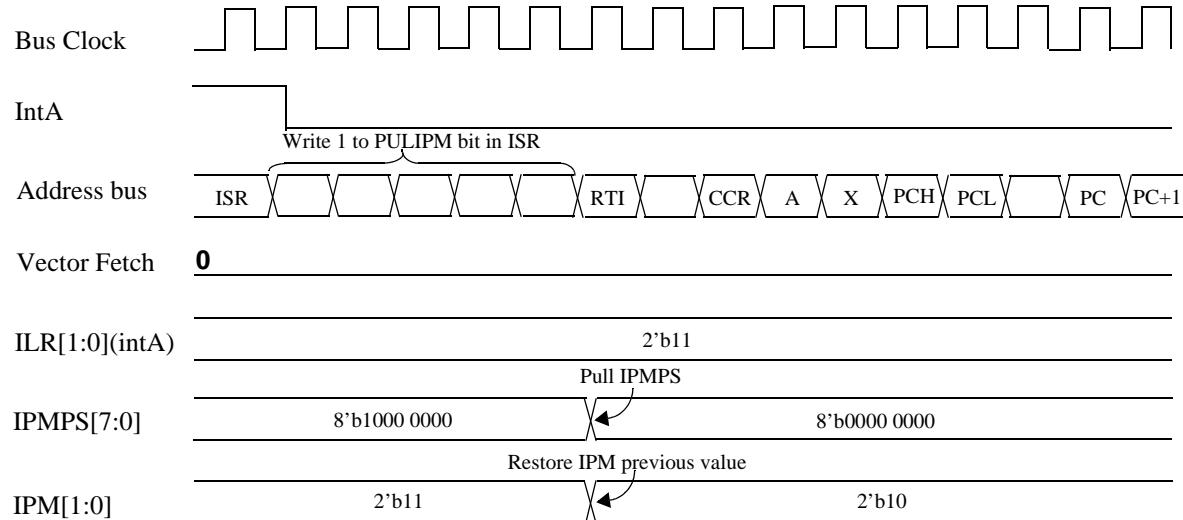


Figure 16-6. IPC Operation at Interrupt Exiting

## 16.6 Initialization/Application Information

- The interrupt priority controller must be enabled to function.. While inside an interrupt service routine, some work has to be done to enable other higher priority interrupts . The following is a pseudo code per example written in assembly language:

```
INT_SER :
    BCLR    INTFLAG, INTFLAG_R ; clear flag that generate interrupt
    .
    .
    .
    .
    .
    CLI     ; global interrupt enable and nested interrupt enabled
    .
    .
    .
    .
    .
    .
    BSET    PULIPM, PULIPM_R ; restore the old IPM value before leaving
    RTI
```

- A minimum overhead of six bus clock cycles is added inside an interrupt services routine to enable preemptive interrupts.
- As interrupt of same priority level is allowed to pass through IPC to HCS08 CPU thus the flag generating the interrupt should be cleared before doing CLI to enable preemptive interrupts.
- The IPM is automatically updated to the level the interrupt is servicing and the original level is kept in IPMPS.. Watch out for the full(PSF) bit if nesting for more than 4 level is expected.
- Before leaving the interrupt service routine the previous levels should be restored manually by setting PULIPM bitWatch out for the full(PSF) bit and empty(PSE) bit.



# Chapter 17

## Modulo Timer (S08MTIMV1)

### 17.1 Introduction

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

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

[Figure 17-1](#) shows the MC9S08MP16 Series block diagram with the MTIM highlighted.

### 17.2 Module Configurations

This section provides device-specific information for configuring the MTIM on MC9S08MP16 Series.

#### 17.2.1 MTIM Clock Gating

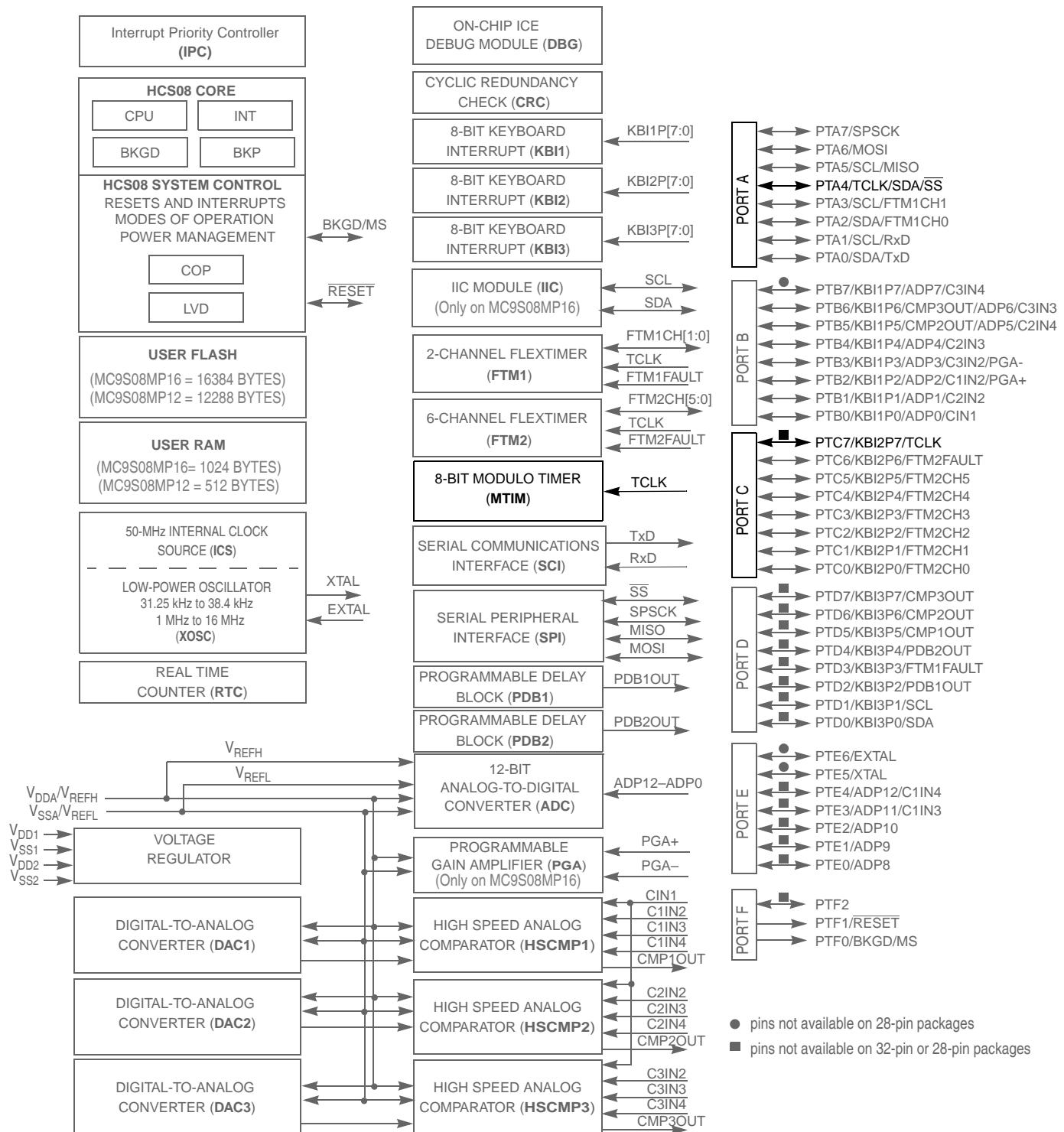
The bus clock to the MTIM can be gated on and off using the SCGC1[MTIM] bit. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the MTIM bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 17.2.2 MTIM External Clock

The external clock for the MTIM module, TCLK, is selected by setting MTIMCLK[CLKS] to 11 or 10, which selects the TCLK pin input. The TCLK input can be enabled as external clock inputs to the MTIM and both FTM modules simultaneously.

#### 17.2.3 MTIM External Clock Pin Repositioning

The TCLK pin can be repositioned under software control using SOPT2[TCLKPS]. When TCLKPS is clear (default), PTC7 is the TCLK input pin. When TCLKPS is set, PTA4 is the TCLK input pin.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

**Figure 17-1. MC9S08MP16 Series Block Diagram Highlighting MTIM Block and Pins**



## 17.2.4 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

## 17.2.5 Modes of Operation

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

### 17.2.5.1 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.

### 17.2.5.2 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.

### 17.2.5.3 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 MTIMMOD written).

## 17.3 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 17-1](#).

**Table 17-1. 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.

## 17.4 Memory Map and Register Definition

### 17.4.1 Memory Map (Register Summary)

**Table 17-2. MTIM Register Summary**

Name		7	6	5	4	3	2	1	0						
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0						
	W			TRST											
MTIMCLK	R	0	0	CLKS	PS										
	W														
MTIMCNT	R	COUNT													
	W														
MTIMMOD	R	MOD													
	W														

### 17.4.2 Register Descriptions

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 section of this data sheet for the absolute address assignments for all MTIM registers. This section refers to registers and control bits only by their names 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.

### 17.4.2.1 MTIM Status and Control Register (MTIMSC)

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

	7	6	5	4		3	2	1	0
R	TOF		0		TSTP	0	0	0	0
W		TOIE		TRST					
Reset:	0	0	0	1		0	0	0	0

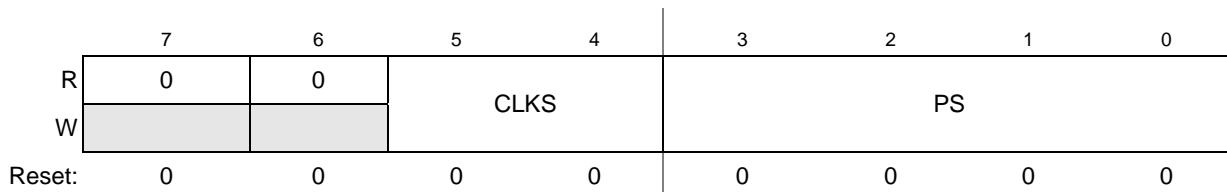
Figure 17-2. MTIM Status and Control Register

Table 17-3. MTIM Status and Control Register Field Descriptions

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

### 17.4.2.2 MTIM Clock Configuration Register (MTIMCLK)

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



**Figure 17-3. MTIM Clock Configuration Register**

**Table 17-4. MTIM Clock Configuration Register Field Description**

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	<b>Clock Source Select</b> — These two read/write bits select one of four different clock sources as the input to the MTIM prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 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	<b>Clock Source Prescaler</b> — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM clock source $\div 1$ 0001 Encoding 1. MTIM clock source $\div 2$ 0010 Encoding 2. MTIM clock source $\div 4$ 0011 Encoding 3. MTIM clock source $\div 8$ 0100 Encoding 4. MTIM clock source $\div 16$ 0101 Encoding 5. MTIM clock source $\div 32$ 0110 Encoding 6. MTIM clock source $\div 64$ 0111 Encoding 7. MTIM clock source $\div 128$ 1000 Encoding 8. MTIM clock source $\div 256$ All other encodings default to MTIM clock source $\div 256$ .

### 17.4.2.3 MTIM Counter Register (MTIMCNT)

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

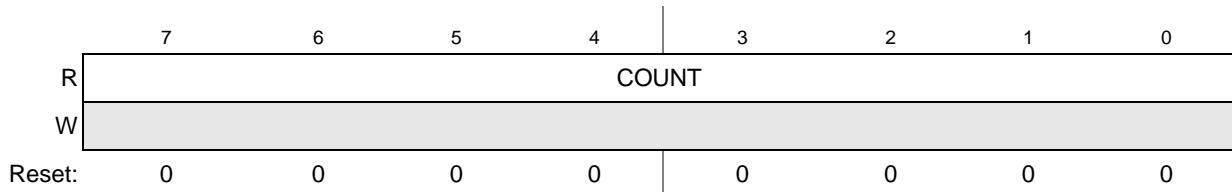


Figure 17-4. MTIM Counter Register

Table 17-5. MTIM Counter Register Field Description

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

### 17.4.2.4 MTIM Modulo Register (MTIMMOD)

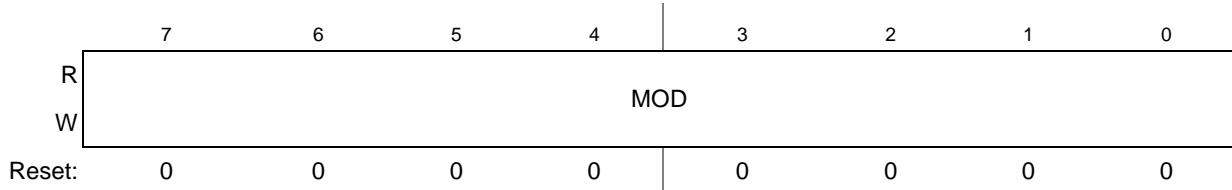


Figure 17-5. MTIM Modulo Register

Table 17-6. MTIM Modulo Register Field Descriptions

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

## 17.5 Functional Description

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

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

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

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

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

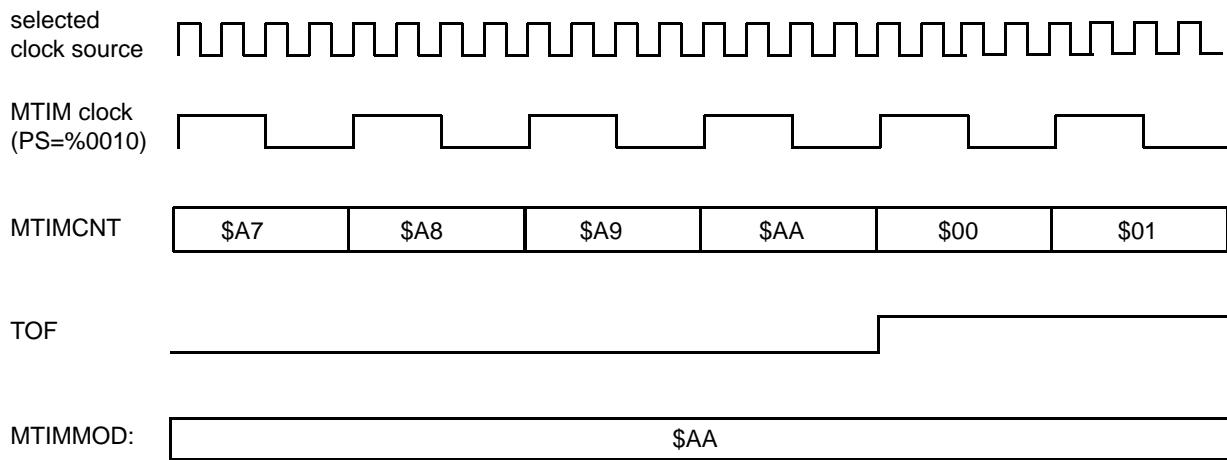
When the counter is active (TSTP = 0), the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter overflows to \$00 and continues counting. The MTIM overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition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

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

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

## 17.5.1 MTIM Operation Example

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



**Figure 17-6. MTIM counter overflow example**

In the example of [Figure 17-6](#), 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 MTIMMOD register is set to \$AA. When the counter, MTIMCNT, reaches the modulo value of \$AA, the counter overflows to \$00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from \$AA to \$00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

# Chapter 18

## Programmable Delay Block (S08PDBV1)

### 18.1 Introduction

The primary function of the programmable delay block (PDB) is to provide:

- a controllable delay from the Flextimer module's SYNC output to the sample trigger input of the programmable gain amplifiers and ADC
- a controllable window which is synchronized with PWM pulses for analog comparators to compare the analog signals in a defined window

An alternate function of the PDB is to generate PWM pulses that are synchronized to the Flextimer, comparators output, and RTC. The trigger signal (which can be selected for the Flextimer's initial trigger, comparators output, RTC overflow, or software) starts PBD or resets the PBD counter, if the PDB is in continuous mode. A single one-shot pulse or pulse string can be generated in result to a response trigger signal.

The MC9S08MP16 Series has two PDB modules, referred to as PDB1 and PDB2. [Figure 18-2](#) shows the MC9S08MP16 Series block diagram with the PDB module highlighted.

### 18.2 Module Configurations

This section provides device-specific information for configuring the PDB on MC9S08MP16 Series.

#### 18.2.1 PDB Clock Gating

The bus clock to the PDB1 and PDB2 modules can be gated on and off using the PDB bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the PDB bit can be cleared to disable the clock to these modules when neither is in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

## 18.2.2 PDB Input Trigger Assignments

Each PDB module supports up to 7 external trigger sources selectable via the TRIGSEL bits in PDBxCTRL2. The PDB input trigger assignments for the MC9S08MP16 Series devices are shown in [Table 18-1](#). The same trigger sources are used for PDB1 and PDB2 triggering.

**Table 18-1. PDB1 and PDB2 Input Trigger Assignment**

TRIGSEL	Trigger Source	
	Module	Signal
000	RTC	Counter Overflow
001	HSCMP1	COUT
010	HSCMP2	COUT
011	HSCMP3	COUT
100	FTM1	Init Trigger
101	FTM2	Init Trigger
110	Unused	—
111		Internal Software Trigger

## 18.2.3 PDB Trigger Output Usage

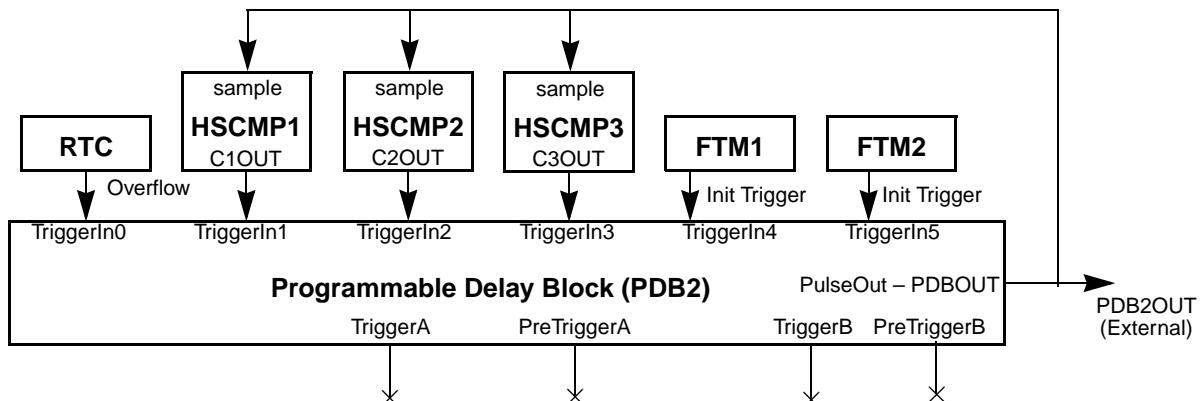
### 18.2.3.1 PDB1 Interfacing to PGA and ADC Hardware Trigger

The PDB1 Trigger A is used by the PGA to initiate differential-to-single-ended conversions. For more details on PGA triggering see [Chapter 19, “Programmable Gain Amplifier \(S08PGAV1\)”. When the PGA is disabled, the PGA is bypassed and the PDB1 Trigger A output triggers the ADC hardware trigger events directly.](#)

PDB1 Trigger B output is always used to trigger the ADC hardware trigger events directly. For more details on ADC triggering see [Chapter 9, “Analog-to-Digital Converter \(S08ADC12V1\)”.](#)

### 18.2.3.2 PDB2 Interfacing to HSCMP Windowing Function

On MC9S08MP16 Series, each of the HSCMP modules can be configured for Windowing or Sampling modes of operation. The PDB2 PulseOut output is used as the Window/Sample input to all three HSCMP modules. [Figure 18-1](#) shows the configuration used for PDB2.



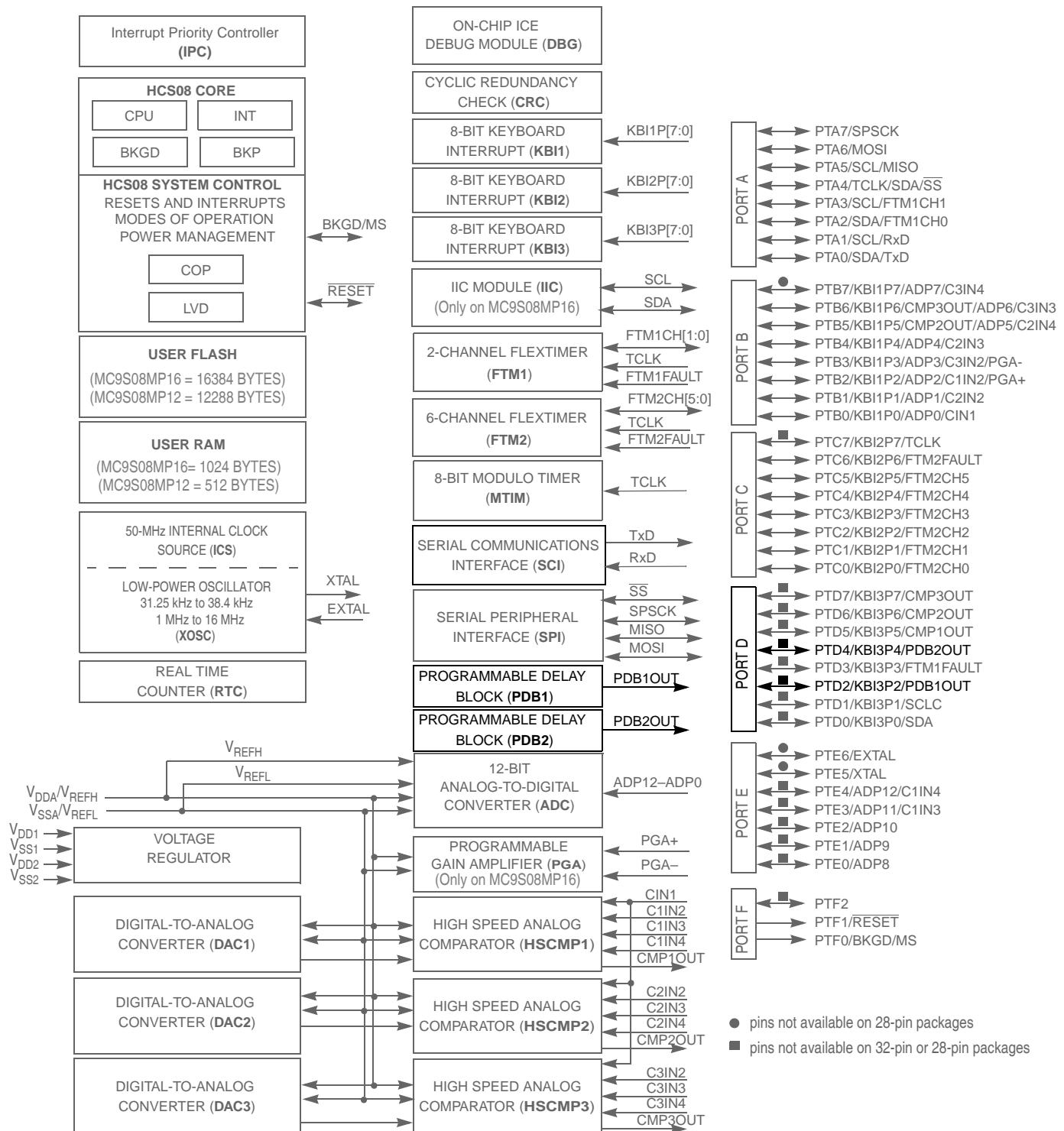
**Figure 18-1. PDB2 Interface to HSCMP1/HSCMP2/HSCMP3 Windowing**

The TriggerA and TriggerB outputs from PDB2 are not used on MC9S08MP16 Series devices and writes to ENA, AOS, ENB and BOS bits in PDB2CTRL1 have no effect.

### 18.2.3.3 PDB Outputs

The PDB outputs are available on the following pins:

- PTD2/KBI3P2/PDB1OUT
- PTD4/KBI3P4/PDB2OUT



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

Figure 18-2. MC9S08MP16 Series Block Diagram Highlighting PDB Block and Pins

## 18.2.4 Features

- 16-bit resolution with prescaler
- Positive transition of trigger event signal initiates the counter
- Supports two triggered delay outputs. Each has an independently-controlled delay from the trigger event
- PDB outputs can be ORed together to schedule two conversions from one input trigger event
- PDB outputs can schedule precise edge placement for a pulsed output. This feature is used to generate the control signal for the HSCMP windowing feature and output to a package pin if needed for applications, such as critical conductive mode power factor correction.
- Continuous-pulse output or single-shot mode supported
- Bypass mode supported
- Each PDB output is independently enabled
- Seven possible trigger events

## 18.2.5 Modes of Operation

Modes of operation include:

- Disabled — Counter is off and TriggerA and TriggerB are low.
- Enabled One-Shot — Counter is enabled and restarted at count one upon receiving a positive edge on the trigger input. TriggerA and TriggerB see only one output trigger per input trigger.
- Enabled Continuous — Counter is enabled and restarted at count one. The counter is rolled over to one again when the count reaches the value specified in the MOD register, and counting restarted. This enables a continuous stream of triggers out as a result of a single trigger input.
- Bypassed — The input trigger bypasses the PDB logic entirely. It is possible to bypass either of the two trigger outputs or both.
- In Enabled one-shot and enabled continuous, the outputs of the delay A and delay B comparators can be combined so that two ADC events can be triggered from a single input event. These are referred to as two-shot and continuous two-shot modes.
- In enabled one-shot and enabled continuous, the outputs of the delay A and delay B comparators can be combined so that an output pulse(s) can be generated with precisely-controlled rising and falling times. These are referred to as single pulse and continuous pulse modes.

## 18.2.6 Block Diagram

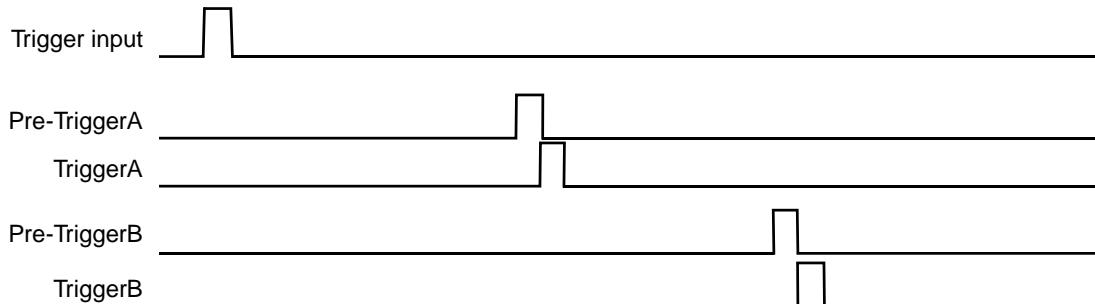
Figure 18-8 illustrates the basic structure of the PDB block. It contains a single counter whose output is compared against three different digital values. delayA and delayB determine the time between assertion of the trigger input to the point at which changes in the trigger output signals are initiated. These times are defined as:

- Trigger input to Pre-TriggerA = (prescaler × delayA) + 1 peripheral bus clock cycle
- Trigger input to Pre-TriggerB = (prescaler × delayB) + 1 peripheral bus clock cycle

- Add one additional peripheral bus clock cycle to determine the time at which the trigger outputs change

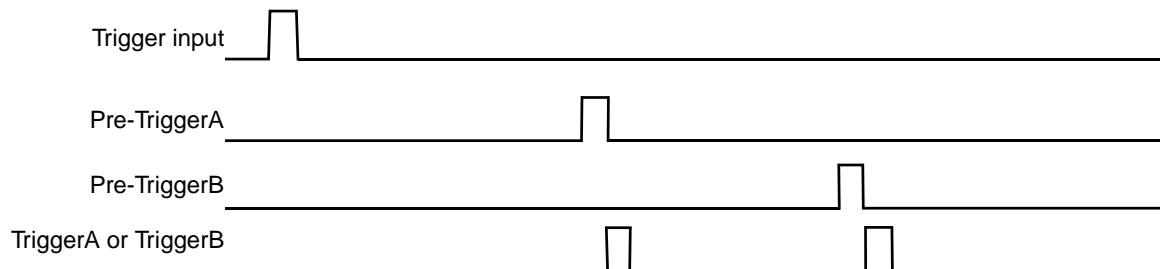
If the device includes the PGAs, Pre-TriggerA, and Pre-TriggerB are used to pre-trigger the PGA in the PGA/ADC blocks with one peripheral bus clock period prior to the actual ADC measurement trigger.

If the ADC block contains duplicate control and result registers, TriggerA and TriggerB allow them to operate in a ping-pong fashion, alternating conversions between two different analog sources (per converter). The Pre-Trigger signals are used to specify which signal is sampled next. The signals shown in [Figure 18-3](#) would be used to operate the ADC to sample signal A and sample signal B in one-shot mode. The trigger delays for the ADC are independently set via the Delay A and Delay B parameters. The one trigger signal from PWM timer module can start the ADC twice with two different delays. But the time between TriggerA and TriggerB must be longer than the ADC conversion time, then the second trigger signal can take effect.



**Figure 18-3. Decoupled A & B Trigger Generation**

If the device integrates two ADCs, two-shot mode is shown in [Figure 18-4](#). In this case, both ADC A and ADC B are given the same trigger; resulting in a total of four ADC conversions (two on ADC A, and two on ADC B).



**Figure 18-4. Trigger Configured for Simultaneous Sampling / Ping-Pong Operation.**

The third digital value, modulus, is used to reset the counter back to one at the end of the count. If PDBxCTRL2[CONT] is set, the counter then resumes a new count. Otherwise, the timer operation ceases until the next trigger input event occurs.

The pulsed modes are shown in [Figure 18-5](#). In this case, Pre-TriggerA and Pre-Trigger B are used to precisely schedule the rising and falling edges for the output waveform.

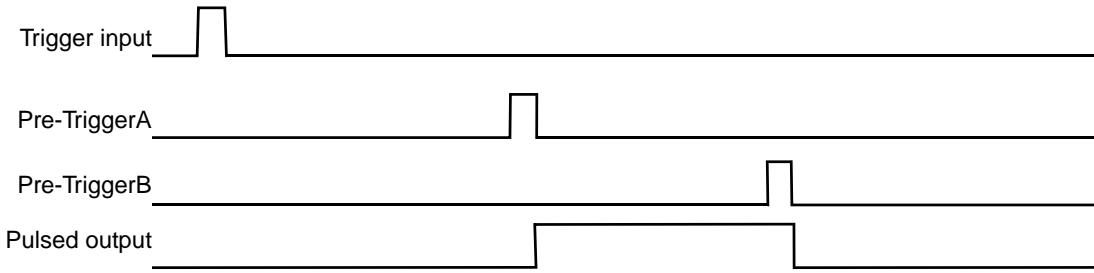


Figure 18-5. Trigger Pulsed Output Operation.

DelayA and DelayB and modulus registers are buffered. The values written into these registers does not take effect until:

- A logic 1 is written to PDBxCTRL1[LDOK] if PDBxCTRL1[LDMOD] is cleared.
- Either counter rolls over in continuous mode or trigger signal is received in one-shot mode after a logic 1 is written to PDBxCTRL1[LDOK] if PDBxCTRL1[LDMOD] is set. In this case, if any value is written to any of these registers, after a logic 1 is written to LDOK, it is ignored until the values in these registers are loaded into the buffers .

LDOK bit remains logic 1 after being set until the values in DelayA and DelayB and modulus registers are loaded into buffers. This bit is readable.

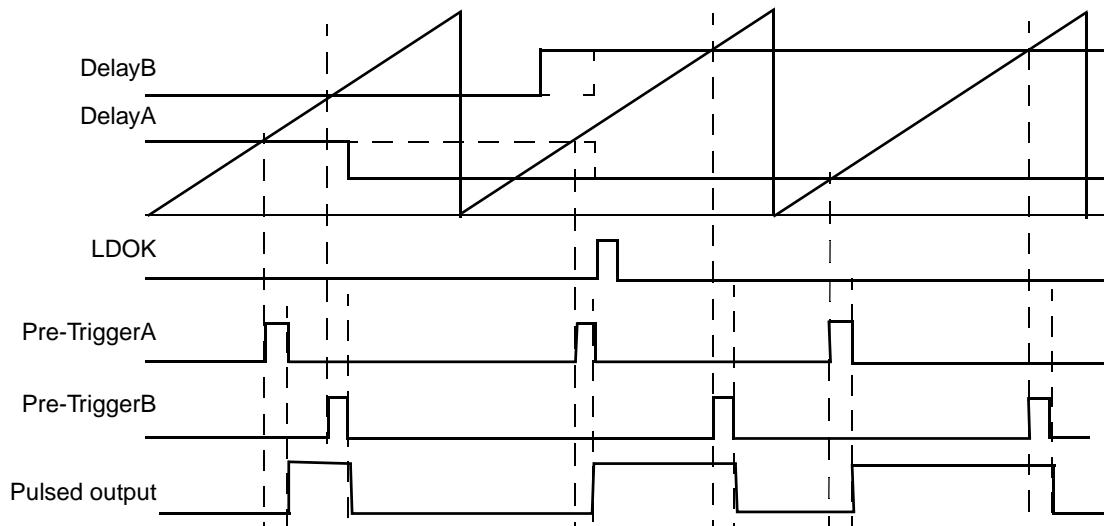


Figure 18-6. Registers Update with LDMOD bit = 0

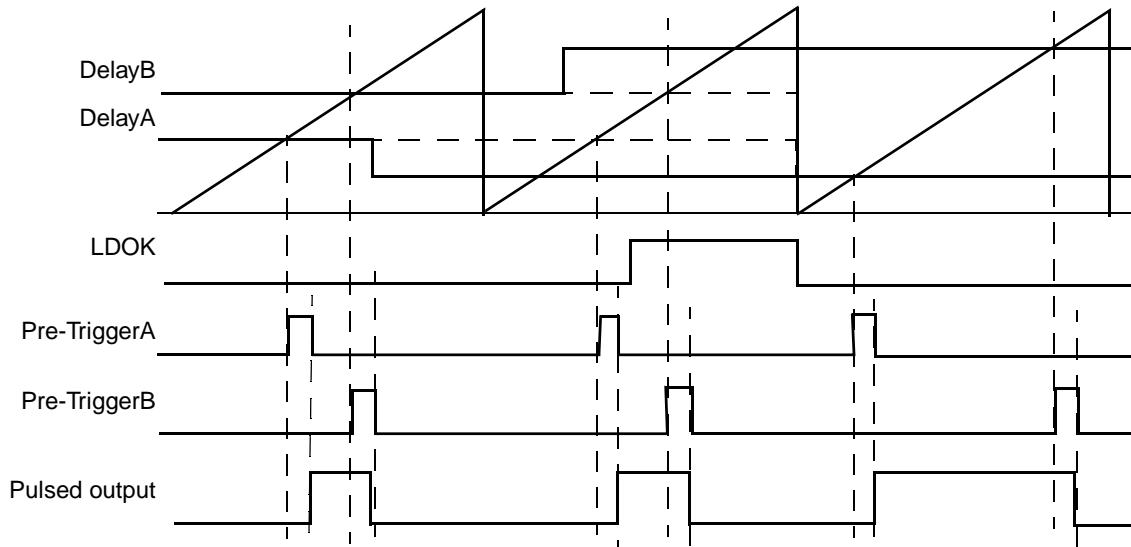


Figure 18-7. Registers Update with LDMOD bit = 1

The 16-bit counter operates in up-count mode. The two read-only counter registers contain the high and low bytes of the value in the PDB counter. Reading either byte latches the contents of both bytes into a buffer where they remain latched until either byte is read. Odd numbered reads return new data from the counter. Even numbered reads return latched data.

Pulse output can be driven on an external pin when PDBxSCR[PADEN] is set.

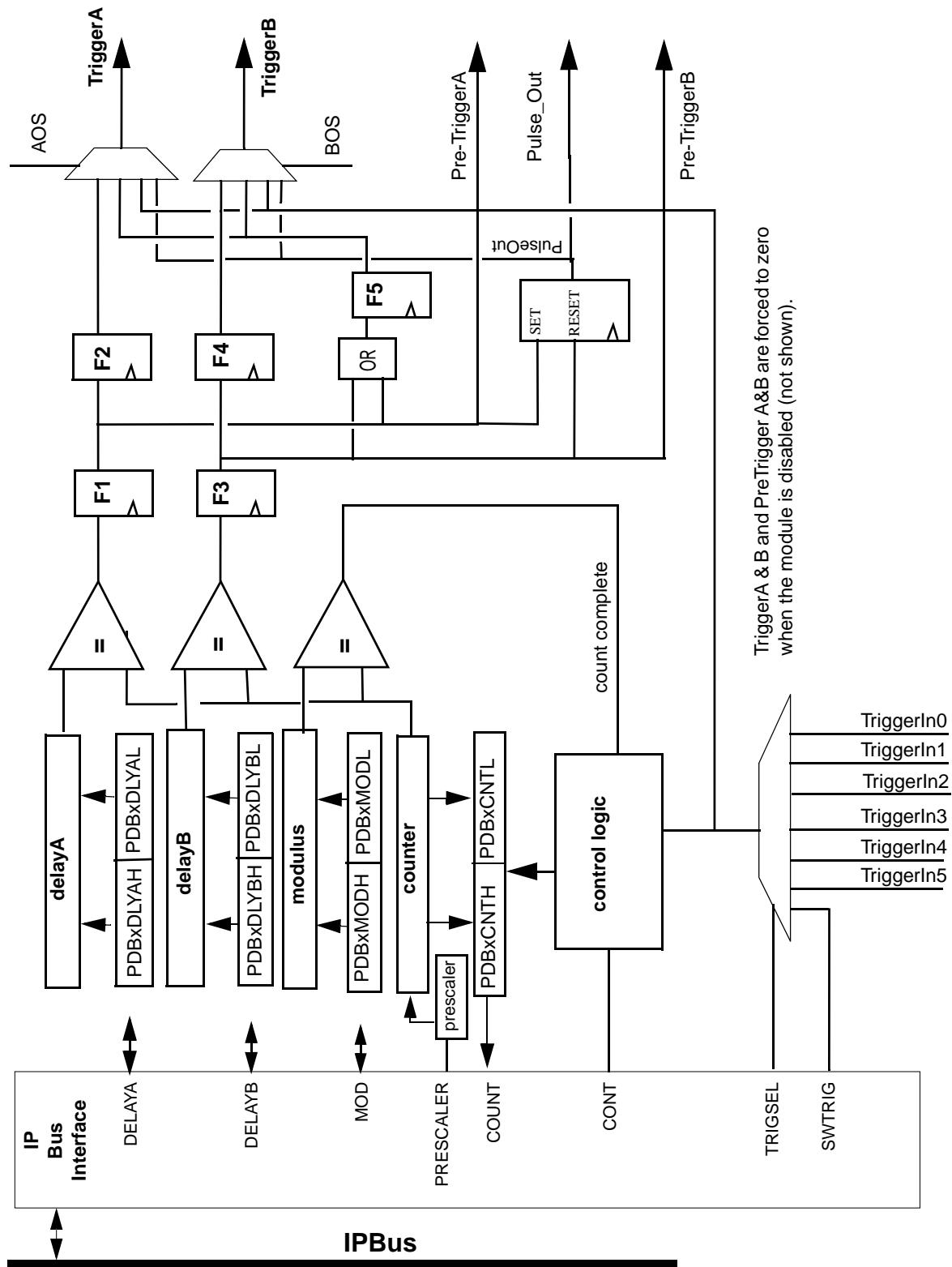


Figure 18-8. PDB Block Diagram

## 18.3 Memory Map/Register Definitions

Table 18-2. PDB Memory Map

Offset	Register	Description
0x00	PDBxCTRL1	PDB control register 1
0x01	PDBxCTRL2	PDB control register 2
0x02	PDBxDLYAH	PDB delay A register high
0x03	PDBxDLYAL	PDB delay A register low
0x04	PDBxDLYBH	PDB delay B register high
0x05	PDBxDLYBL	PDB delay B register low
0x06	PDBxMODH	PDB counter modulus register high
0x07	PDBxMODL	PDB counter modulus register low
0x08	PDBxCNTH	PDB counter value register high
0x09	PDBxCNTL	PDB counter value register low
0x0A	PDBxSCR	PDB status & control register

### 18.3.1 PDB Control Register 1 (PDBxCTRL1)

This register contains control bits for the programmable delay block.

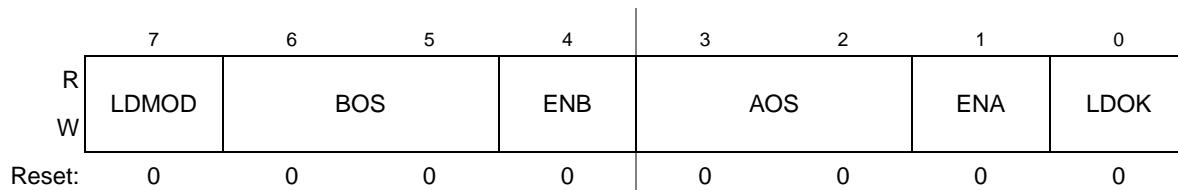


Figure 18-9. PDB Control Register 1 (PDBxCTRL1)

Table 18-3. PDBxCTRL1 Field Descriptions

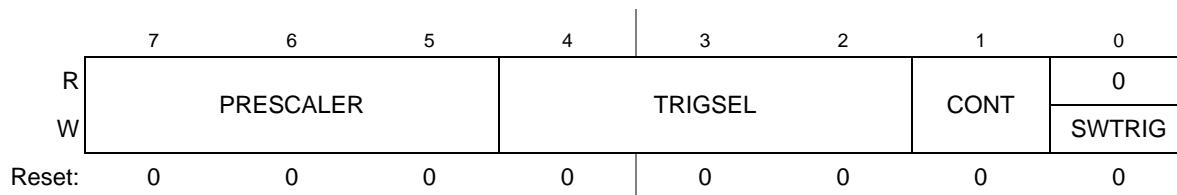
Field	Description
7 LDMOD	Load mode select 0 PDBxDLYAH, PDBxDLYAL, PDBxDLYBH, PDBxDLYBL, PDBxMODH, and PDBxMODL registers are loaded into a set of buffers and take effect immediately after logic 1 is written into LDOCK bit 1 PDBxDLYAH, PDBxDLYAL, PDBxDLYBH, PDBxDLYBL, PDBxMODH, and PDBxMODL registers are loaded into a set of buffers and take effect when the counter rolls over in continuous mode or trigger signal is received in one shot mode after logic 1 is written into LDOCK bit
6:5 BOS	Trigger B output select 00 Counter delay is bypassed 01 Trigger B is function of Delay B only 10 Trigger B is function of both Delay A and Delay B 11 Trigger B = PulseOut
4 ENB	Trigger B enable 0 Trigger B and PreTrigger B outputs are disabled (and forced to zero) 1 Trigger B and PreTrigger B outputs are enabled

**Table 18-3. PDBxCTRL1 Field Descriptions (continued)**

Field	Description
3:2 AOS	Trigger A output select 00 Counter delay is bypassed 01 Trigger A is function of Delay A only 10 Trigger A is function of both Delay A and Delay B 11 Trigger A = PulseOut
1 ENA	PreTrigger A enable 0 Trigger A and PreTrigger A outputs are disabled (and forced to zero) 1 Trigger A and PreTrigger A outputs are enabled
0 LDOK	Load OK- Writing logic 1 to this bit loads PDBxDLYAH, PDBxDLYAL, PDBxDLYBH, PDBxDLYBL, PDBxMODH, and PDBxMODL registers into a set of buffers. The buffered delayA, delayB, and modulus value take effect immediately if LDMOD = 0, or if the counter rolls over in continuous mode or trigger signal is received in one-shot mode when LDMOD = 1. Any value written to any one of these above registers, after a logic 1 is written to LDOK, is ignored until the values in these registers are loaded into the buffers. This bit is cleared when buffers are loaded. Writing logic 0 to this bit has no effect. Reading this bit can determine if the register values are loaded to the buffers and take effect.

### 18.3.2 PDB Control Register 2 (PDBxCTRL2)

This register contains PDB setting bits for the programmable delay block.

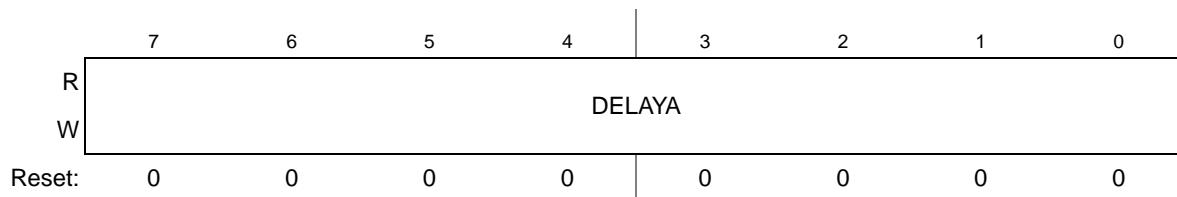
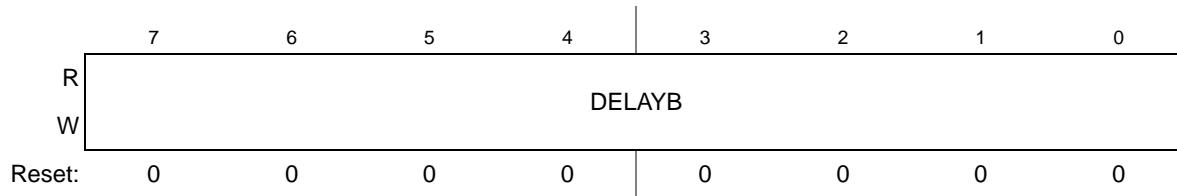
**Figure 18-10. PDB Control Register 2 (PDBxCTRL2)**

**Table 18-4. PDBxCTRL2 Field Descriptions**

Field	Description
7:5 PRESCALER	Clock prescaler select 000 Peripheral clock 001 Peripheral clock / 2 010 Peripheral clock / 4 011 Peripheral clock / 8 100 Peripheral clock / 16 101 Peripheral clock / 32 110 Peripheral clock / 64 111 Peripheral clock / 128
4:2 TRIGSEL	Input trigger select See the <a href="#">Introduction</a> section for which device-specific signals are connected to the triggers. 000 TriggerIn0 001 TriggerIn1 010 TriggerIn2 011 TriggerIn3 100 TriggerIn4 101 TriggerIn5 110 TriggerIn6 111 SWTRIG
1 CONT	Continuous mode enable 0 Module is in one-shot mode 1 Module is in continuous mode
0 SWTRIG	Software trigger- When TRIGSEL = 111 and the module is enabled, writing a one to this field triggers a reset and restart of the counter.

### 18.3.3 PDB Delay A & Delay B Registers

These registers specify the delay from assertion of TriggerIn to assertion of TriggerA and TriggerB out. This delay is only applicable if the module is enabled and the corresponding output trigger has not been bypassed. In each case, the delay is in terms of peripheral clock cycles. A delay value of 0x0000 is never reached and no output trigger occurs since the counter goes from 0x0001 to 0xFFFF.

**Figure 18-11. PDB Delay A Registers (PDBxDLYAL & PDBxDLYAH)****Figure 18-12. PDB Delay B Registers (PDBxDLYBL & PDBxDLYBH)**

### 18.3.4 PDB Modulus Registers

These registers specify the period of the counter in terms of peripheral bus cycles. When the counter reaches this value, it is reset back to one. If PDBxCTRL2[CONT] is set, the count begins a new round.

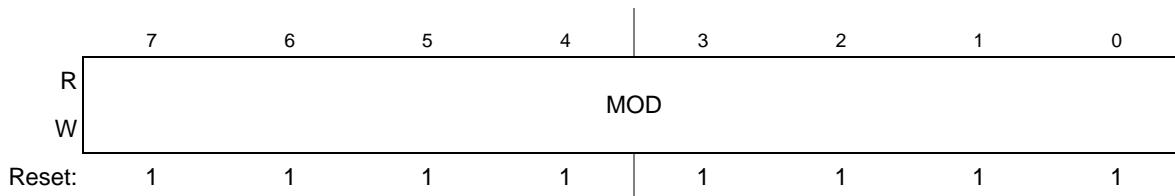


Figure 18-13. PDB Modulus Registers (PDBxMODL & PDBxMODH)

### 18.3.5 PDB Counter Registers (PDBxCNTH & PDBxCNTL)

This is a 16 bit counter which operates in up-count mode. The two read-only counter registers contain the high and low bytes of the value in the PDB counter. Reading either byte latches the contents of both bytes into a buffer where they remain latched until PDBxCNTH or PDBxCNTL is read.

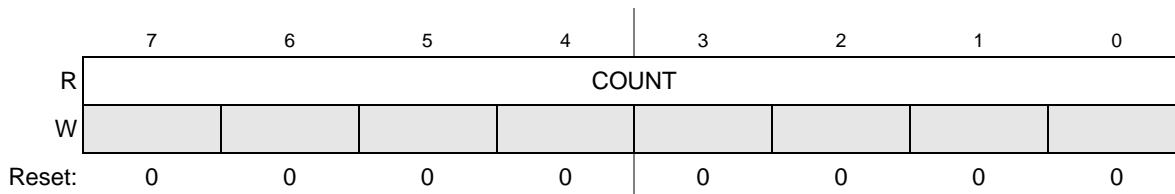


Figure 18-14. PDB Counter Registers (PDBxCNTL & PDBxCNTH)

### 18.3.6 PDB Status and Control Register (PDBxSCR)

This register contains status and control bits for the programmable delay block.

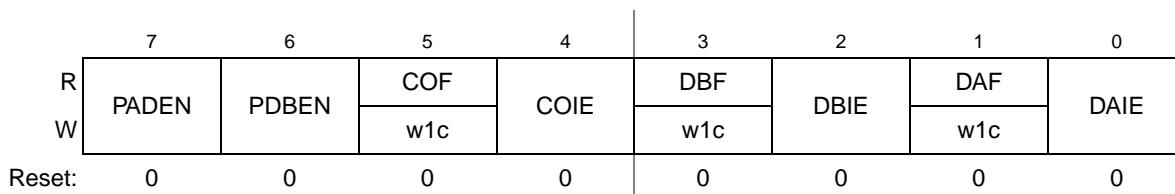


Figure 18-15. PDB Status and Control Register (PDBxSCR)

Table 18-5. PDBxSCR Field Descriptions

Field	Description
7 PADEN	Pulse output PAD enable 0 Pulse output does not control external pin 1 Pulse output is driven on external pin <b>Note:</b> Depending on clock speed, a single pulse may not propagate through the I/O pad.
6 PDBEN	PDB module enable 0 Counter is off and Pre-TriggerA, TriggerA, Pre-TriggerB, and TriggerB are low 1 Counter is enabled

**Table 18-5. PDBxSCR Field Descriptions (continued)**

Field	Description
5 COF	Counter overflow flag This bit is set when a successful compare of the counter value and modulus occurs then counter is rolled over. Clear this bit by writing logic one to it.
4 COIE	Counter overflow interrupt request enable 0 Disabled 1 Enabled
3 DBF	Delay B flag This bit is set when a successful compare of the counter value and delay B occurs. Clear this bit by writing logic one to it.
2 DBIE	Delay B successful compare interrupt request enable 0 Disabled 1 Enabled
1 DAF	Delay A flag This bit is set when a successful compare of the counter value and delay A occurs. Clear this bit by writing logic one to it.
0 DAIE	Delay A successful compare interrupt request enable 0 Disabled 1 Enabled

## 18.4 Functional Description

### 18.4.1 Miscellaneous Concerns and SoC Integration

- The purposes of this block are to:
  - Manage the delay between an external event and the time at which comparator, ADC or PGA sample(s) are taken, and
  - Generate a variable width pulse which is synchronize with PWM timer module.
- Trigger A and Trigger B are defined to be glitch free
- The PDB must correctly respond to an external trigger
- Additional trigger events, after the first, cause the counter to restart even if the counter is still counting from the previous trigger

### 18.4.2 Impacts of Using the Prescaler on Timing Resolution

Using a prescaler greater than one limits the count/delay accuracy in terms of peripheral clocks (to the modulus of the prescaler value). If the prescaler is set to divide-by-2, then the only values of total peripheral clocks that can be detected are even values. If using divide-by-4, then the only values of total peripheral clocks that can be decoded as detected are mod(4) and so forth. If you want to set a long delay value and use div 128, then you are limited to an resolution of 128 bus clocks. Therefore, use the lowest possible prescaler for a given application.

### 18.4.3 Resets

This module has a single reset input, corresponding to the chip-wide peripheral reset. After reset, all registers are set to their reset value.

### 18.4.4 Clocks

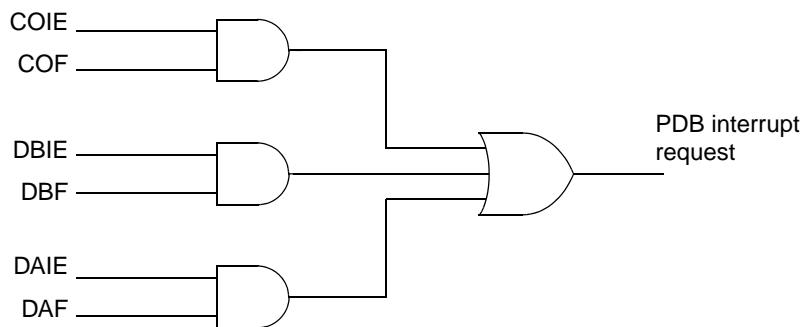
This module has a single clock input, the IP Bus peripheral clock.

### 18.4.5 Interrupts

This module has three interrupt sources

- Delay A successful compare
- Delay B successful compare
- Counter overflow

Each interrupt source has an enable bit that can block the interrupt request from being recognized by the interrupt controller. The module presents only one interrupt request to the interrupt controller as shown in [Figure 18-16](#). The interrupt request source should be determined by reading the corresponding interrupt flags in PDBxSCR.



**Figure 18-16. PDB Interrupt Request Generation**



# Chapter 19

## Programmable Gain Amplifier (S08PGAV1)

### 19.1 Introduction

Figure 19-1 shows the MC9S08MP16 Series block diagram with the PGA module highlighted.

### 19.2 Module Configurations

This section provides device-specific information for configuring the PGA on MC9S08MP16 Series.

#### 19.2.1 PGA Clock Gating

The bus clock to the PGA can be gated on and off using the PGA bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the PGA bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 19.2.2 PGA Clock Connections

On the MC9S08MP16 Series, the bus clock is connected to the PGA clk and clk\_cp inputs. The PGA clk input clocks the conversion sequencing logic and is limited to the settings defined by the PGACNTL2[ADIV] bit. The clk\_cp input is divided by  $2^{\text{PGACNTL1[CPD]}}$  to clock the charge pump. The charge pump frequency must be one-fourth of the PGA clock. Therefore, only certain CPD/ADIV settings are valid for the MC9S08MP16 Series to maintain the 1/4 relationship as shown in [Table 19-1](#).

**Table 19-1. Valid ADIV and CPD Values**

PGACNTL2[ADIV]	PGACNTL1[CPD]
00	010
01	011
10	100
11	101

#### 19.2.3 Analog Supply Connections

On the MC9S08MP16 Series devices, the VDDA and VSSA power inputs for the PGA are supplied by the V<sub>DDA</sub>/V<sub>REFH</sub> and V<sub>SSA</sub>/V<sub>REFL</sub> pins respectively.

#### 19.2.4 PGA Inputs

The PGA has one pair of analog signals, PGA+ and PGA-, available on:

- PTB3/KBI1P3/ADP3/C3IN2/PGA-
- PTB2/KBI1P2/ADP2/C1IN2/PGA+

To select these input pins for the PGA, set PGACNTL0[EN] to 1.

Calibration inputs for offset and gain are selected with CALMODE values of 10 and 11. Refer to the PGACTRL1 register description for details.

### 19.2.5 PDB1/PGA Trigger Input

The PDB1 Trigger A is used by the PGA to initiate differential-to-single-ended conversions.

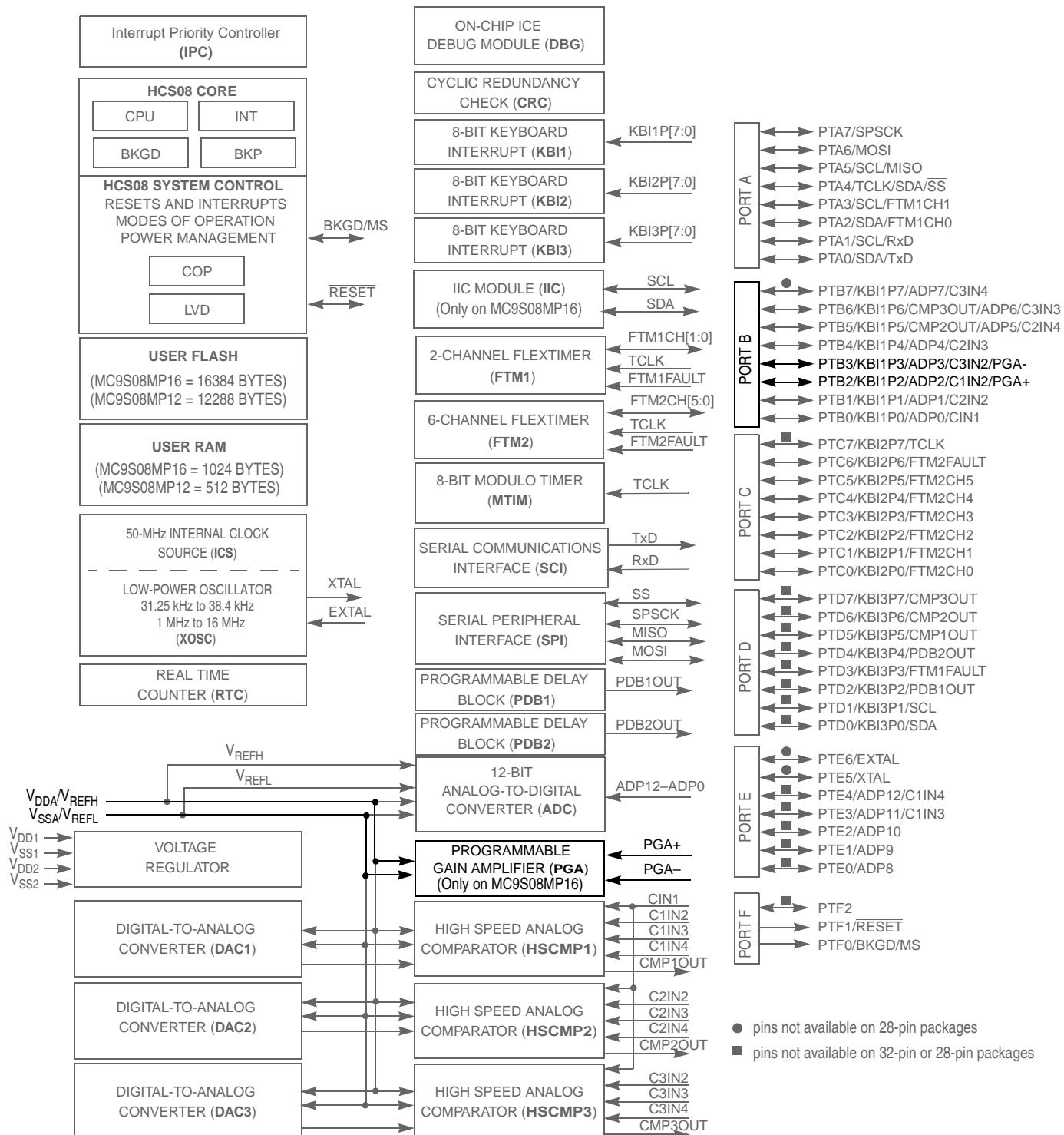
### 19.2.6 PGA Outputs

The PGA analog output is a single-ended output that connects internally to ADC input channel AD13. The PGA analog output is not available on an external pin.

The PGA trigger output is connected to the ADC hardware trigger and can be used as a hardware trigger source. In run mode this trigger initiates an ADC conversion at the completion of the PGA sample. When the PGA is disabled, the PGA is bypassed and the PDB1 Trigger A output triggers the ADC directly. For more details on ADC triggering see [Chapter 9, “Analog-to-Digital Converter \(S08ADC12V1\)”](#).

### 19.2.7 Default Reset Value for NUM\_CLK\_GS Bits

On the MC9S08MP16 Series devices, the reset value for the NUM\_CLK\_GS bits is configured for a value of 4, the nominal value for 5V PGA operation.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

**Figure 19-1. MC9S08MP16 Series Block Diagram Highlighting PGA Block and Pins**

## 19.2.8 Features

This block has the following characteristics:

- Performs differential-to-single-ended conversion of analog signals
- Software and hardware triggers are available
- Input Signal Range: Ground to VDDA
- PGA outputs driven to on-chip ADC input channels
- 1x, 2x, 4x, 8x, 16x or 32x gain
- Integrated Sample/Hold circuit
- Correlated Double Sampling (CDS) in the Diff and Diff2SE gain stages:
  - Eliminates offset error associated with these circuits
  - Reduces 1/f noise
  - Sample/Hold output can be over-sampled by the gain stages under software control
  - Automatic offset cancellation occurs during PGA startup
- Predictive CDS used in the S/H stage
- Includes additional calibration features
  - Offset calibration eliminates any errors in the internal reference used to generate the VDDA/2 output center point.
  - Gain calibration can be used to verify the gain of the overall datapath.
  - Both features require software correction of the ADC result.

## 19.3 Modes of Operation

There are several modes of operation:

- Power Down
- Startup
- Calibration
- Mission Mode: Low Power
- Mission Mode: High Power

These, as well as implications of chip power modes on the PGA, are described in the following sections.

### 19.3.1 PGA Power Down

In this mode of operation, the analog block is powered down. Trigger and pre-trigger inputs are passed unchanged to the digital outputs of the PGA. Configurations A and D in [Figure 19-7](#) are consistent with this mode of operation.

## 19.3.2 PGA Startup

There is a delay from the time the PGA is first enabled to when it is available for conversions. During the 16 PGA clk\_sh/clk\_gs periods immediately after setting PGACNTL0[EN] to 1, the PGA samples differential ground. This is equivalent to the case where PGACNTL1[CALMODE] = 10.

At the end of this period, PGASTS[STCOMP] is set, and the PGA is available for use.

## 19.3.3 PGA Calibration

The PGA supports various types of calibration, which are enabled with PGACNTL1[CALMODE]:

- Internal offset calibration via PGACNTL1[CALMODE] = 10
- External offset calibration via PGA input pins connected together to the same low-noise external DC voltage reference while PGACNTL1[CALMODE] = 00
- Internal gain calibration via PGACNTL1[CALMODE] = 11
- External gain calibration via PGA input pins connected to a differential voltage, derived from low-noise high-accuracy external DC voltage references while PGACNTL1[CALMODE] = 00

Operation of the PGA/ADC is otherwise unchanged.

### NOTE

For best performance, perform offset and gain calibrations just after PGA startup. If the PGA is disabled or any of its operating conditions is changed (VDDA, gain, power mode, first stage bypass, ADIV, CPD, NUM\_CLK\_GS, etc), repeat the offset and gain calibrations.

### 19.3.3.1 Offset Calibration - $V_{\text{offset}}$

Internal offset calibration is enabled by setting PGACNTL1[CALMODE] to 10. In this mode, both PGA inputs sample differential ground (nominally VDDA/2). From [Equation 19-13](#) this should yield a PGA output voltage of VDDA/2. When sampled and converted to a 12-bit value by the ADC, this should result in a value of 0x7FF. Any variance above or below that value represents the amount of offset present in the PGA/ADC conversion datapath.

The value of  $V_{\text{offset}}$ , coupled with  $V_{\text{gain}}$ , can be used in conjunction with [Equation 19-6](#) to cancel gain and offset errors via a simple software calculation, described in [Section 19.3.3.3, “Software Calibration”](#).

External offset calibration is another method for measuring PGA offset errors. In this method PGACNTL1[CALMODE] is set to 00 and both PGA input pins are shorted together and connected to a common low-noise DC voltage reference, external to the chip. In this mode, the PGA must operate in mission mode.

### 19.3.3.2 Gain Calibration - $V_{\text{gain}}$

Gain calibration is enabled by setting PGACNTL1[CALMODE] to “11” and using the PGA/ADC to measure the resulting value,  $V_{\text{gain}}$ . In this mode, A reference voltage of VDDA/3 is placed between the PGA inputs.

The PGA gain must be set to 1x, PGACNTL1[GAINSEL] = "00000", in order to use this particular feature. The reason for this limitation becomes obvious after once again evaluating [Equation 19-13](#):

$$V_{out} = GAIN \times (V_{in+} - V_{in-}) + VDDA/2$$

$$V_{out} = (VDDA/3) + (VDDA/2)$$

$$V_{out} = (5/6) \times VDDA = .83333 VDDA$$

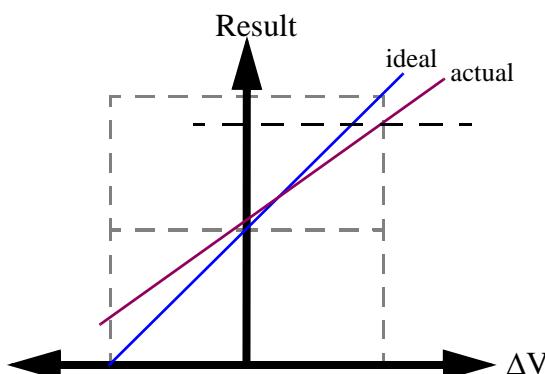
If we used 2x PGA gain, the desired PGA output voltage would exceed the supply limits.

At this ADC input voltage, an ideal device, with zero offset and zero gain error would yield 0xD54 as a result of a 12-bit conversion.

The value of  $V_{gain}$ , coupled with  $V_{offset}$  (described in the previous section), can be used in conjunction with [Equation 19-6](#) to cancel gain and offset errors via a simple software calculation, described in [Section 19.3.3.3, "Software Calibration](#).

### 19.3.3.3 Software Calibration

If we take measurements for both gain and offset errors as outlined in the previous sections, we can correct for inaccuracies in our measurement using basic interpolation. [Figure 19-2](#) illustrates both the ideal PGA/ADC transfer function, as well as a grossly exaggerated non-ideal transfer function.



**Figure 19-2. Overall ADC/PGA Transfer Function**

Assuming  $\Delta V = (V_{in+} - V_{in-})$ , the ideal transfer function for the PGA is:

$$V_{out} = GAIN \times \Delta V + VDDA/2 \quad \text{Eqn. 19-1}$$

The transfer function for the ADC in 12-bit mode is:

$$\text{Digital Result} = 0xFFFF \times V_{out}/VDDA \quad \text{Eqn. 19-2}$$

Setting GAIN=1, and combining these for the overall datapath:

$$\text{Result} = (0xFFFF/VDDA) \times [ GAIN \times \Delta V + VDDA/2 ] \quad \text{Eqn. 19-3}$$

$$\text{Result} = 0xFFFF \times (\Delta V/VDDA) + 0x7FF \quad \text{Eqn. 19-4}$$

Re-Arranging:

$$\Delta V = VDDA \times (\text{Result} - 0x7FF) / 0xFFFF$$

Eqn. 19-5

Now consider the “actual” case where we define:

$$\Delta V = A \times \text{Result} - B$$

Eqn. 19-6

Given:

- $V_{\text{offset}}$  = measurement resulting from offset calibration
- $V_{\text{gain}}$  = measurement resulting from gain calibration

we can solve for A and B in [Equation 19-6](#).

$$0 = A \times V_{\text{offset}} - B$$

Eqn. 19-7

$$VDDA/3 = A \times V_{\text{gain}} - B$$

Eqn. 19-8

Solving for A & B:

$$A = VDDA / [3 \times (V_{\text{gain}} - V_{\text{offset}})]$$

Eqn. 19-9

$$B = VDDA \times V_{\text{offset}} / [3 \times (V_{\text{gain}} - V_{\text{offset}})]$$

Eqn. 19-10

A more general calibration routine, suitable for all gain settings and external gain calibration is:

- For a given VDDA, any analog input voltage can be derived from:

$$V_{\text{input}} = V_{\text{input\_for\_gain\_calibration}} \times (\text{Result} - V_{\text{offset}}) / (V_{\text{gain}} - V_{\text{offset}})$$

Eqn. 19-11

#### 19.3.3.4 Calibration Conclusion

The PGA calibration features are intended for run-time use. Values for  $V_{\text{gain}}$  and  $V_{\text{offset}}$  should be measured via the ADC during device startup. Then A & B parameters for [Equation 19-6](#) should be pre-calculated and stored in RAM for use during operation. If the device is intended to be used in varying environmental conditions, it may be advisable to recalibrate the coefficients on a regular basis.

#### 19.3.4 PGA Mission Mode

Mission mode encompasses a number of options. These include:

- Number of gain stage clocks per conversion : PGACNTL2[NUM\_CLK\_GS]
- Low/Full Power : PGACNTL0[LP]
- Choice of Hardware or Software Trigger : PGACNTL1[TM]
- Any gain setting : PGACNTL0[GAINSEL]

## 19.4 Memory Map/Register Definitions

Table 19-2. PGA Memory Map

Address Offset	Register Name	Description
0x0	PGACNTL0	Control register 0
0x1	PGACNTL1	Control register 1
0x2	PGACNTL2	Control register 2
0x3	PGASTS	Status register

### 19.4.1 Control Register 0 (PGACNTL0)

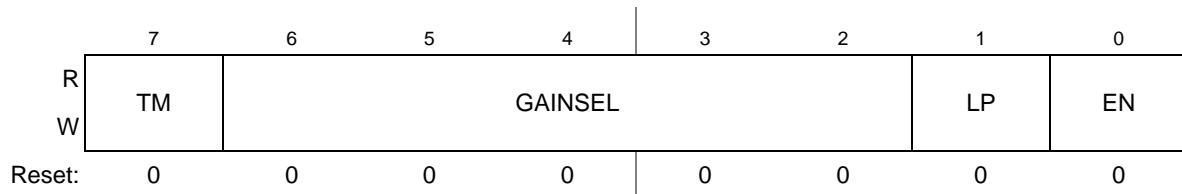


Figure 19-3. PGA Control Register 0 (PGACNTL0)

**Table 19-3. PGACNTL0 Field Descriptions**

Field	Description																																																																																																												
7 TM	Trigger mode 0 Hardware trigger mode — Conversions are initiated on the positive edge of the hardware trigger 1 Software trigger mode — Writing a 1 to PGACNTL2[SWTRIG] initiates a PGA conversion																																																																																																												
6–2 GAINSEL	Gain select. <ul style="list-style-type: none"> <li>• GAINSEL[0] selects the gain for the S/H stage of the PGA</li> <li>• GAINSEL[2:1] selects the gain for the differential gain stage of the PGA</li> <li>• GAINSEL[4:3] selects the gain for the differential to singled ended stage of the PGA</li> </ul> Selects the overall gain of the PGA. The table below outlines the effects of each possible setting.																																																																																																												
	<table border="1"> <thead> <tr> <th>#</th> <th>Gain</th> <th>GAINSEL</th> <th>DSE gain</th> <th>DIFF gain</th> <th>S/H gain</th> </tr> </thead> <tbody> <tr><td>1<sup>1</sup></td><td>1x</td><td>00000</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>2<sup>1</sup></td><td>2x</td><td>00001</td><td>1</td><td>1</td><td>2</td></tr> <tr><td>3</td><td>2x</td><td>00010</td><td>1</td><td>2</td><td>1</td></tr> <tr><td>4<sup>1</sup></td><td>4x</td><td>00011</td><td>1</td><td>2</td><td>2</td></tr> <tr><td>5<sup>1</sup></td><td>3x</td><td>00100</td><td>1</td><td>3</td><td>1</td></tr> <tr><td>6<sup>1</sup></td><td>6x</td><td>00101</td><td>1</td><td>3</td><td>2</td></tr> <tr><td>7</td><td>4x</td><td>00110</td><td>1</td><td>4</td><td>1</td></tr> <tr><td>8<sup>1</sup></td><td>8x</td><td>00111</td><td>1</td><td>4</td><td>2</td></tr> <tr><td>9</td><td>2x</td><td>01000</td><td>2</td><td>1</td><td>1</td></tr> <tr><td>10</td><td>4x</td><td>01001</td><td>2</td><td>1</td><td>2</td></tr> <tr><td>11</td><td>4x</td><td>01010</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>12</td><td>8x</td><td>01011</td><td>2</td><td>2</td><td>2</td></tr> <tr><td>13</td><td>6x</td><td>01100</td><td>2</td><td>3</td><td>1</td></tr> <tr><td>14<sup>1</sup></td><td>12x</td><td>01101</td><td>2</td><td>3</td><td>2</td></tr> <tr><td>15</td><td>8x</td><td>01110</td><td>2</td><td>4</td><td>1</td></tr> <tr><td>16<sup>1</sup></td><td>16x</td><td>01111</td><td>2</td><td>4</td><td>2</td></tr> </tbody> </table>							#	Gain	GAINSEL	DSE gain	DIFF gain	S/H gain	1 <sup>1</sup>	1x	00000	1	1	1	2 <sup>1</sup>	2x	00001	1	1	2	3	2x	00010	1	2	1	4 <sup>1</sup>	4x	00011	1	2	2	5 <sup>1</sup>	3x	00100	1	3	1	6 <sup>1</sup>	6x	00101	1	3	2	7	4x	00110	1	4	1	8 <sup>1</sup>	8x	00111	1	4	2	9	2x	01000	2	1	1	10	4x	01001	2	1	2	11	4x	01010	2	2	1	12	8x	01011	2	2	2	13	6x	01100	2	3	1	14 <sup>1</sup>	12x	01101	2	3	2	15	8x	01110	2	4	1	16 <sup>1</sup>	16x	01111	2	4	2
#	Gain	GAINSEL	DSE gain	DIFF gain	S/H gain																																																																																																								
1 <sup>1</sup>	1x	00000	1	1	1																																																																																																								
2 <sup>1</sup>	2x	00001	1	1	2																																																																																																								
3	2x	00010	1	2	1																																																																																																								
4 <sup>1</sup>	4x	00011	1	2	2																																																																																																								
5 <sup>1</sup>	3x	00100	1	3	1																																																																																																								
6 <sup>1</sup>	6x	00101	1	3	2																																																																																																								
7	4x	00110	1	4	1																																																																																																								
8 <sup>1</sup>	8x	00111	1	4	2																																																																																																								
9	2x	01000	2	1	1																																																																																																								
10	4x	01001	2	1	2																																																																																																								
11	4x	01010	2	2	1																																																																																																								
12	8x	01011	2	2	2																																																																																																								
13	6x	01100	2	3	1																																																																																																								
14 <sup>1</sup>	12x	01101	2	3	2																																																																																																								
15	8x	01110	2	4	1																																																																																																								
16 <sup>1</sup>	16x	01111	2	4	2																																																																																																								
	<sup>1</sup> Recommended setting for this gain value																																																																																																												
1 LP	Power mode 0 High power (maximum performance) 1 Low power (limited performance)																																																																																																												
0 EN	Module enable 0 PGA disabled and powered down (output shorted to VSSA) 1 PGA is powered & enabled																																																																																																												

## 19.4.2 Control Register 1 (PGACNTL1)

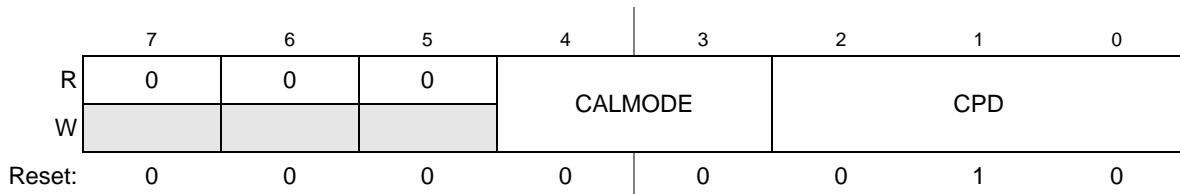


Figure 19-4. PGA Control Register 1 (PGACNTL1)

Table 19-4. PGACNTL1 Field Descriptions

Field	Description
7–5	Reserved, must be cleared.
4–3 CALMODE	Calibration mode 00 Mission mode 01 Reserved. 10 Offset calibration — inputs sample differential ground 11 Gain calibration — inputs sample an internal DC reference between the PGA inputs
2–0 CPD	Charge pump divisor. The programmable gain amplifier uses an internal charge pump for bias purposes. There is a required operating range of the charge pump. Program the CPD bits to ensure that the charge pump frequency is within the valid range. Refer to the PGA section in the device data sheet.  <b>Charge pump clock frequency = Input clock / <math>2^{CPD}</math></b> <span style="float: right;"><i>Eqn. 19-12</i></span>  The reset value of this field is 0x2, resulting in a divisor of 4.

## 19.4.3 Control Register 2 (PGACNTL2)

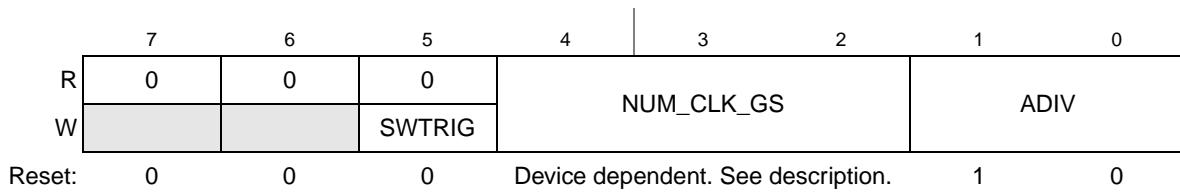


Figure 19-5. PGA Control Register 2 (PGACNTL2)

Table 19-5. PGACNTL2 Field Descriptions

Field	Description
7–6	Reserved, must be cleared.
5 SWTRIG	Software trigger. When software triggering is enabled by setting PGACNTL0[TM], writing a one to this bit initiates a conversion. <b>Note:</b> NUM_CLK_GS must be set to a minimum value of 001 when using software triggers. This is required for the signal to propagate through both gain stages of the PGA. This bit always reads as zero.

**Table 19-5. PGACNTL2 Field Descriptions (continued)**

Field	Description
4–2 NUM_CLK_GS	Number of clk_gs pulses per conversion. This parameter controls how many times the gain and diff-SE stages of the PGA are clocked per conversion (NUM_CLK_GS+1). Refer to the <a href="#">Section 19.5.5, “PGA Clock Requirements”</a> and <a href="#">Section 19.5.6, “Effects on ADC Latency”</a> for additional details. The reset value of this field is device specific. See the <a href="#">Introduction</a> section for the reset value for this particular device.
1–0 ADIV	Clock divide select. Determines the clock rate for the PGA clock. These two bits must be set to the same value as ADCCFG[ADIV] in the associated ADC module, so that this clock runs at the same rate as the ADC clock. The two clocks are derived from the same source. 00 Input clock 01 Input clock ÷ 2 10 Input clock ÷ 4 11 Input clock ÷ 8

#### 19.4.4 Status Register (PGASTS)

	7	6	5	4		3	2	1	0
R	0	0	0	0	0	0	RUNNING	STCOMP	
W									
Reset:	0	0	0	0	0	0	0	0	0

**Figure 19-6. PGA Status Register (PGASTS)**

**Table 19-6. PGASTS Field Descriptions**

Field	Description
7–2	Reserved, must be cleared.
1 RUNNING	PGA run sequence underway. 0 The PGA state machine is inactive 1 The PGA is performing a differential-to-single-ended conversion
0 STCOMP	Startup complete. 0 The PGA is disabled, or the PGA startup sequence is incomplete. The PGA is not available for conversions. 1 The PGA is enabled and has completed its startup sequence

## 19.5 Functional Description

### 19.5.1 Transfer Function

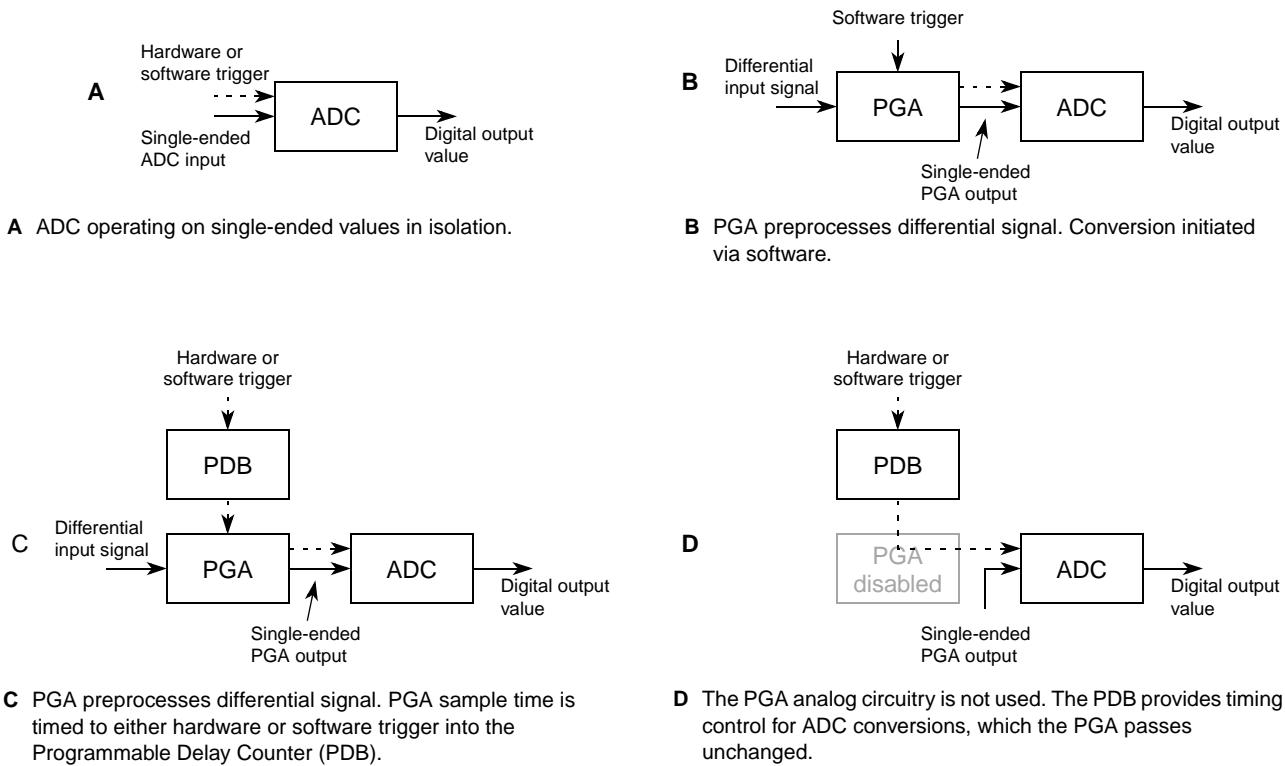
The PGA differential amplifier is a switched-capacitor (SC) circuit that amplifies a differential input signal and converts it to single-ended output. The mathematical description of the output voltage is given by:

$$V_{\text{out}} = \text{GAIN}(V_{\text{in}+} - V_{\text{in}-}) + \left(\frac{V_{\text{DDA}}}{2}\right) \quad \text{Eqn. 19-13}$$

where  $(V_{\text{in}+} - V_{\text{in}-})$  is the differential input voltage,  $V_{\text{DDA}}$  is the power supply voltage, GAIN defines the amplifier gain. Also, differential stages have both high immunity to virtual ground (or differential ground) variations as well as to finite gain and offset operational amplifier non-idealities.

## 19.5.2 Options for On-Chip Analog Conversions

The PGA is designed to operate as part of a larger system designed for precise conversion of analog values. Some possible configurations of on-chip components are shown in [Figure 19-7](#). In the figure, PDB is the programmable delay block. This is a digital function designed to generate precisely-timed hardware triggers for the ADC and PGA blocks.



**Figure 19-7. Analog Sub-System Configuration Options**

## 19.5.3 PGA Prerequisites

When using the PGA, you must:

- Use long ADC sample times:  $\text{ADCCFG[ADLSMP]} = 1$
- Configure the ADC to use hardware triggering:  $\text{ADCSC2[ADTRG]} = 1$
- If supported, configure the ADC to export a continuous clock source for use by the PGA:  $\text{ADCSC2[ECC]} = 1$
- Configure the PGA's PGACNTL2[ADIV] to the same as the ADC's ADCCFG[ADIV]
- Do not exceed sampling intervals as specified in [Table 19-7](#)

## 19.5.4 Analog Block Diagram

[Figure 19-8](#) illustrates the structure of the PGA analog block. A differential voltage is presented for conversion across  $V_{in+}$  and  $V_{in-}$ . During mission mode operation, these are passed unchanged through the

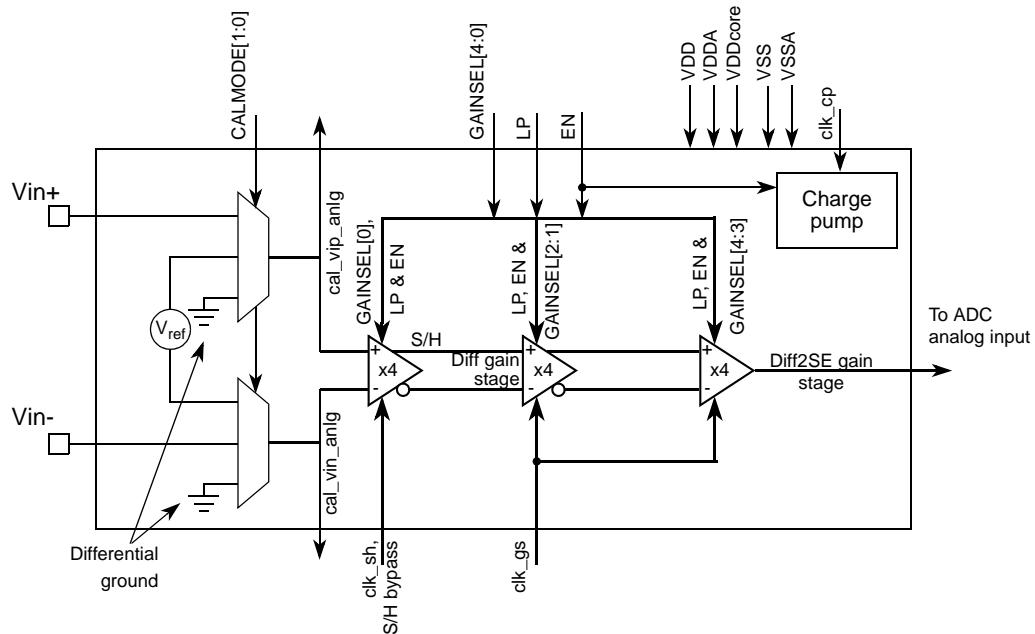
input muxes and arrive at the sample/hold of the PGA. The sampling window of the PGA can be precisely placed, and has a minimum sampling aperture of 1us.

Figure 19-8 shows the maximum allowable gain for each stage of the PGA. Each stage also supports lower values. The S/H stage can be programmed for gains of 1x and 2x. The differential and differential-to-single ended (diff-SE) gain stages can be programmed for 1x, 2x, 3x or 4x gain.

The PGA design uses a switched capacitor architecture. Clock details are provided in Section 19.5.5, “[PGA Clock Requirements](#)”.

The input muxes can be used to place zero volts across ( $V_{in+} - V_{in-}$ ), which allows measurement of offset errors associated with the S/H and virtual ground. Offset errors can then be compensated for in software.

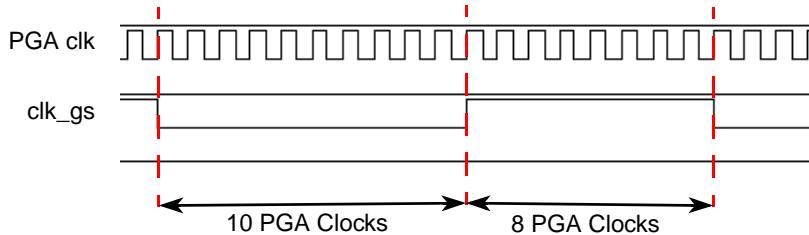
The charge pump manages bias levels in the PGA. It is enabled automatically whenever the PGA is enabled. The recommended charge pump frequency is the PGA/ADC clock divided by four. Higher frequencies inject noise into the PGA output, while lower frequencies slow PGA response time to the point its output is corrupted. Program the CPD bits to ensure that the charge pump frequency is at the recommended frequency.  $clk_{cp}$  in Figure 19-8 is divided down to generate the charge pump clock based on the value of PGACNTL1[CPD].



**Figure 19-8. Analog Block Diagram of the PGA**

## 19.5.5 PGA Clock Requirements

The maximum PGA clock rate is 8 MHz (4 MHz in low power mode), therefore the minimum pulse width high is 1  $\mu$ s.

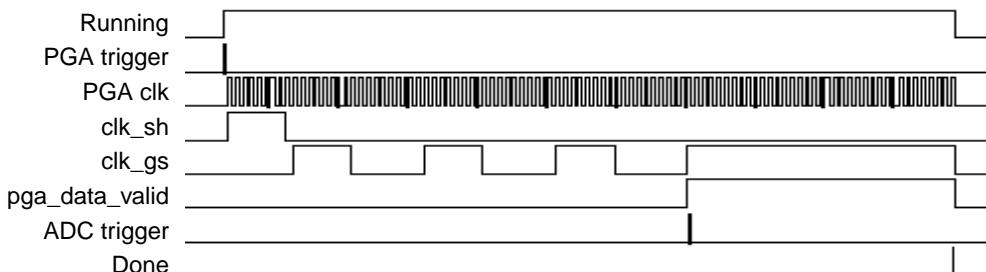


**Figure 19-9. PGA Clock Generation**

The sample/hold stage of the PGA samples when `clk_sh` is high. A high transition on `clk_sh` is initiated via a hardware or software trigger into the PGA. The latency between hardware trigger and beginning of the sampling window is typically only a few clock periods. This is minimized when the system is running at maximum frequency. In this case, latency is less than 0.1 $\mu$ s between trigger event and start of the sampling window. This allows precise placement of the sampling window.

Figure 19-10 shows one possible PGA conversion sequence. From top to bottom, waveforms are:

- Running      Indicates that the PGA is converting a signal.
- PGA Trigger      This is the signal to the PGA that an analog value needs to be processed. This pulse can be the result of either hardware or software triggers.
- PGA clk      The PGA clock is restarted from the off condition when a trigger event is detected. `clk_sh` and `clk_gs` are derived from the PGA clk. PGA clk is derived from the same clock source used by the corresponding ADC channel. This may be asynchronous to the standard peripheral clock.
- `clk_sh`      Is high only during the sample interval for the S/H stage.
- `clk_gs`      When operating with short, non-periodic signals, there are offset and noise benefits to clocking the gain and diff-SE stages of the PGA multiple times for each sample taken by the S/H stage. In the case shown, these stages are clocked four times. During the fourth period, the `clk_gs` signal is held high for a total of 36 PGA clock cycles.
- ADC trigger      Issued at the beginning of the last `clk_gs` HIGH phase, this signal is used as a hardware trigger to the ADC, which must sample the PGA output while `clk_gs` remains high.
- Done      Signifies that the PGA has completed operation. The output of the PGA is no longer guaranteed to be valid, and the PGA clocks are shut down.



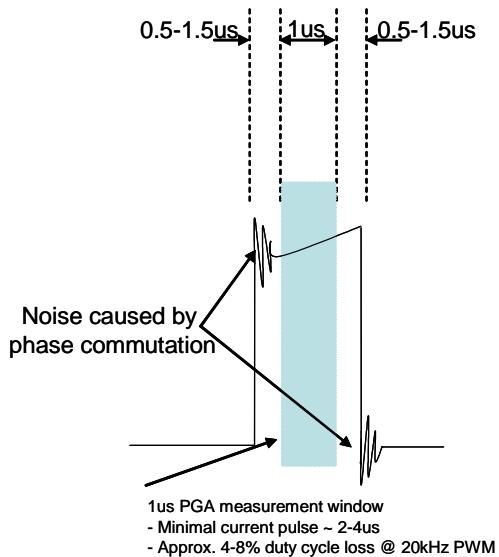
**Figure 19-10. PGA Clock Sequencing 4:1**

## NOTE

When operated in conjunction with the PGA, the ADC must be programmed to use its long sampling time. This is 23.5 ADC clock periods long.

Because the PGA and ADC clocks operate at the same frequency (although out of phase), there is more than sufficient time for the ADC to sample the PGA output during the final clk\_gs HIGH phase, which is 36 PGA clock periods long.

PGACNTL2[NUM\_CLK\_GS] specifies how many times the gain and diff-SE stages are clocked per conversion. Overclocking these stages results in improved offset and noise performance when sampling waveforms such as that shown in [Figure 19-11](#). This is an example of a non-periodic waveform of short duration. In this case, the S/H stage can obtain a valid sample which is centered within the 2 $\mu$ s signal being sampled. Clocking the gain and diff-SE multiple times allows those stages to use correlated-double-sampling techniques on the output of the S/H to reduce their offset error.



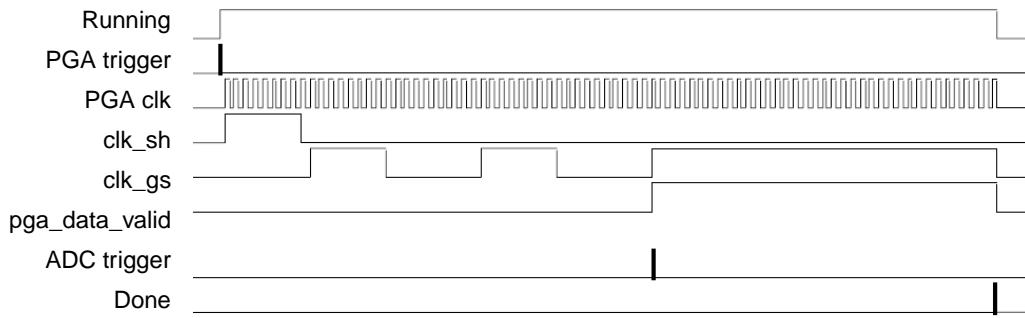
**Figure 19-11. Targeted Sample Window**

Overclocking is not required when using the PGA/ADC to perform continuous conversions of signals which meet the bandwidth limitations listed in the device data sheet.

A minimum of 2x overclocking should be used if not continuously sampling the input signal. This is because two clocks are required to propagate the signal through both the gain and diff-SE stages. This restriction applies even for low bandwidth signals.

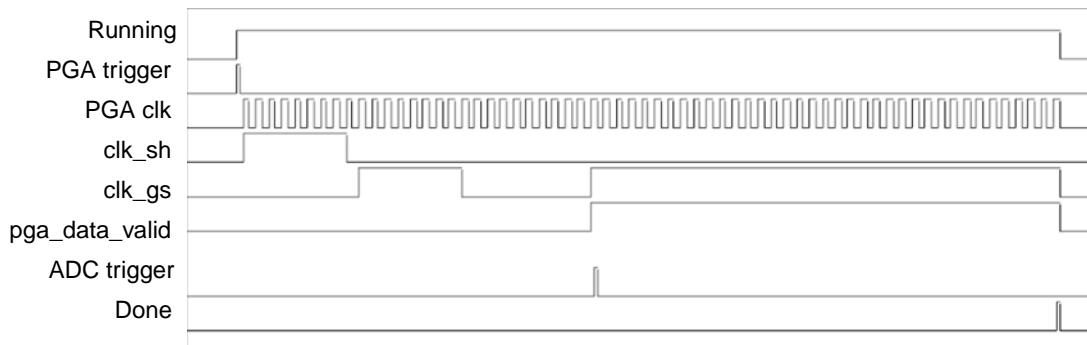
[Figure 19-10](#) illustrates the case where PGACNTL2[NUM\_CLK\_GS] has a value of 0x3, which corresponds to four assertions of clk\_gs per conversion.

[Figure 19-12](#) illustrates the case where PGACNTL2[NUM\_CLK\_GS] has a value of 0x2, which corresponds to three assertions of clk\_gs per conversion.



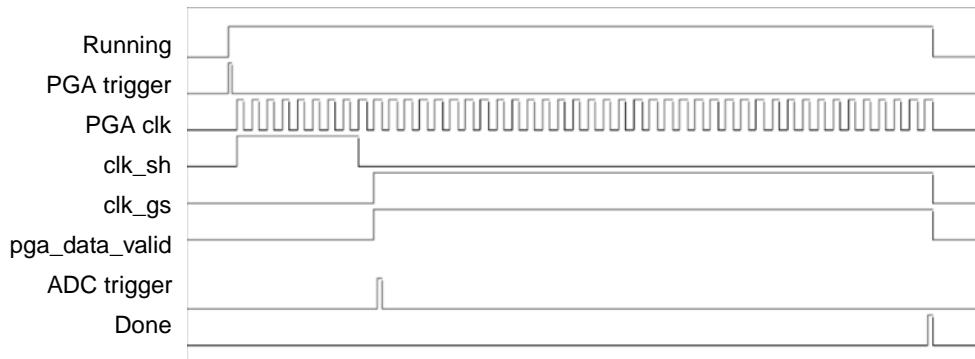
**Figure 19-12. PGA Clock Sequencing 3:1**

Figure 19-13 illustrates the case where PGACNTL2[NUM\_CLK\_GS] has a value of 0x1, which corresponds to two assertions of clk\_gs per conversion.



**Figure 19-13. PGA Clock Sequencing 2:1**

Figure 19-14 illustrates the case where PGACNTL2[NUM\_CLK\_GS] has a value of 0x0, which corresponds to one assertion of clk\_gs per conversion.



**Figure 19-14. PGA Clock Sequencing 1:1**

## 19.5.6 Effects on ADC Latency

Inspection of [Figure 19-10](#), [Figure 19-12](#), [Figure 19-13](#) and [Figure 19-14](#) show that the PGA adds approximately:

$$12 + (18 \times \text{NUM\_CLK\_GS}) \quad \text{Eqn. 19-14}$$

ADC/PGA clock periods to the latency of the ADC by itself. So, at a value of  $\text{NUM\_CLK\_GS} = 3$ , with an 8 MHz ADC clock rate, we have:

$$[12 + (18 \times 3)] \times 125 \text{ ns} = 8.25 \mu\text{s additional latency} \quad \text{Eqn. 19-15}$$

Assuming a 12-bit conversion with long sample times by the ADC (required), the ADC conversion time is 43 ADC clock cycles + 5 pclk cycles. With an 8 MHz ADC clock and 25 MHz pclk, this is 5.575  $\mu\text{s}$ . So the total conversion time from triggerIn of the PGA to ADC conversion complete is 13.825  $\mu\text{s}$ .

If  $\text{NUM\_CLK\_GS}$  is set to zero (valid when continuously sampling bandwidth-limited signals), then the additional latency is only twelve ADC/PGA clock periods, or 1.5 $\mu\text{s}$ . The total conversion time from triggerIn of the PGA to ADC conversion complete is only 7.075  $\mu\text{s}$ .

To minimize conversion latency,  $\text{NUM\_CLK\_GS}$  should be set to the minimum value that yields the accuracy required for a given application. To maximize conversion accuracy, set  $\text{NUM\_CLK\_GS}$  to the maximum value allowed by the required conversion rate. [Table 19-7](#) below summarizes latency effects and conversion rates as a function of  $\text{NUM\_CLK\_GS}$  for the case when the peripheral clock rate is 25 MHz and the ADC/PGA clock rate is 8 MHz.

**Table 19-7. PGA/ADC Conversion Times and Rates  
(25 MHz Peripheral Bus Clock & 8 MHz ADC/PGA)**

#	NUM_CLK_GS	ADC conversion time ( $\mu\text{s}$ )	Additional PGA clocks required	Latency adder due to PGA ( $\mu\text{s}$ )	Total conversion time ( $\mu\text{s}$ )	Maximum PGA/ADC conversions per second
1	N/A	5.575	0	0	5.575	179,372
2	0	5.575	12	1.5	7.075	141,343
3	1	5.575	30	3.75	9.325	107,239
4	2	5.575	48	6	11.575	86,393
5	3	5.575	66	8.25	13.825	72,333
6	4	5.575	84	10.5	16.075	62,208
7	5	5.575	102	12.75	18.325	54,570
8	6	5.575	120	15	20.575	48,603
9	7	5.575	138	17.25	22.825	43,812

Conversion accuracy depends on a combination of input bandwidth and  $\text{NUM\_CLK\_GS}$ . Higher bandwidth signals result in the higher variation in S/H output from one sample to another, and more clocks are required by GS and diff-SE to output an accurate signal.

## **19.5.7 ADC Triggers**

## **19.5.8 Interrupts**

The PGA does not generate any interrupts to the CPU. ADC conversion complete interrupts can be used to process converted values.

## **19.5.9 Reset Considerations**

The PGA is inactive during device reset. All PGA registers are reset to their default values.

# Chapter 20

## Real-Time Counter (S08RTCV1)

### 20.1 Introduction

The real-time counter (RTC) consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, two clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake up from low power modes without the need of external components.

### 20.2 Module Configurations

This section provides device-specific information for configuring the RTC on MC9S08MP16 Series.

#### 20.2.1 RTC Clock Gating

The bus clock to the RTC can be gated on and off using the RTC bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the RTC bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 20.2.2 RTC/PDB Trigger

This RTC can be enabled as a trigger by the PDB1 and PDB2 modules by appropriate setting of the TRIGSEL bits. When enabled, the PDB will be triggered every time RTCCNT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the PDB.

#### 20.2.3 RTC Clock Sources

The RTC module on MC9S08MP16 Series can be clocked from the ICSIRCLK, OSCOUT or the LPO. ICSEERCLK is not available as a source to the RTC in this MCU.

#### 20.2.4 RTC Modes of Operation

All clock sources are available in all modes except stop2. The LPO can be enabled as the clock source of the RTC in stop2.



## 20.2.5 Features

Features of the RTC module include:

- 8-bit up-counter
  - 8-bit modulo match limit
  - Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler with selectable binary-based and decimal-based divider values
  - 1-kHz internal low-power oscillator (LPO)
  - External clock (ERCLK)
  - 32-kHz internal clock (IRCLK)

## 20.2.6 Modes of Operation

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

### 20.2.6.1 Wait Mode

The RTC continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the RTC can bring the MCU out of wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during wait mode.

### 20.2.6.2 Stop Modes

The RTC continues to run in stop2 or stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

The LPO clock can be used in stop2 and stop3 modes. ERCLK and IRCLK clocks are only available in stop3 mode.

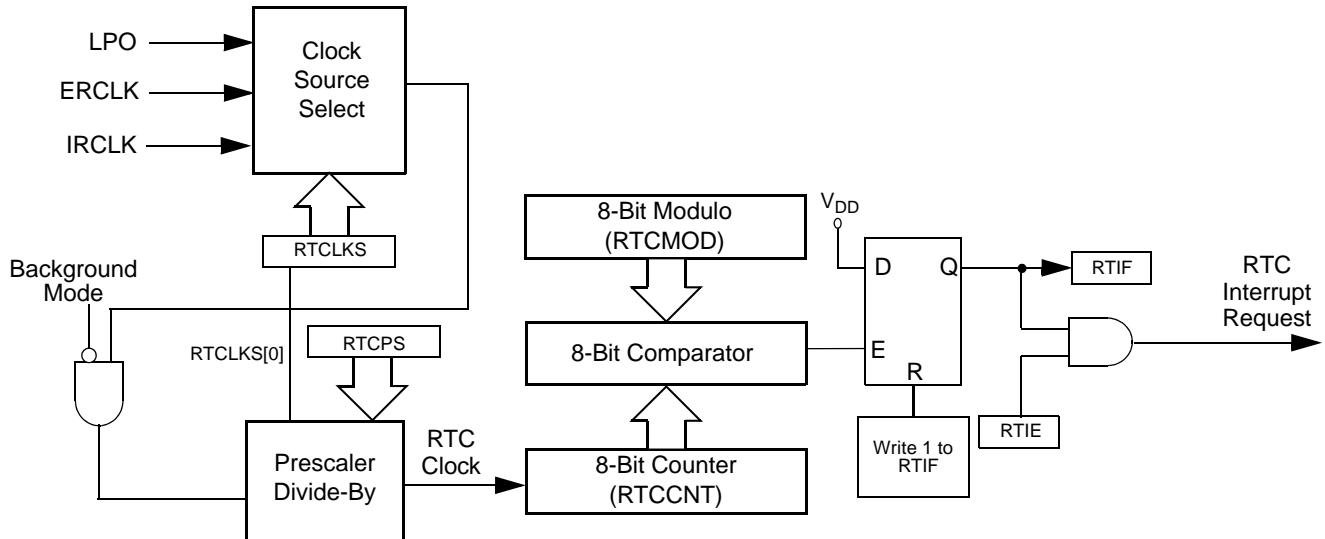
Power consumption is lower when all clock sources are disabled, but in that case, the real-time interrupt cannot wake up the MCU from stop modes.

### 20.2.6.3 Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

## 20.2.7 Block Diagram

The block diagram for the RTC module is shown in [Figure 20-1](#).



**Figure 20-1. Real-Time Counter (RTC) Block Diagram**

## 20.3 External Signal Description

The RTC does not include any off-chip signals.

## 20.4 Register Definition

The RTC includes a status and control register, an 8-bit counter register, and an 8-bit modulo register.

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

[Table 20-1](#) is a summary of RTC registers.

**Table 20-1. RTC Register Summary**

Name		7	6	5	4	3	2	1	0							
RTCSC	R	RTIF	RTCLKS		RTIE	RTCPs										
	W															
RTCCNT	R	RTCCNT														
	W															
RTCMOD	R	RTCMOD														
	W															

## 20.4.1 RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).

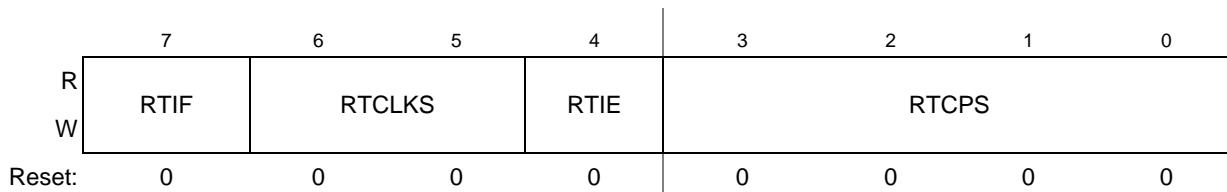


Figure 20-2. RTC Status and Control Register (RTCSC)

Table 20-2. RTCSC Field Descriptions

Field	Description
7 RTIF	Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF. 0 RTC counter has not reached the value in the RTC modulo register. 1 RTC counter has reached the value in the RTC modulo register.
6–5 RTCLKS	Real-Time Clock Source Select. These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. When selecting a clock source, ensure that the clock source is properly enabled (if applicable) to ensure correct operation of the RTC. Reset clears RTCLKS. 00 Real-time clock source is the 1-kHz low power oscillator (LPO) 01 Real-time clock source is the external clock (ERCLK) 1x Real-time clock source is the internal clock (IRCLK)
4 RTIE	Real-Time Interrupt Enable. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE. 0 Real-time interrupt requests are disabled. Use software polling. 1 Real-time interrupt requests are enabled.
3–0 RTCPS	Real-Time Clock Prescaler Select. These four read/write bits select binary-based or decimal-based divide-by values for the clock source. See Table 20-3. Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS.

Table 20-3. RTC Prescaler Divide-by values

RTCLKS[0]	RTCPS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Off	$2^3$	$2^5$	$2^6$	$2^7$	$2^8$	$2^9$	$2^{10}$	1	2	$2^2$	10	$2^4$	$10^2$	$5 \times 10^2$	$10^3$
1	Off	$2^{10}$	$2^{11}$	$2^{12}$	$2^{13}$	$2^{14}$	$2^{15}$	$2^{16}$	$10^3$	$2 \times 10^3$	$5 \times 10^3$	$10^4$	$2 \times 10^4$	$5 \times 10^4$	$10^5$	$2 \times 10^5$

## 20.4.2 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

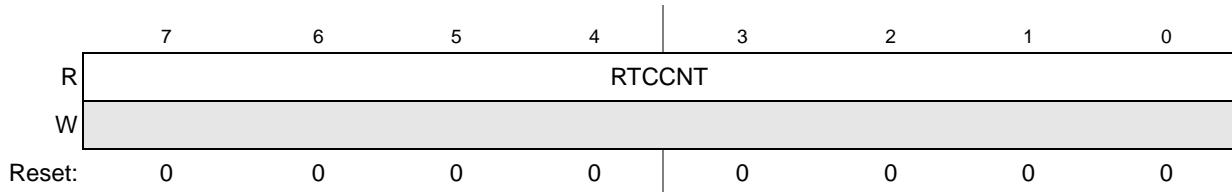


Figure 20-3. RTC Counter Register (RTCCNT)

Table 20-4. RTCCNT Field Descriptions

Field	Description
7:0 RTCCNT	RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00.

## 20.4.3 RTC Modulo Register (RTCMOD)

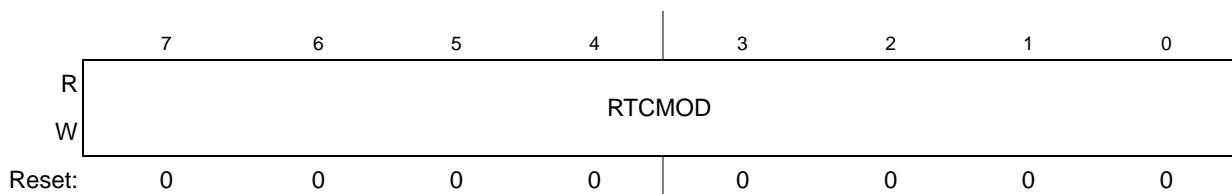


Figure 20-4. RTC Modulo Register (RTCMOD)

Table 20-5. RTCMOD Field Descriptions

Field	Description
7:0 RTCMOD	RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00.

## 20.5 Functional Description

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

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

RTCPS and the RTCLKS[0] bit select the desired divide-by value. If a different value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. [Table 20-6](#) shows different prescaler period values.

**Table 20-6. Prescaler Period**

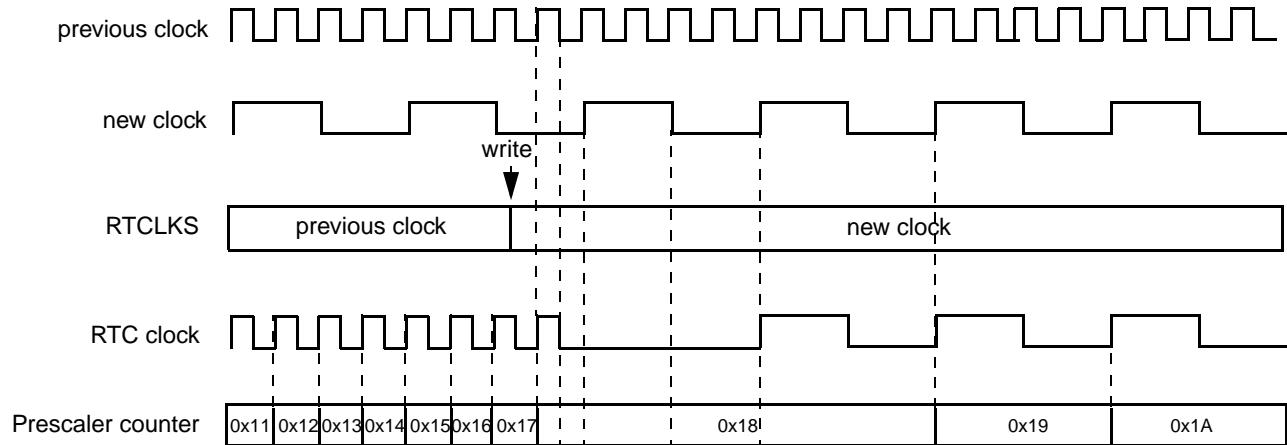
RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 µs	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 µs	31.25 ms
1001	2 ms	2 ms	62.5 µs	62.5 ms
1010	4 ms	5 ms	125 µs	156.25 ms
1011	10 ms	10 ms	312.5 µs	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

The RTC modulo register (RTCMOD) allows the compare value to be set to any value from 0x00 to 0xFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x00 and continues counting. The real-time interrupt flag (RTIF) is set when a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00.

The RTC allows for an interrupt to be generated when RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

### 20.5.1 RTC Source Clock Switching

When the RTC clock source is selected (a new value is written to RTCLKS[1:0] bits), the prescaler and main counters are frozen during the clock switching to avoid that the RTC clock has some glitch. An example of the clock switching is shown in [Figure 20-5](#).

**Figure 20-5. RTC Source Clock Switching**

According to [Figure 20-5](#) the switching time in the worst case (the new value is written to RTCLKS a "little" after of the previous clock rising edge and the next previous clock falling edge occurs a "little" after of the new clock rising edge) is:

$$T_{worst\_case} = (1.5 * T_{previous\_clock}) + (3 * T_{new\_clock})$$

where  $T_{previous\_clock}$  is the previous clock period and  $T_{new\_clock}$  is the new clock period.

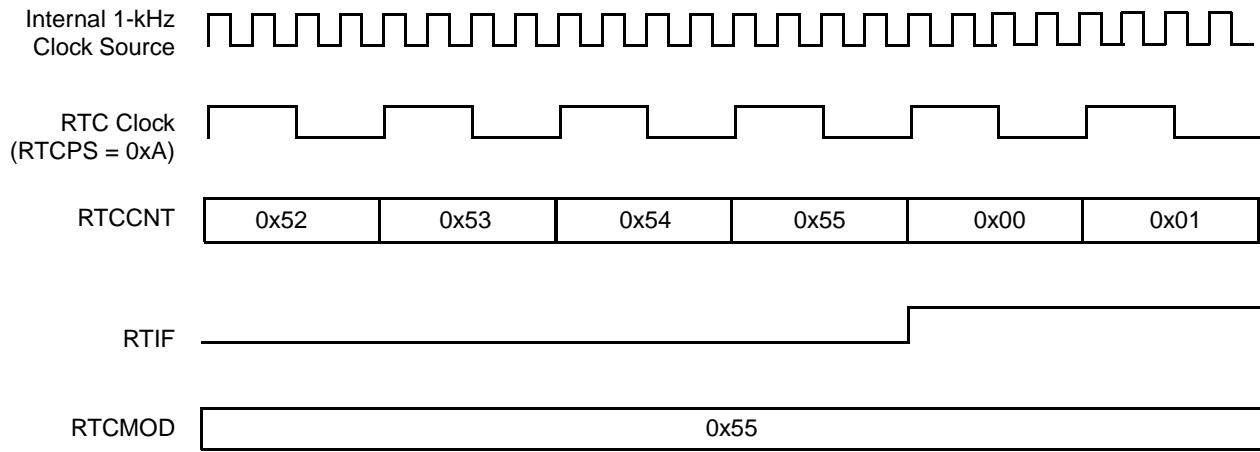
And the switching time in the best case (the new value is written to RTCLKS a "little" before of the previous clock rising edge and the next previous clock falling edge occurs a "little" before of the new clock rising edge) is:

$$T_{best\_case} = (0.5 * T_{previous\_clock}) + (2 * T_{new\_clock})$$

where  $T_{previous\_clock}$  is the previous clock period and  $T_{new\_clock}$  is the new clock period.

## 20.5.2 RTC Operation Example

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

**Figure 20-6. RTC Counter Overflow Example**

In the example of [Figure 20-6](#), the selected clock source is the 1-kHz internal oscillator clock source. The prescaler (RTCPS) is set to 0xA or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

## 20.6 Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```

/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;

*****
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*****/

```

## Real-Time Counter (S08RTCV1)

```
#pragma TRAP_PROC
void RTC_ISR(void)
{
    /* Clear the interrupt flag */
    RTCSC.byte = RTCSC.byte | 0x80;
    /* RTC interrupts every 1 Second */
    Seconds++;
    /* 60 seconds in a minute */
    if (Seconds > 59){
        Minutes++;
        Seconds = 0;
    }
    /* 60 minutes in an hour */
    if (Minutes > 59){
        Hours++;
        Minutes = 0;
    }
    /* 24 hours in a day */
    if (Hours > 23){
        Days++;
        Hours = 0;
    }
}
```

# **Chapter 21**

## **Serial Communications Interface (S08SCIV4)**

### **21.1 Introduction**

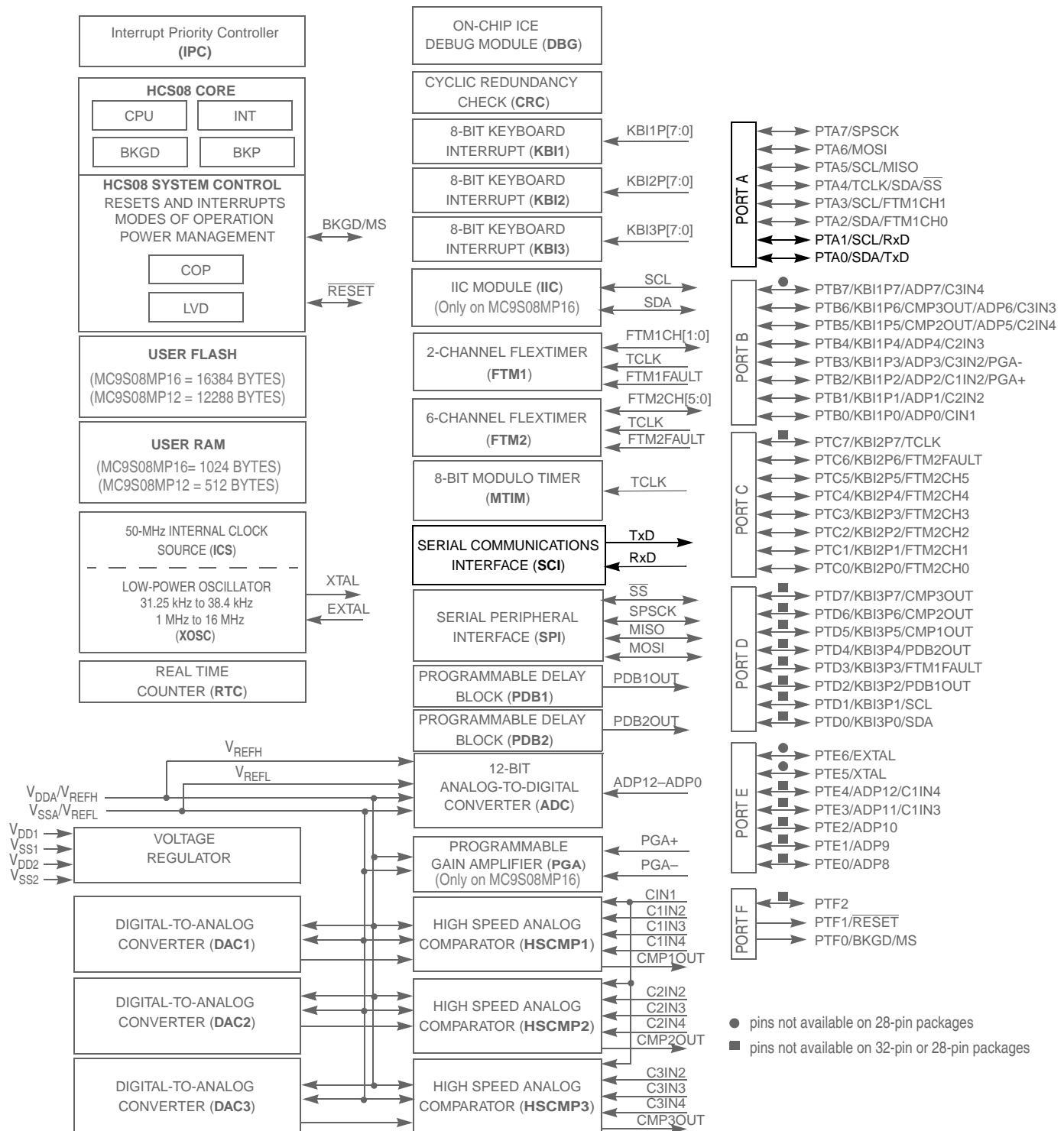
Figure 21-1 shows the MC9S08MP16 Series block diagram with the SCI module highlighted.

### **21.2 Module Configurations**

This section provides device-specific information for configuring the SCI modules on MC9S08MP16 Series.

#### **21.2.1 SCI Clock Gating**

The bus clock to the SCI can be gated on and off using the SCI bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the SCI bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

V<sub>DD2</sub> pad is tied internally on 32-pin and 28-pin packages,

V<sub>SS2</sub> pad is tied internally on 28-pin packages

Figure 21-1. MC9S08MP16 Series Block Diagram Highlighting SCI Block and Pins

## 21.2.2 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

## 21.2.3 Modes of Operation

See [Section 21.4, “Functional Description,”](#) For details concerning SCI operation in these modes:

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

## 21.2.4 Block Diagram

Figure 21-2 shows the transmitter portion of the SCI.

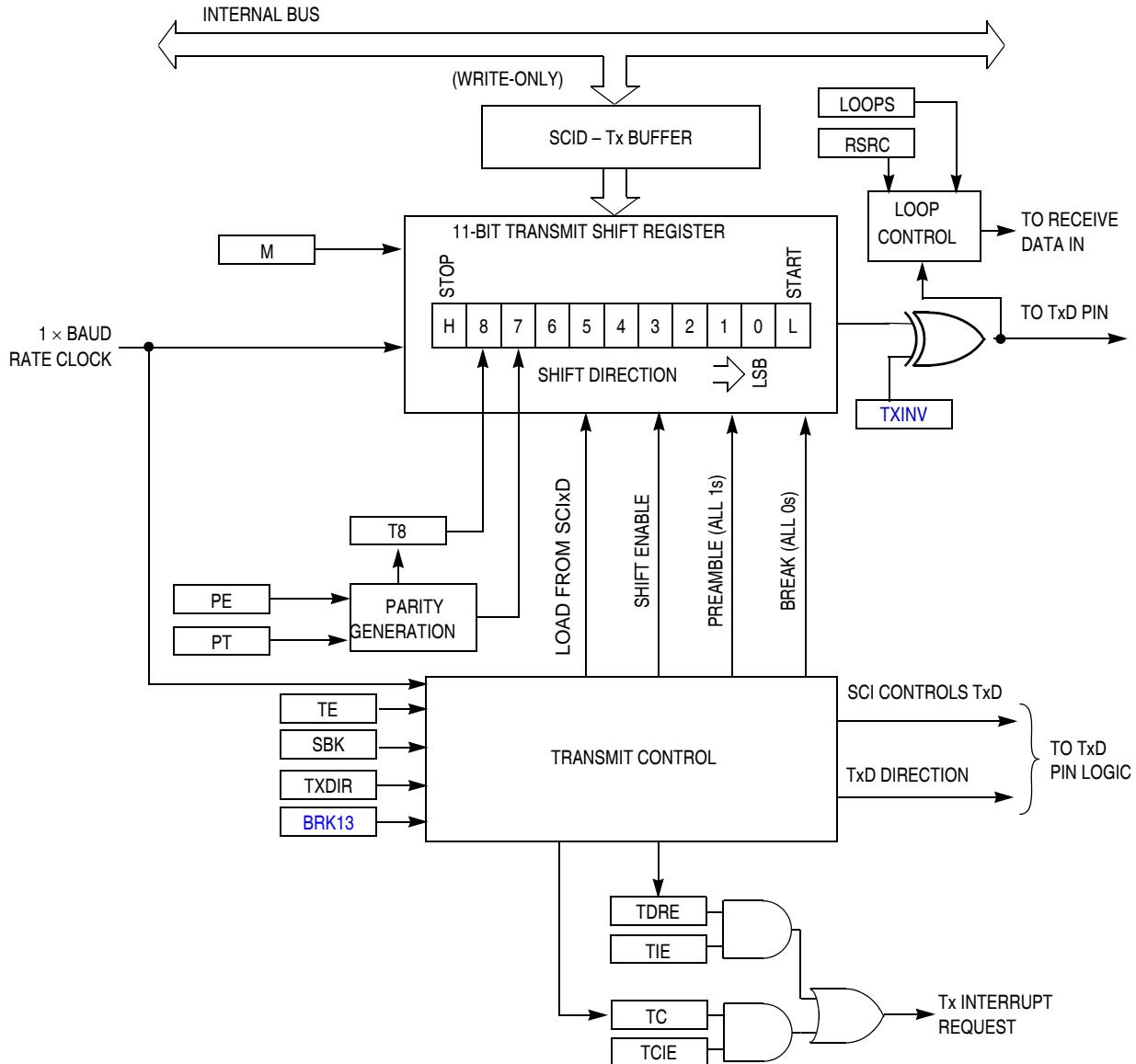


Figure 21-2. SCI Transmitter Block Diagram

Figure 21-3 shows the receiver portion of the SCI.

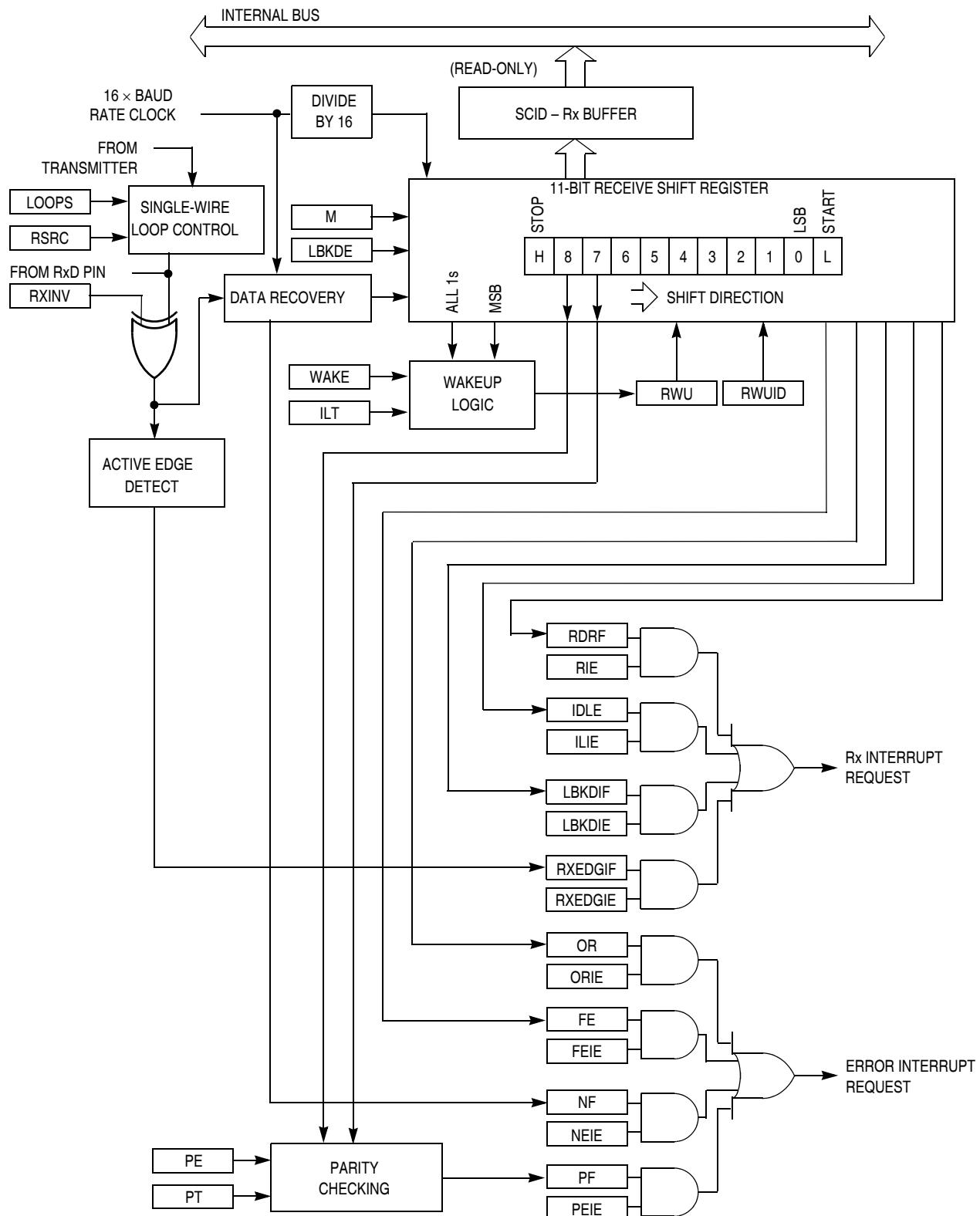


Figure 21-3. SCI Receiver Block Diagram

## 21.3 Register Definition

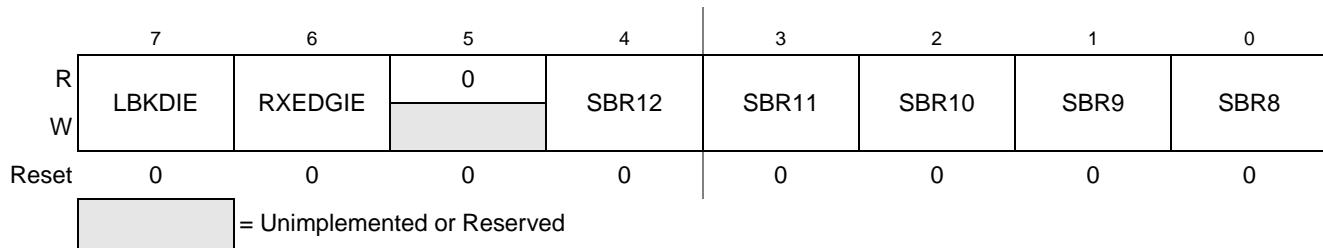
The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

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.

### 21.3.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).



**Figure 21-4. SCI Baud Rate Register (SCIxBDH)**

**Table 21-1. SCIxBDH Field Descriptions**

Field	Description
7 LBKDIIE	<b>LIN Break Detect Interrupt Enable (for LBKDIF)</b> 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	<b>RxD Input Active Edge Interrupt Enable (for RXEDGIF)</b> 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	<b>Baud Rate Modulo Divisor</b> — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in <a href="#">Table 21-2</a> .

	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 21-5. SCI Baud Rate Register (SCIxBDL)

Table 21-2. SCIxBDL Field Descriptions

Field	Description
7:0 SBR[7:0]	<b>Baud Rate Modulo Divisor</b> — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in <a href="#">Table 21-1</a> .

### 21.3.2 SCI Control Register 1 (SCIxC1)

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 21-6. SCI Control Register 1 (SCIxC1)

Table 21-3. SCIxC1 Field Descriptions

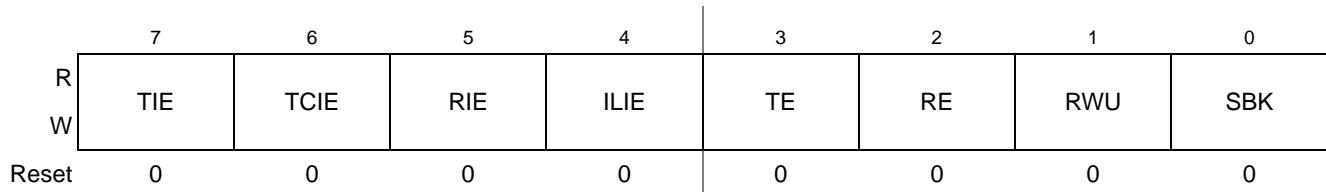
Field	Description
7 LOOPS	<b>Loop Mode Select</b> — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See <b>RSRC</b> bit.) RxD pin is not used by SCI.
6 SCISWAI	<b>SCI Stops in Wait Mode</b> 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	<b>Receiver Source Select</b> — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	<b>9-Bit or 8-Bit Mode Select</b> 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.

**Table 21-3. SCIx C1 Field Descriptions (continued)**

Field	Description
3 WAKE	<b>Receiver Wakeup Method Select</b> — Refer to <a href="#">Section 21.4.3.2, “Receiver Wakeup Operation”</a> for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	<b>Idle Line Type Select</b> — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to <a href="#">Section 21.4.3.2.1, “Idle-Line Wakeup”</a> for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	<b>Parity Enable</b> — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	<b>Parity Type</b> — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

### 21.3.3 SCI Control Register 2 (SCIx C2)

This register can be read or written at any time.

**Figure 21-7. SCI Control Register 2 (SCIx C2)****Table 21-4. SCIx C2 Field Descriptions**

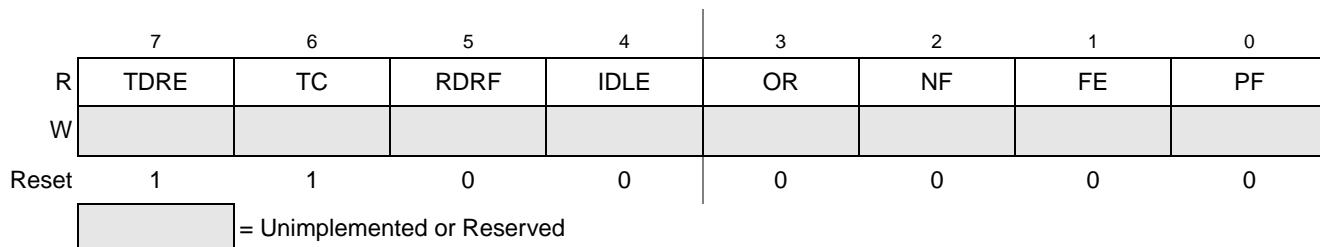
Field	Description
7 TIE	<b>Transmit Interrupt Enable (for TDRE)</b> 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	<b>Transmission Complete Interrupt Enable (for TC)</b> 0 Hardware interrupts from TC disabled (use polling). 1 Hardware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

**Table 21-4. SCIxC2 Field Descriptions (continued)**

Field	Description
3 TE	<b>Transmitter Enable</b> 0 Transmitter off. 1 Transmitter on. TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to <a href="#">Section 21.4.2.1, "Send Break and Queued Idle"</a> for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.
2 RE	<b>Receiver Enable</b> — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	<b>Receiver Wakeup Control</b> — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to <a href="#">Section 21.4.3.2, "Receiver Wakeup Operation"</a> for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	<b>Send Break</b> — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to <a href="#">Section 21.4.2.1, "Send Break and Queued Idle"</a> for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

### 21.3.4 SCI Status Register 1 (SCIxS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

**Figure 21-8. SCI Status Register 1 (SCIxS1)**

**Table 21-5. SCIxS1 Field Descriptions**

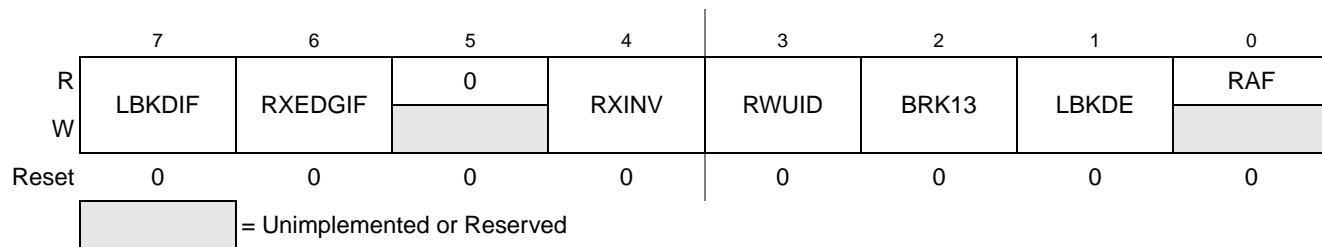
<b>Field</b>	<b>Description</b>
7 TDRE	<b>Transmit Data Register Empty Flag</b> — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE = 1 and then write to the SCI data register (SCIxD). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	<b>Transmission Complete Flag</b> — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SCIxS1 with TC = 1 and then doing one of the following three things: <ul style="list-style-type: none"> <li>• Write to the SCI data register (SCIxD) to transmit new data</li> <li>• Queue a preamble by changing TE from 0 to 1</li> <li>• Queue a break character by writing 1 to SBK in SCIxC2</li> </ul>
5 RDRF	<b>Receive Data Register Full Flag</b> — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). To clear RDRF, read SCIxS1 with RDRF = 1 and then read the SCI data register (SCIxD). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	<b>Idle Line Flag</b> — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SCIxS1 with IDLE = 1 and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	<b>Receiver Overrun Flag</b> — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with OR = 1 and then read the SCI data register (SCIxD). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	<b>Noise Flag</b> — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No noise detected. 1 Noise detected in the received character in SCIxD.

**Table 21-5. SCIxS1 Field Descriptions (continued)**

Field	Description
1 FE	<b>Framing Error Flag</b> — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIxS1 with FE = 1 and then read the SCI data register (SCIxD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	<b>Parity Error Flag</b> — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No parity error. 1 Parity error.

### 21.3.5 SCI Status Register 2 (SCIxS2)

This register has one read-only status flag.

**Figure 21-9. SCI Status Register 2 (SCIxS2)****Table 21-6. SCIxS2 Field Descriptions**

Field	Description
7 LBKDIF	<b>LIN Break Detect Interrupt Flag</b> — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	<b>RxD Pin Active Edge Interrupt Flag</b> — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV <sup>1</sup>	<b>Receive Data Inversion</b> — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	<b>Receive Wake Up Idle Detect</b> — RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	<b>Break Character Generation Length</b> — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

Table 21-6. SCIxS2 Field Descriptions (continued)

Field	Description
1 LBKDE	<b>LIN Break Detection Enable</b> — LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	<b>Receiver Active Flag</b> — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxID input not idle).

<sup>1</sup> Setting RXINV inverts the RxID 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.

### 21.3.6 SCI Control Register 3 (SCIxC3)

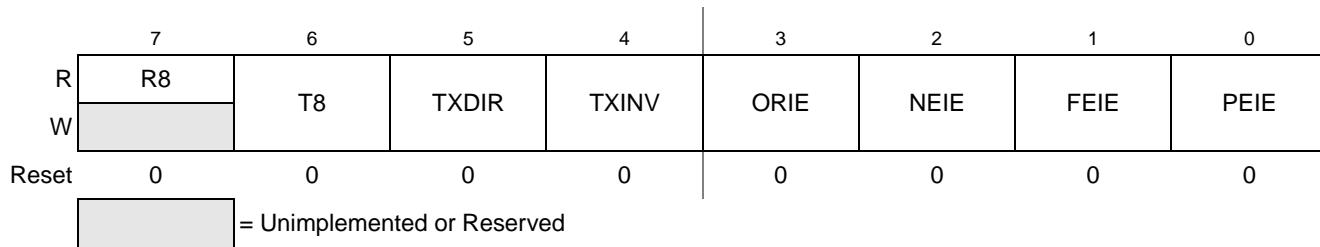


Figure 21-10. SCI Control Register 3 (SCIxC3)

Table 21-7. SCIxC3 Field Descriptions

Field	Description
7 R8	<b>Ninth Data Bit for Receiver</b> — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCIxD register. When reading 9-bit data, read R8 before reading SCIxD because reading SCIxD completes automatic flag clearing sequences which could allow R8 and SCIxD to be overwritten with new data.
6 T8	<b>Ninth Data Bit for Transmitter</b> — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written.
5 TXDIR	<b>TxD Pin Direction in Single-Wire Mode</b> — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

**Table 21-7. SCIx C3 Field Descriptions (continued)**

Field	Description
4 TXINV <sup>1</sup>	<b>Transmit Data Inversion</b> — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	<b>Overrun Interrupt Enable</b> — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	<b>Noise Error Interrupt Enable</b> — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	<b>Framing Error Interrupt Enable</b> — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	<b>Parity Error Interrupt Enable</b> — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

<sup>1</sup> Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

### 21.3.7 SCI Data Register (SCIx D)

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	0

**Figure 21-11. SCI Data Register (SCIx D)**

## 21.4 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.

### 21.4.1 Baud Rate Generation

As shown in [Figure 21-12](#), the clock source for the SCI baud rate generator is the bus-rate clock.

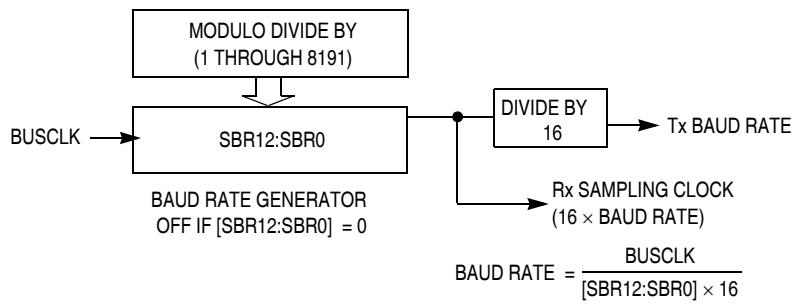


Figure 21-12. SCI Baud Rate Generation

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  $\pm 4.5$  percent for 8-bit data format and about  $\pm 4$  percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

## 21.4.2 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 21-2](#).

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIxC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

#### 21.4.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

**Table 21-8. 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

#### 21.4.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 21-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to [Section 21.4.5.1, “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 ( $\text{RDRF} = 1$ ), it gets the data from the receive data register by reading  $\text{SCIxD}$ . The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to [Section 21.4.4, "Interrupts and Status Flags"](#) for more details about flag clearing.

### 21.4.3.1 Data Sampling Technique

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

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

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

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

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

### 21.4.3.2 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  $\text{SCIxC2}$ . When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant

message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

#### **21.4.3.2.1 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.

#### **21.4.3.2.2 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.

### **21.4.4 Interrupts and Status Flags**

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIXD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while RDRF = 1 and then reading SCIxD.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while IDLE = 1 and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled (RE = 1).

## 21.4.5 Additional SCI Functions

The following sections describe additional SCI functions.

### 21.4.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCIxD.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

### 21.4.5.2 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.

### 21.4.5.3 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.

### 21.4.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.



# Chapter 22

## Serial Peripheral Interface (S08SPIV4)

### 22.1 Introduction

Figure 22-1 shows the MC9S08MP16 Series block diagram with the SPI modules highlighted.

### 22.2 Module Configurations

This section provides device-specific information for configuring the SPI modules on MC9S08MP16 Series.

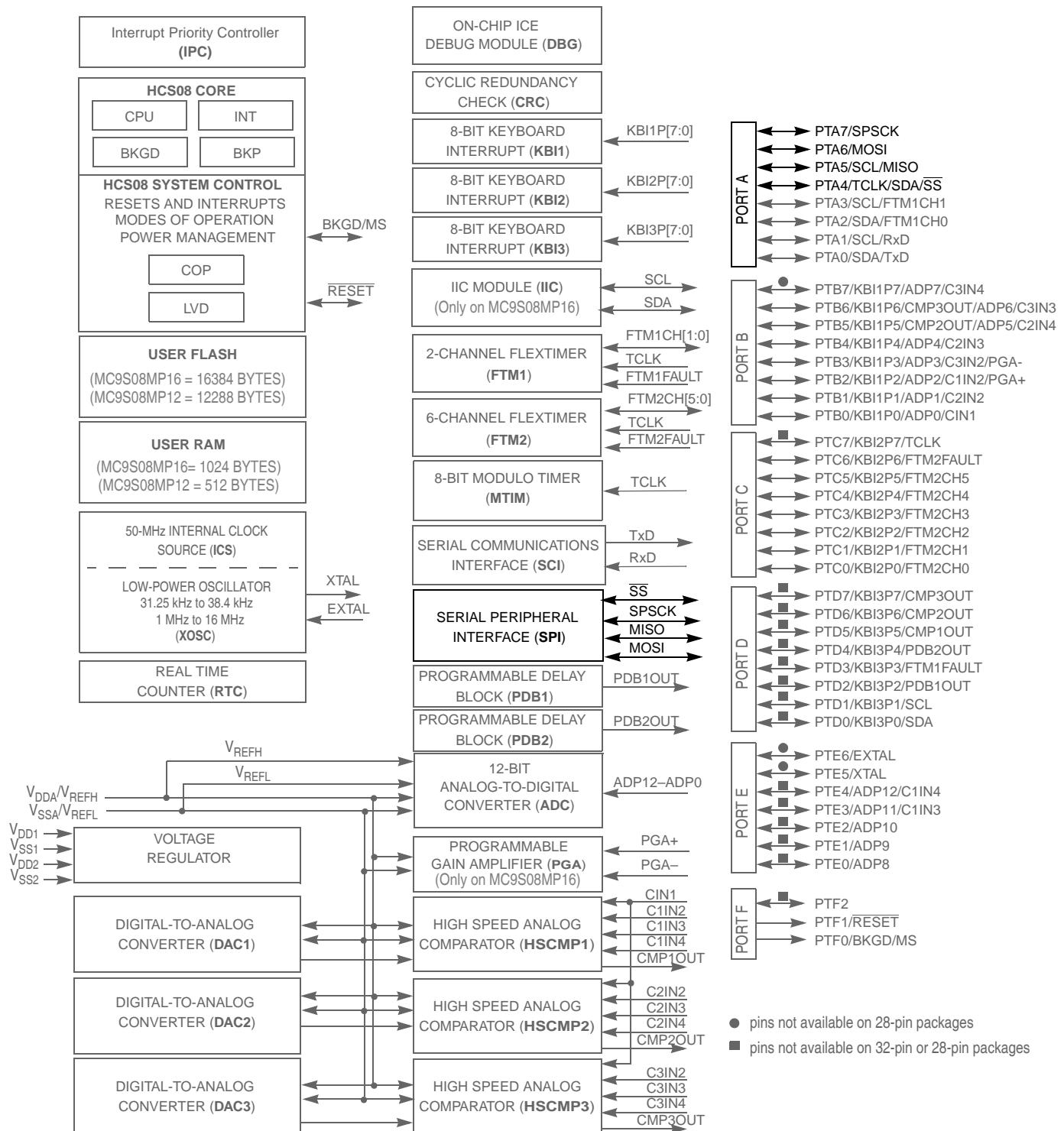
#### 22.2.1 SPI Port Configuration Information

To configure the SPI for high speed operation (operating frequency greater than 5 MHz), the input filters on the SPI port pins must be disabled by clearing SOPT1[SPIFE] and enabling the high output drive strength on the port pins corresponding to the SPI function pins. Doing so allows data transfers to occur at higher frequency, but at a risk of introducing noise during data transfers.

By default, the input filters on the SPI port pins are disabled (SPIFE=0).

#### 22.2.2 SPI Clock Gating

The bus clock to the SPI can be gated on and off using the SPI bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the SPI bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.



**Notes:** When PTF1 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

When PTF0 is configured as BKGD, pin becomes bi-directional.

$V_{DD2}$  pad is tied internally on 32-pin and 28-pin packages,

$V_{SS2}$  pad is tied internally on 28-pin packages

Figure 22-1. MC9S08MP16 Series Block Diagram Highlighting SPI Block and Pin

## 22.2.3 Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- Programmable transmit bit rate
- Double-buffered transmit and receive
- Serial clock phase and polarity options
- Slave select output
- Selectable MSB-first or LSB-first shifting

## 22.2.4 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

### 22.2.4.1 SPI System Block Diagram

[Figure 22-2](#) shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ( $\overline{SS}$  pin). In this system, the master device has configured its  $\overline{SS}$  pin as an optional slave select output.

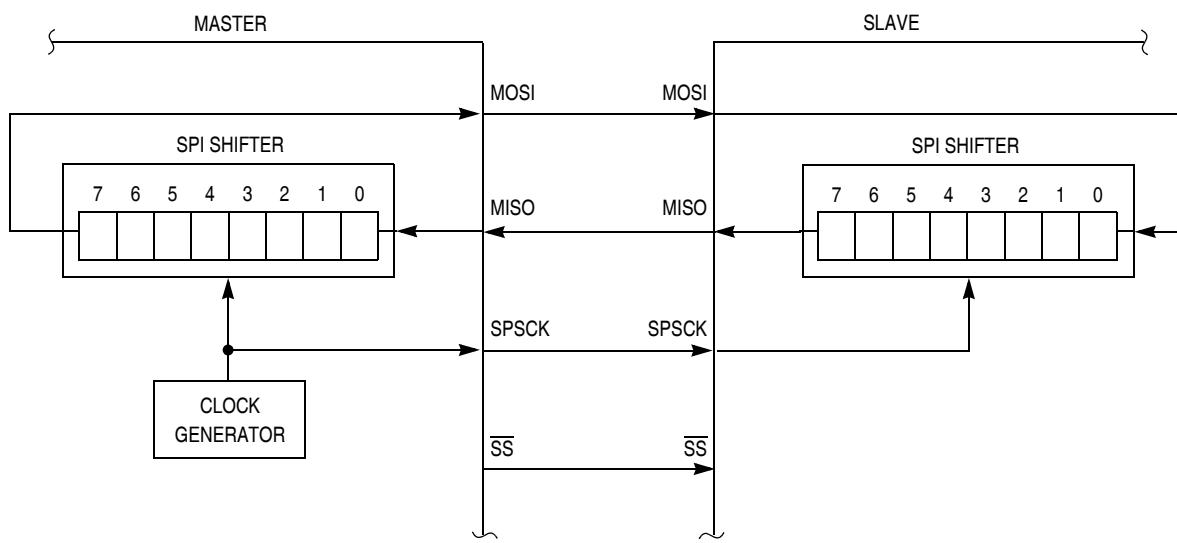


Figure 22-2. SPI System Connections

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although [Figure 22-2](#) shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

#### 22.2.4.2 SPI Module Block Diagram

[Figure 22-3](#) is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPID) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

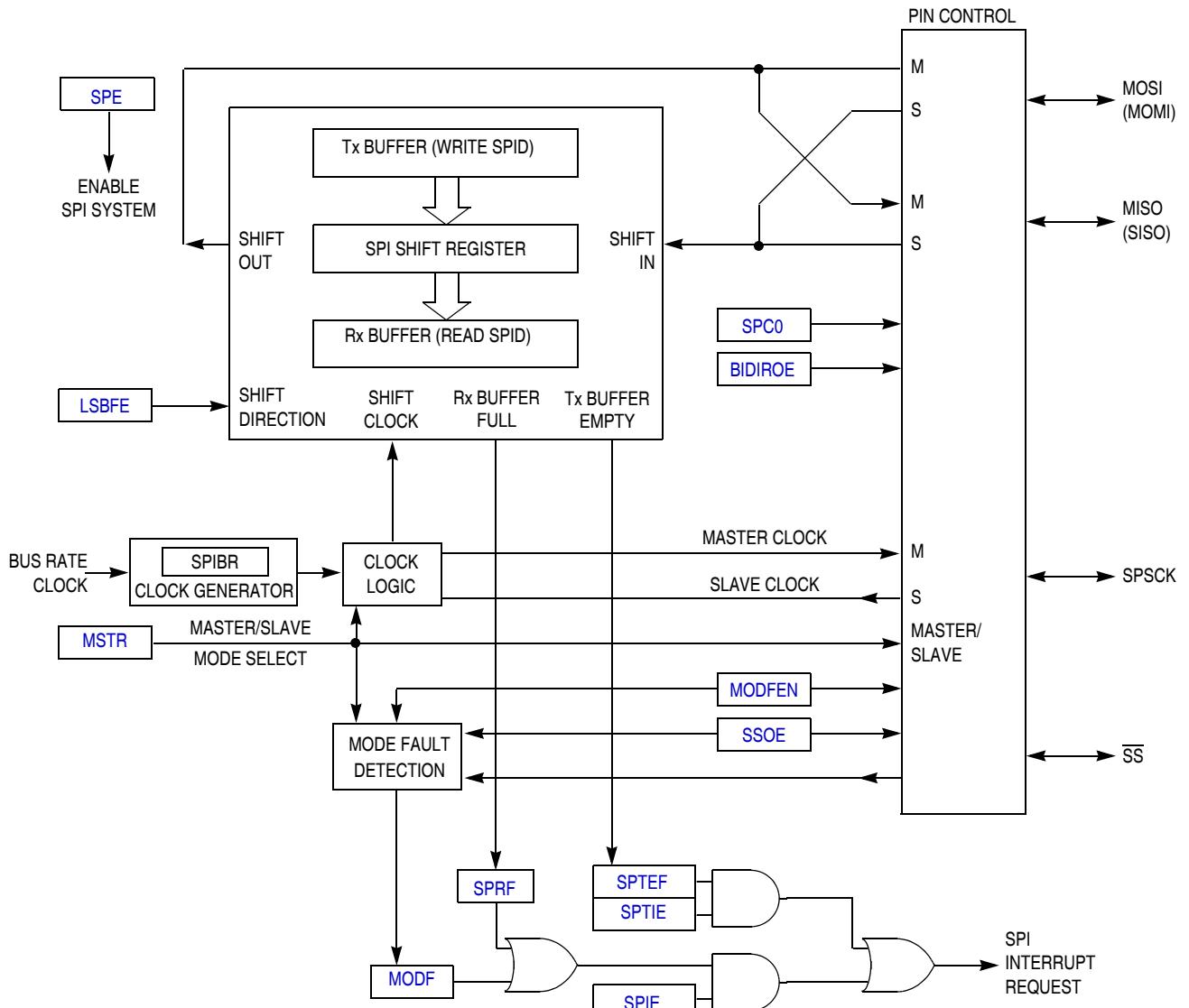


Figure 22-3. SPI Module Block Diagram

## 22.2.5 SPI Baud Rate Generation

As shown in Figure 22-4, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256, or 512 to get the internal SPI master mode bit-rate clock.

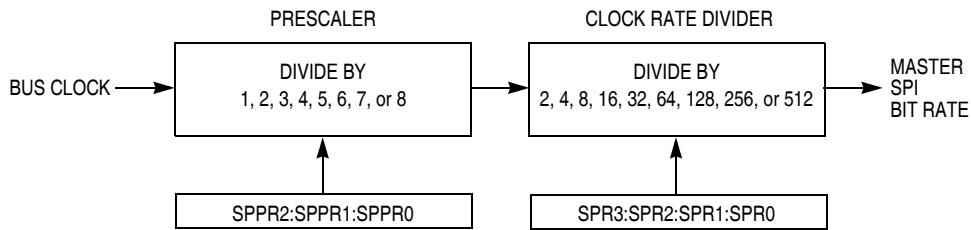


Figure 22-4. SPI Baud Rate Generation

## 22.3 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled ( $SPE = 0$ ), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 22.3.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 22.3.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and  $SPC0 = 0$ , this pin is the serial data input. If  $SPC0 = 1$  to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input ( $BIDIROE = 0$ ) or an output ( $BIDIROE = 1$ ). If  $SPC0 = 1$  and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 22.3.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and  $SPC0 = 0$ , this pin is the serial data output. If  $SPC0 = 1$  to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input ( $BIDIROE = 0$ ) or an output ( $BIDIROE = 1$ ). If  $SPC0 = 1$  and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 22.3.4 SS — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off ( $MODFEN = 0$ ), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and  $MODFEN = 1$ , the slave select output enable bit determines whether this pin acts as the mode fault input ( $SSOE = 0$ ) or as the slave select output ( $SSOE = 1$ ).

## 22.4 Modes of Operation

### 22.4.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

## 22.5 Register Definition

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data.

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 22.5.1 SPI Control Register 1 (SPIC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	7	6	5	4		3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE	
Reset	0	0	0	0	0	1	0	0	0

Figure 22-5. SPI Control Register 1 (SPIC1)

Table 22-1. SPIC1 Field Descriptions

Field	Description
7 SPIE	<b>SPI Interrupt Enable (for SPRF and MODF)</b> — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	<b>SPI System Enable</b> — Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	<b>SPI Transmit Interrupt Enable</b> — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested

**Table 22-1. SPIC1 Field Descriptions (continued)**

Field	Description
4 MSTR	<b>Master/Slave Mode Select</b> 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	<b>Clock Polarity</b> — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to <a href="#">Section 22.6.4, “SPI Clock Formats”</a> for more details. 0 Active-high SPI clock (idle low) 1 Active-low SPI clock (idle high)
2 CPHA	<b>Clock Phase</b> — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to <a href="#">Section 22.6.4, “SPI Clock Formats”</a> for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer 1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer
1 SSOE	<b>Slave Select Output Enable</b> — This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in <a href="#">Table 22-2</a> .
0 LSBFE	<b>LSB First (Shifter Direction)</b> 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

**Table 22-2.  $\overline{SS}$  Pin Function**

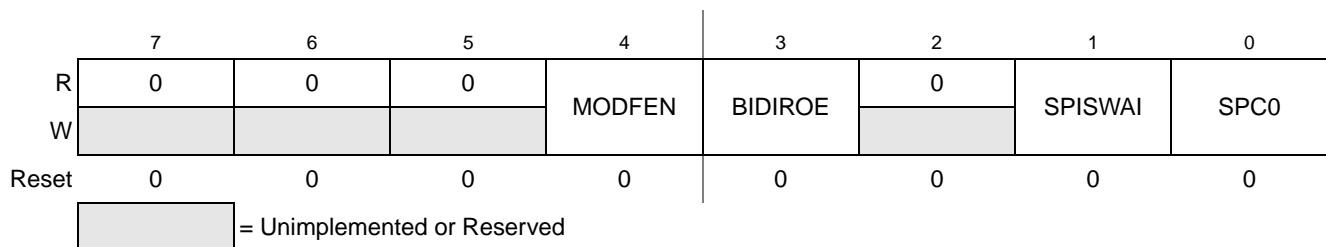
MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

**NOTE**

Ensure that the SPI should not be disabled (SPE=0) at the same time as a bit change to the CPHA bit. These changes should be performed as separate operations or unexpected behavior may occur.

## 22.5.2 SPI Control Register 2 (SPIC2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

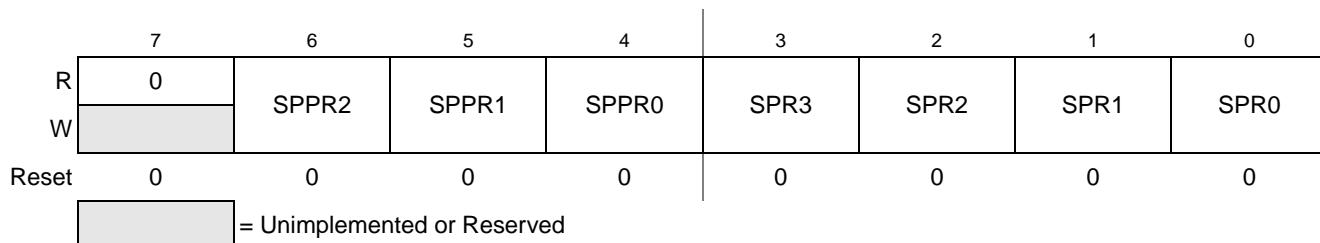
**Figure 22-6. SPI Control Register 2 (SPIC2)**

**Table 22-3. SPIC2 Register Field Descriptions**

Field	Description
4 MODFEN	<b>Master Mode-Fault Function Enable</b> — When the SPI is configured for slave mode, this bit has no meaning or effect. (The SS pin is the slave select input.) In master mode, this bit determines how the SS pin is used (refer to <a href="#">Table 22-2</a> for more details). 0 Mode fault function disabled, master SS pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master SS pin acts as the mode fault input or the slave select output
3 BIDIROE	<b>Bidirectional Mode Output Enable</b> — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	<b>SPI Stop in Wait Mode</b> 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	<b>SPI Pin Control 0</b> — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

### 22.5.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

**Figure 22-7. SPI Baud Rate Register (SPIBR)****Table 22-4. SPIBR Register Field Descriptions**

Field	Description
6:4 SPPR[2:0]	<b>SPI Baud Rate Prescale Divisor</b> — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in <a href="#">Table 22-5</a> . The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see <a href="#">Figure 22-4</a> ).
2:0 SPR[3:0]	<b>SPI Baud Rate Divider</b> — This 4-bit field selects one of nine divisors for the SPI baud rate divider as shown in <a href="#">Table 22-6</a> . The input to this divider comes from the SPI baud rate prescaler (see <a href="#">Figure 22-4</a> ). The output of this divider is the SPI bit rate clock for master mode.

**Table 22-5. SPI Baud Rate Prescaler Divisor**

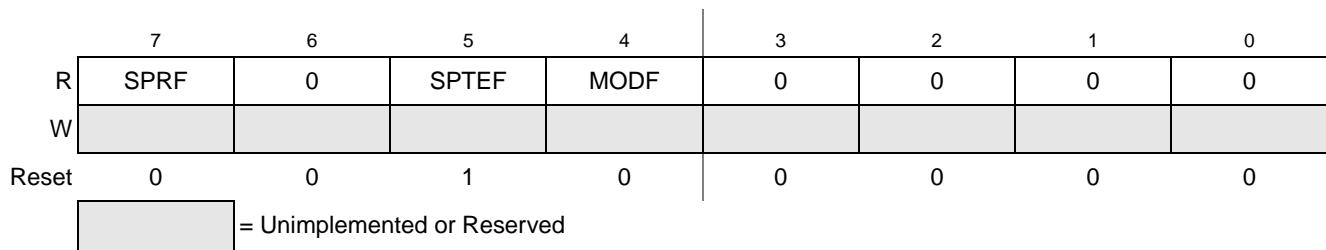
SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

**Table 22-6. SPI Baud Rate Divisor**

SPR3:SPR2:SPR1:SPR0	Rate Divisor
0:0:0:0	2
0:0:0:1	4
0:0:1:0	8
0:0:1:1	16
0:1:0:0	32
0:1:0:1	64
0:1:1:0	128
0:1:1:1	256
1:0:0:0	512
All other combinations	reserved

## 22.5.4 SPI Status Register (SPIS)

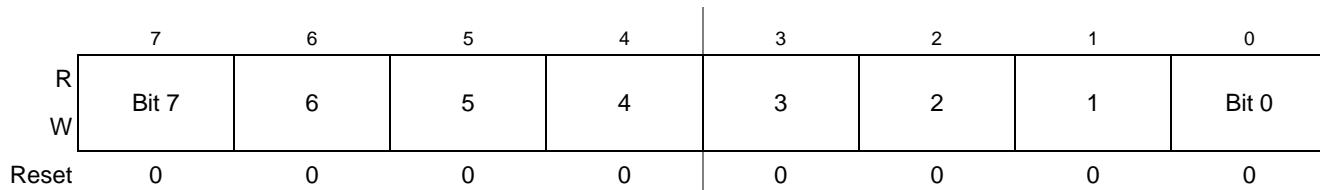
This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0. Writes have no meaning or effect.

**Figure 22-8. SPI Status Register (SPIS)**

**Table 22-7. SPIS Register Field Descriptions**

Field	Description
7 SPRF	<b>SPI Read Buffer Full Flag</b> — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPID). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register. 0 No data available in the receive data buffer 1 Data available in the receive data buffer
5 SPTEF	<b>SPI Transmit Buffer Empty Flag</b> — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIS with SPTEF set, followed by writing a data value to the transmit buffer at SPID. SPIS must be read with SPTEF = 1 before writing data to SPID or the SPID write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPIC1 is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPID is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty
4 MODF	<b>Master Mode Fault Flag</b> — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The SS pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIC1). 0 No mode fault error 1 Mode fault error detected

## 22.5.5 SPI Data Register (SPID)

**Figure 22-9. SPI Data Register (SPID)**

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPID any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

## 22.6 Functional Description

### 22.6.1 General

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI Control Register 1. While the SPE bit is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select (SS)
- Serial clock (SPSCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

An SPI transfer is initiated in the master SPI device by reading the SPI status register (SPIxS) when SPTEF = 1 and then writing data to the transmit data buffer (write to SPIxD). When a transfer is complete, received data is moved into the receive data buffer. The SPIxD register acts as the SPI receive data buffer for reads and as the SPI transmit data buffer for writes.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI Control Register 1 (SPIxC1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SPSCK edges or on even numbered SPSCK edges.

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI control register 1 is set, master mode is selected, when the MSTR bit is clear, slave mode is selected.

### 22.6.2 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by reading the SPIxS register while SPTEF = 1 and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- SPSCK

The SPR3, SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

- MOSI, MISO pin

In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.

- SS pin

If MODFEN and SSOE bit are set, the SS pin is configured as slave select output. The SS output becomes low during each transmission and is high when the SPI is in idle state.

If MODFEN is set and SSOE is cleared, the SS pin is configured as input for detecting mode fault error. If the SS input becomes low this indicates a mode fault error where another master tries to drive the MOSI

and SPSCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI Status Register (SPIxS). If the SPI interrupt enable bit (SPIE) is set when the MODF flag gets set, then an SPI interrupt sequence is also requested.

When a write to the SPI Data Register in the master occurs, there is a half SPSCK-cycle delay. After the delay, SPSCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see Section 22.6.4, “[SPI Clock Formats](#)”).

#### NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPPR2-SPPR0 and SPR3-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

### 22.6.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register1 is clear.

- SPSCK

In slave mode, SPSCK is the SPI clock input from the master.

- MISO, MOSI pin

In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI Control Register 2.

- SS pin

The SS pin is the slave select input. Before a data transmission occurs, the SS pin of the slave SPI must be low. SS must remain low until the transmission is complete. If SS goes high, the SPI is forced into idle state.

The SS input also controls the serial data output pin, if SS is high (not selected), the serial data output pin is high impedance, and, if SS is low the first bit in the SPI Data Register is driven out of the serial data output pin. Also, if the slave is not selected (SS is high), then the SPSCK input is ignored and no internal shifting of the SPI shift register takes place.

Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

**NOTE**

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI Control Register 1 is clear, odd numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the SS input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth shift, the transfer is considered complete and the received data is transferred into the SPI data registers. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

**NOTE**

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0 and BIDIROE with SPC0 set in slave mode will corrupt a transmission in progress and has to be avoided.

## 22.6.4 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

[Figure 22-10](#) shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The SS OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master SS output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The SS IN waveform applies to the slave select input of a slave.

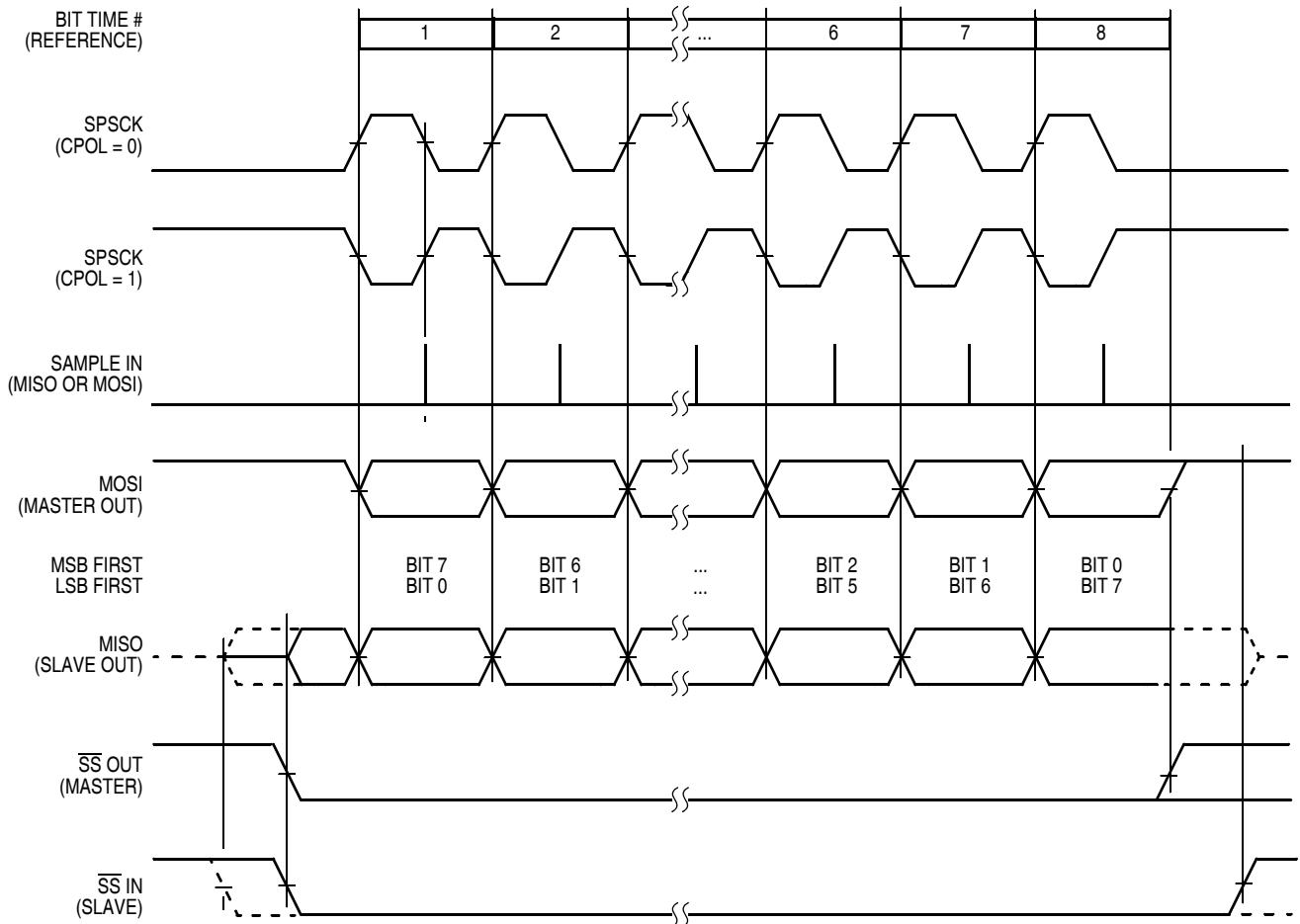
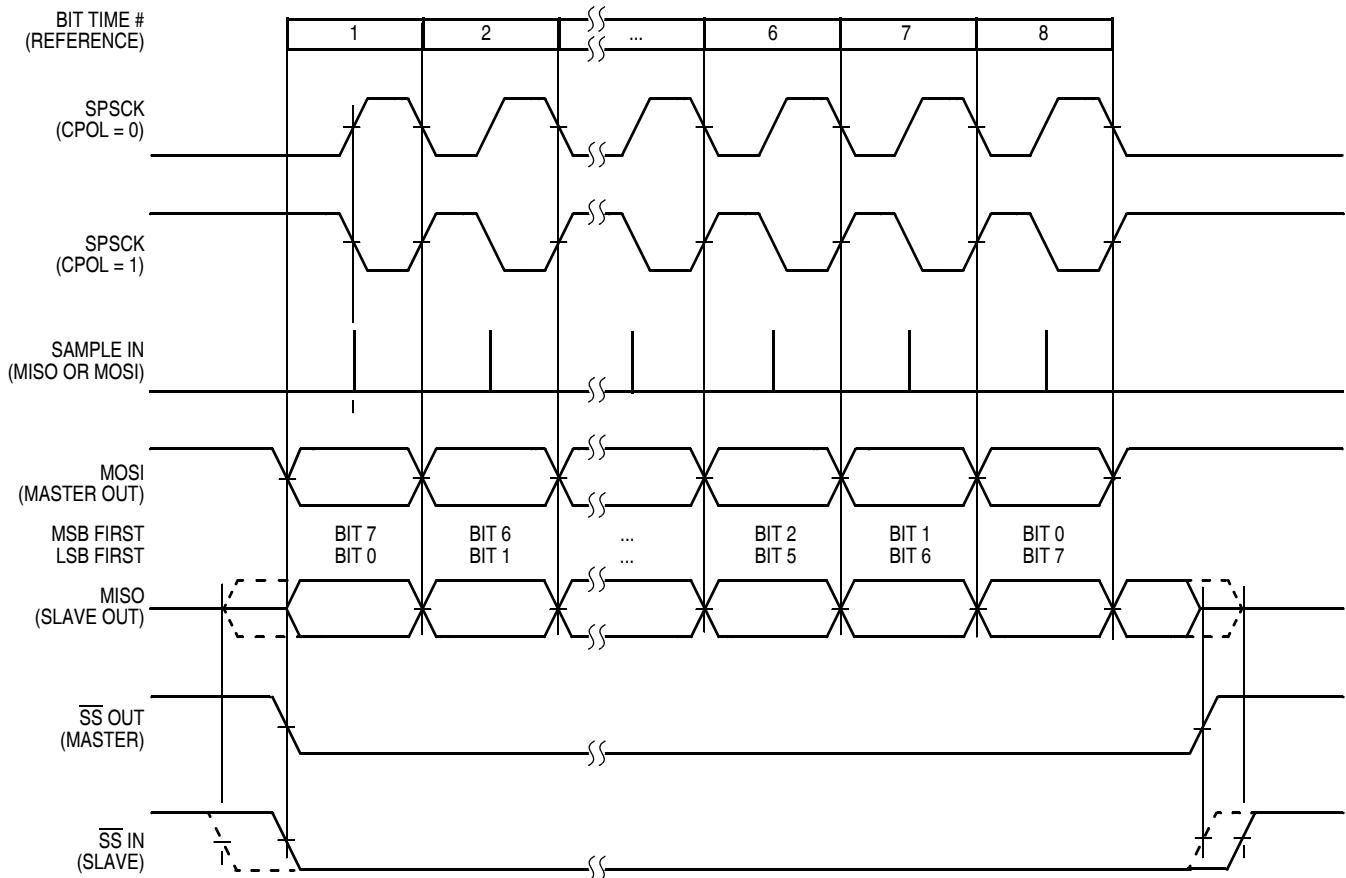


Figure 22-10. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when  $\overline{SS}$  goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's  $\overline{SS}$  input is not required to go to its inactive high level between transfers.

Figure 22-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ( $\overline{SS}$  IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The  $\overline{SS}$  OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master  $\overline{SS}$  output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after

the end of the eighth bit time of the transfer. The  $\overline{SS}$  IN waveform applies to the slave select input of a slave.



**Figure 22-11. SPI Clock Formats (CPHA = 0)**

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when SS goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's  $\overline{SS}$  input must go to its inactive high level between transfers.

## 22.6.5 Special Features

### 22.6.5.1 SS Output

The SS output feature automatically drives the SS pin low during transmission to select external devices and drives it high during idle to deselect external devices. When SS output is selected, the SS output pin is connected to the SS input pin of the external device.

The SS output is available only in master mode during normal SPI operation by asserting the SSOE and MODFEN bits as shown in [Table 22-2., “SS Pin Function](#).

The mode fault feature is disabled while SS output is enabled.

#### NOTE

Care must be taken when using the SS output feature in a multi-master system since the mode fault feature is not available for detecting system errors between masters.

#### 22.6.5.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see Section Table ). In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

**Table 22-8. Normal Mode and Bidirectional Mode**

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
<b>Normal Mode SPC0 = 0</b>	<pre> graph LR     SPI[SPI] -- "Serial Out" --&gt; MOSI[MOSI]     MOSI --&gt; MISO[MISO]     MISO --&gt; "Serial In" --&gt; SPI   </pre>	<pre> graph LR     SPI[SPI] -- "Serial Out" --&gt; MOSI[MOSI]     MOSI --&gt; "Serial In" --&gt; SPI     MISO[MISO] --&gt; "Serial In" --&gt; SPI   </pre>
<b>Bidirectional Mode SPC0 = 1</b>	<pre> graph LR     SPI[SPI] -- "Serial Out" --&gt; BIDIROE[BIDIROE]     BIDIROE --&gt; MOMI[MOMI]     MOMI --&gt; "Serial In" --&gt; SPI     BIDIROE --&gt; "Serial In" --&gt; SPI   </pre>	<pre> graph LR     SPI[SPI] -- "Serial Out" --&gt; BIDIROE[BIDIROE]     BIDIROE --&gt; SOSI[SOSI]     SOSI --&gt; "Serial In" --&gt; SPI     BIDIROE --&gt; "Serial In" --&gt; SPI   </pre>

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

The SPSCK is output for the master mode and input for the slave mode.

The SS is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SPSCK and SS functions.

**NOTE**

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode, in this case MISO becomes occupied by the SPI and MOSI is not used. This has to be considered, if the MISO pin is used for another purpose.

## 22.6.6 SPI Interrupts

There are three flag bits, two interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

## 22.6.7 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the  $\overline{SS}$  pin (provided the  $\overline{SS}$  pin is configured as the mode fault input signal). The  $\overline{SS}$  pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's  $\overline{SS}$  pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPIC1). User software should verify the error condition has been corrected before changing the SPI back to master mode.

# Chapter 23

## Development Support

### 23.1 Introduction

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

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

#### 23.1.1 Forcing Active Background

The method for forcing active background mode depends on the specific HCS08 derivative. For the MC9S08MP16 Series, you can force active background after a power-on reset by holding the BKGD pin low as the device exits the reset condition. You can also force active background by driving BKGD low immediately after a serial background command that writes a one to the BDFR bit in the SBDFR register. Other causes of reset including an external pin reset or an internally generated error reset ignore the state of the BKGD pin and reset into normal user mode. If no debug pod is connected to the BKGD pin, the MCU will always reset into normal operating mode.

#### 23.1.2 Module Configuration

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

### 23.1.3 Features

Features of the BDC module include:

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

## 23.2 Background Debug Controller (BDC)

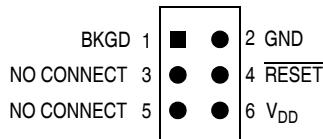
All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources.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

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.
- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, RESET, and sometimes V<sub>DD</sub>. An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V<sub>DD</sub> can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered

separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.



**Figure 23-1. BDM Tool Connector**

### 23.2.1 BKGD Pin Description

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

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to [Section 23.2.2, “Communication Details.”](#)

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

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 23.2.2, “Communication Details,”](#) for more detail.

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

### 23.2.2 Communication Details

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

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if

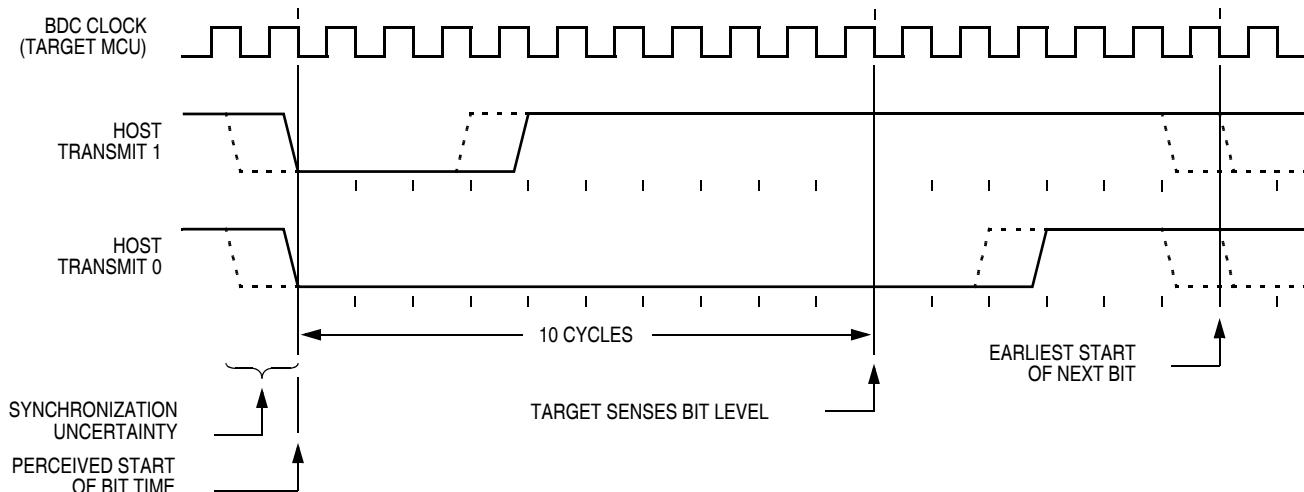
512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

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

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

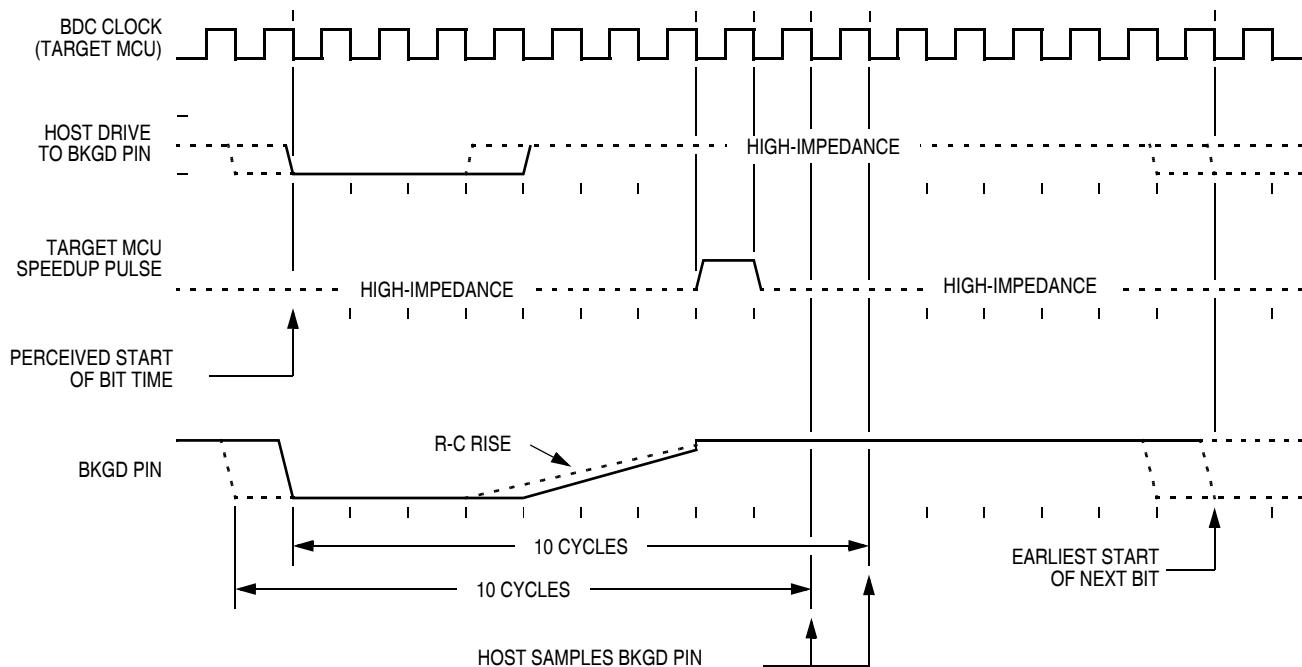
The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting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

**Figure 23-2** shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.



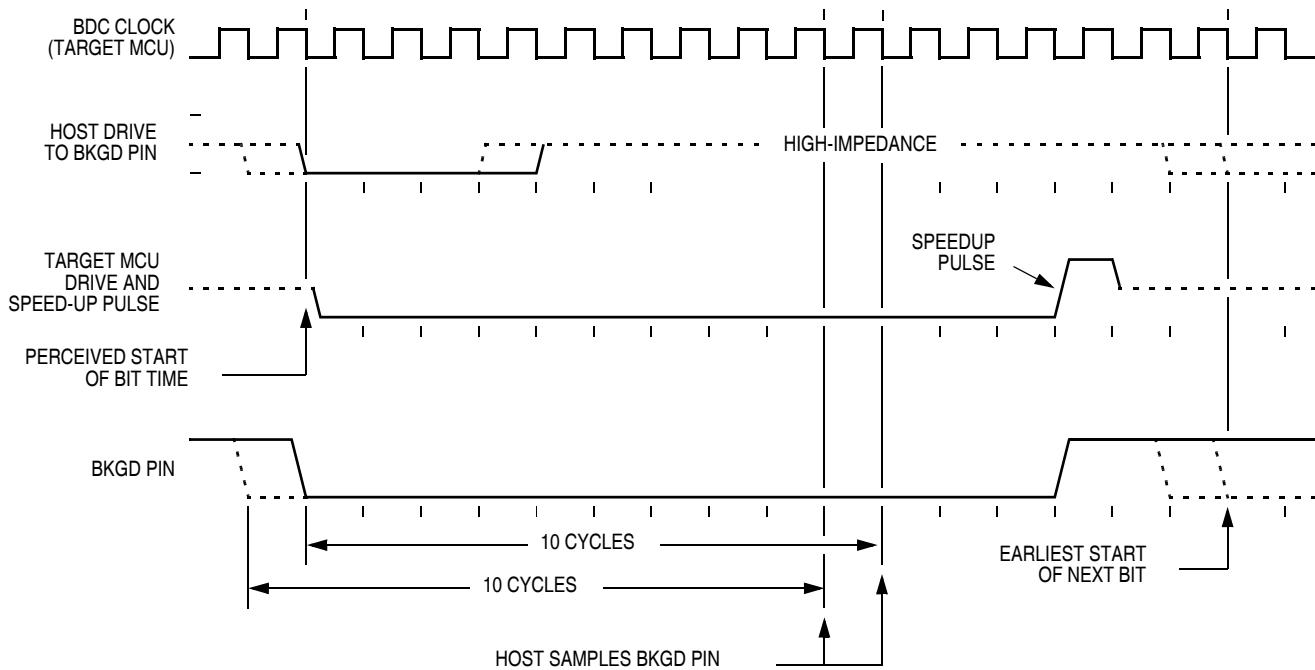
**Figure 23-2. BDC Host-to-Target Serial Bit Timing**

Figure 23-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.



**Figure 23-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 23-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.



**Figure 23-4. BDM Target-to-Host Serial Bit Timing (Logic 0)**

### 23.2.3 BDC Commands

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

[Table 23-1](#) shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

#### Coding Structure Nomenclature

This nomenclature is used in [Table 23-1](#) to describe the coding structure of the BDC commands.

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

**Table 23-1. BDC Command Summary**

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

<sup>1</sup> The SYNC command is a special operation that does not have a command code.

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

To issue a SYNC command, the host:

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

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

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

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

### 23.2.4 BDC Hardware Breakpoint

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

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

## 23.3 Register Definition

This section contains the descriptions of the BDC registers and control bits.

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.

### 23.3.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE\_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

### 23.3.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ\_STATUS and WRITE\_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

  = Unimplemented or Reserved

Figure 23-5. BDC Status and Control Register (BDCSCR)

Table 23-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	<b>Enable BDM (Permit Active Background Mode)</b> — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	<b>Background Mode Active Status</b> — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	<b>BDC Breakpoint Enable</b> — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	<b>Force/Tag Select</b> — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	<b>Select Source for BDC Communications Clock</b> — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

**Table 23-2. BDCSCR Register Field Descriptions (continued)**

<b>Field</b>	<b>Description</b>
2 WS	<b>Wait or Stop Status</b> — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host 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	<b>Wait or Stop Failure Status</b> — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.) 0 Memory access did not conflict with a wait or stop instruction 1 Memory access command failed because the CPU entered wait or stop mode
0 DVF	<b>Data Valid Failure Status</b> — This status bit is not used in the MC9S08MP16 Series because it does not have any slow access memory. 0 Memory access did not conflict with a slow memory access 1 Memory access command failed because CPU was not finished with a slow memory access

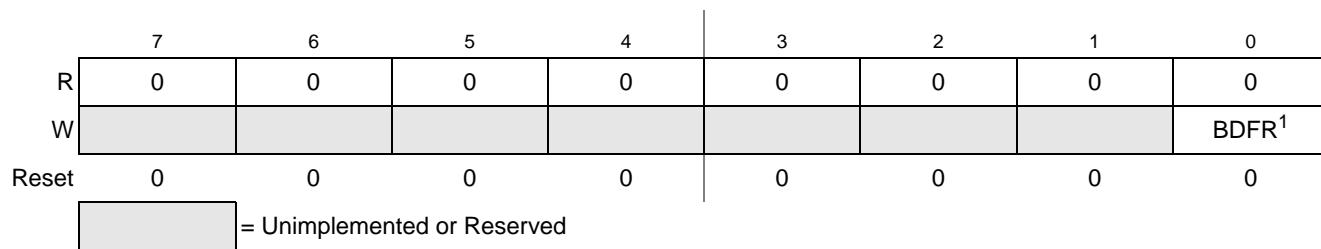
### 23.3.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ\_BKPT and WRITE\_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU.

Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to [Section 23.2.4, “BDC Hardware Breakpoint.”](#)

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

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



<sup>1</sup> BDFR is writable only through serial background mode debug commands, not from user programs.

**Figure 23-6. System Background Debug Force Reset Register (SBDFR)**

**Table 23-3. SBDFR Register Field Description**

Field	Description
0 BDFR	<b>Background Debug Force Reset</b> — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program.



# **Chapter 24**

## **Debug Module (S08DBGV3) (64K)**

### **24.1 Introduction**

The DBG module implements an on-chip ICE (in-circuit emulation) system and allows non-intrusive debug of application software by providing an on-chip trace buffer with flexible triggering capability. The trigger also can provide extended breakpoint capacity. The on-chip ICE system is optimized for the HCS08 8-bit architecture and supports 64K bytes or 128K bytes of memory space.

#### **24.1.1 Features**

The on-chip ICE system includes these distinctive features:

- Three comparators (A, B, and C) with ability to match addresses in 64K space
  - Dual mode, Comparators A and B used to compare addresses
  - Full mode, Comparator A compares address and Comparator B compares data
  - Can be used as triggers and/or breakpoints
  - Comparator C can be used as a normal hardware breakpoint
  - Loop1 capture mode, Comparator C is used to track most recent COF event captured into FIFO
- Tag and Force type breakpoints
- Nine trigger modes
  - A
  - A Or B
  - A Then B
  - A And B, where B is data (Full mode)
  - A And Not B, where B is data (Full mode)
  - Event Only B, store data
  - A Then Event Only B, store data
  - Inside Range,  $A \leq \text{Address} \leq B$
  - Outside Range,  $\text{Address} < A$  or  $\text{Address} > B$
- FIFO for storing change of flow information and event only data
  - Source address of conditional branches taken
  - Destination address of indirect JMP and JSR instruction
  - Destination address of interrupts, RTI, RTC, and RTS instruction
  - Data associated with Event B trigger modes

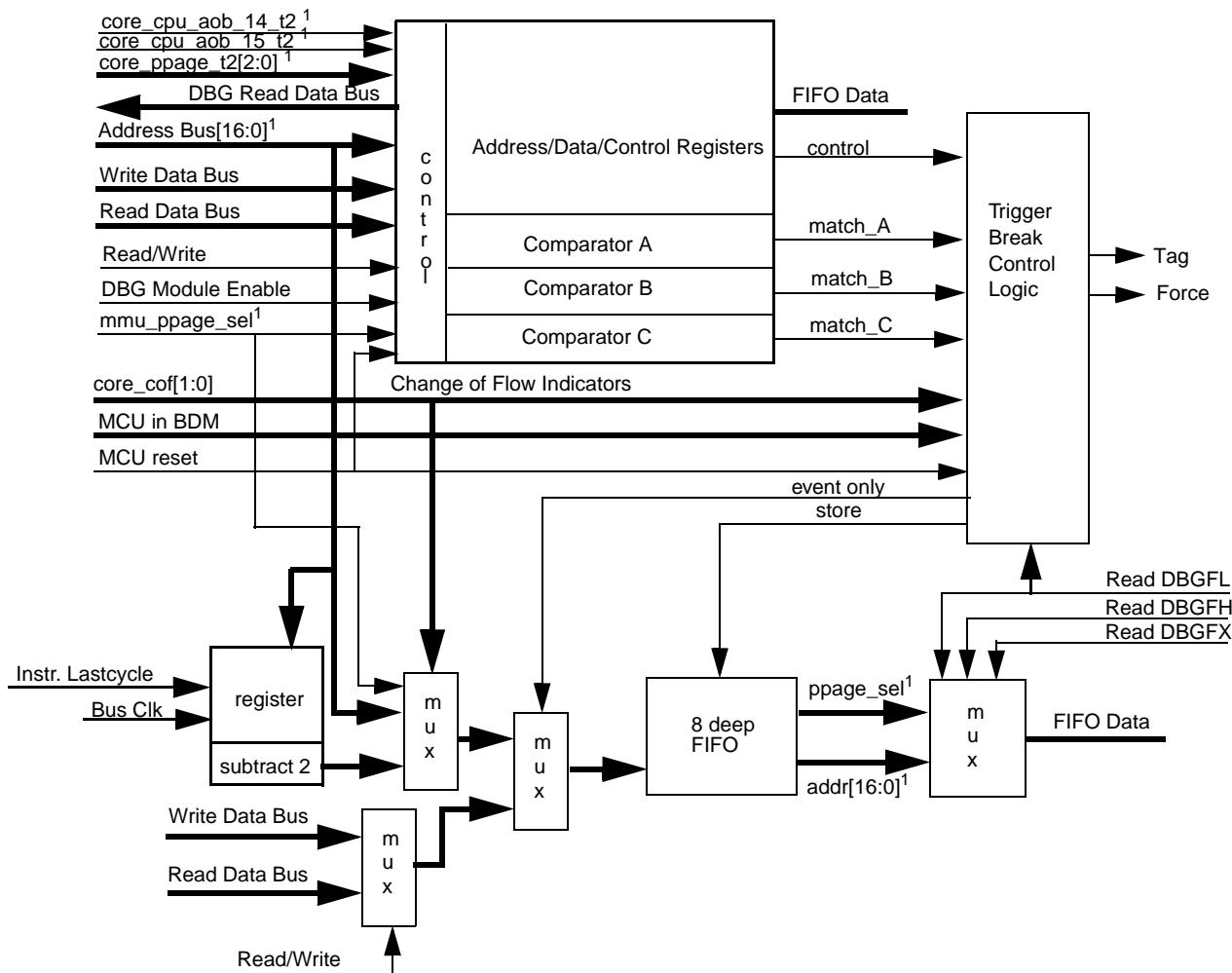
- Ability to End-trace until reset and Begin-trace from reset

### 24.1.2 Modes of Operation

The on-chip ICE system can be enabled in all MCU functional modes. The DBG module is disabled if the MCU is secure. The DBG module comparators are disabled when executing a Background Debug Mode (BDM) command.

### 24.1.3 Block Diagram

Figure 24-1 shows the structure of the DBG module.



1. In 64K versions of this module there are only 16 address lines [15:0], there are no core\_cpu\_aob\_14\_t2, core\_cpu\_aob\_15\_t2, core\_ppage\_t2[2:0], and ppage\_sel signals.

Figure 24-1. DBG Block Diagram

## 24.2 Signal Description

The DBG module contains no external signals.

## 24.3 Memory Map and Registers

This section provides a detailed description of all DBG registers accessible to the end user.

### 24.3.1 Module Memory Map

[Table 24-1](#) shows the registers contained in the DBG module.

**Table 24-1. Module Memory Map**

Address	Use	Access
Base + \$0000	Debug Comparator A High Register (DBGCAH)	Read/write
Base + \$0001	Debug Comparator A Low Register (DBGCAL)	Read/write
Base + \$0002	Debug Comparator B High Register (DBGCBH)	Read/write
Base + \$0003	Debug Comparator B Low Register (DBGCBL)	Read/write
Base + \$0004	Debug Comparator C High Register (DBGCH)	Read/write
Base + \$0005	Debug Comparator C Low Register (BGCC)	Read/write
Base + \$0006	Debug FIFO High Register (DBGFH)	Read only
Base + \$0007	Debug FIFO Low Register (DBGFL)	Read only
Base + \$0008	Debug Comparator A Extension Register (BGCA)	Read/write
Base + \$0009	Debug Comparator B Extension Register (BGCB)	Read/write
Base + \$000A	Debug Comparator C Extension Register (BGCC)	Read/write
Base + \$000B	reserved	Read only
Base + \$000C	Debug Control Register (DBGC)	Read/write
Base + \$000D	Debug Trigger Register (DBGT)	Read/write
Base + \$000E	Debug Status Register (DBGS)	Read only
Base + \$000F	Debug FIFO Count Register (BGCNT)	Read only

**Table 24-2. Register Bit Summary**

	7	6	5	4	3	2	1	0
<b>DBGCAH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>DBGCAL</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>DBGCBH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>DBGCBL</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>DBGCH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>BGCC</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>DBGFH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>DBGFL</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>BGCA</b>	RWAEN	RWA	0	0	0	0	0	0

**Table 24-2. Register Bit Summary**

	7	6	5	4	3	2	1	0
<b>DBGCBX</b>	RWBEN	RWB	0	0	0	0	0	0
<b>DBGCCX</b>	RWCEN	RWC	0	0	0	0	0	0
<b>reserved</b>	0	0	0	0	0	0	0	0
<b>DBGC</b>	DBGEN	ARM	TAG	BRKEN	-	-	-	LOOP1
<b>DBGT</b>	TRGSEL	BEGIN	0	0	TRG[3:0]			
<b>DBGS</b>	AF	BF	CF	0	0	0	0	ARMF
<b>DBGCNT</b>	0	0	0	0	CNT[3:0]			

## 24.3.2 Register Descriptions

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

Note: For all registers below, consider: U = Unchanged, bit maintain its value after reset.

### 24.3.2.1 Debug Comparator A High Register (DBGCAH)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
POR or non-end-run	1	1	1	1	1	1	1	1
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

**Figure 24-2. Debug Comparator A High Register (DBGCAH)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 24-3. DBGCAH Field Descriptions**

Field	Description
Bits 15–8	<b>Comparator A High Compare Bits</b> — The Comparator A High compare bits control whether Comparator A will compare the address bus bits [15:8] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

### 24.3.2.2 Debug Comparator A Low Register (DBGCAL)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POR or non-end-run	1	1	1	1	1	1	1	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

**Figure 24-3. Debug Comparator A Low Register (DBGCAL)**<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.**Table 24-4. DBGCAL Field Descriptions**

Field	Description
Bits 7–0	<b>Comparator A Low Compare Bits</b> — The Comparator A Low compare bits control whether Comparator A will compare the address bus bits [7:0] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

### 24.3.2.3 Debug Comparator B High Register (DBGCBH)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

**Figure 24-4. Debug Comparator B High Register (DBGCBH)**<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.**Table 24-5. DBGCBH Field Descriptions**

Field	Description
Bits 15–8	<b>Comparator B High Compare Bits</b> — The Comparator B High compare bits control whether Comparator B will compare the address bus bits [15:8] to a logic 1 or logic 0. Not used in Full mode. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

### 24.3.2.4 Debug Comparator B Low Register (DBGCBL)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

**Figure 24-5. Debug Comparator B Low Register (DBGCBL)**<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.**Table 24-6. DBGCBL Field Descriptions**

Field	Description
Bits 7–0	<b>Comparator B Low Compare Bits</b> — The Comparator B Low compare bits control whether Comparator B will compare the address bus or data bus bits [7:0] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0, compares to data if in Full mode 1 Compare corresponding address bit to a logic 1, compares to data if in Full mode

### 24.3.2.5 Debug Comparator C High Register (DBGCCH)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

**Figure 24-6. Debug Comparator C High Register (DBGCCH)**<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.**Table 24-7. DBGCCH Field Descriptions**

Field	Description
Bits 15–8	<b>Comparator C High Compare Bits</b> — The Comparator C High compare bits control whether Comparator C will compare the address bus bits [15:8] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

### 24.3.2.6 Debug Comparator C Low Register (DBGCCL)

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

**Figure 24-7. Debug Comparator C Low Register (DBGCCL)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 24-8. DBGCL Field Descriptions**

Field	Description
Bits 7–0	<b>Comparator C Low Compare Bits</b> — The Comparator C Low compare bits control whether Comparator C will compare the address bus bits [7:0] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

### 24.3.2.7 Debug FIFO High Register (DBGFH)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

= Unimplemented or Reserved

**Figure 24-8. Debug FIFO High Register (DBGFH)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 24-9. DBGFH Field Descriptions**

Field	Description
Bits 15–8	<b>FIFO High Data Bits</b> — The FIFO High data bits provide access to bits [15:8] of data in the FIFO. This register is not used in event only modes and will read a \$00 for valid FIFO words.

### 24.3.2.8 Debug FIFO Low Register (DBGFL)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	U	U	U	U	U

= Unimplemented or Reserved

**Figure 24-9. Debug FIFO Low Register (DBGFL)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 24-10. DBGFL Field Descriptions**

Field	Description
Bits 7–0	<b>FIFO Low Data Bits</b> — The FIFO Low data bits contain the least significant byte of data in the FIFO. When reading FIFO words, read DBGFX and DBGFH before reading DBGFL because reading DBGFL causes the FIFO pointers to advance to the next FIFO location. In event-only modes, there is no useful information in DBGFX and DBGFH so it is not necessary to read them before reading DBGFL.

### 24.3.2.9 Debug Comparator A Extension Register (DBGCAx)

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R	RWAEN	RWA	0	0	0	0	0	0
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	0	0	0	0	U

= Unimplemented or Reserved

**Figure 24-10. Debug Comparator A Extension Register (DBGCAx)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 24-11. DBGCAx Field Descriptions**

Field	Description
7 RWAEN	<b>Read/Write Comparator A Enable Bit</b> — The RWAEN bit controls whether read or write comparison is enabled for Comparator A. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
6 RWA	<b>Read/Write Comparator A Value Bit</b> — The RWA bit controls whether read or write is used in compare for Comparator A. The RWA bit is not used if RWAEN = 0. 0 Write cycle will be matched 1 Read cycle will be matched

### 24.3.2.10 Debug Comparator B Extension Register (DBGCBx)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	RWBEN	RWB	0	0	0	0	0	0
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	0	0	0	0	U

= Unimplemented or Reserved

**Figure 24-11. Debug Comparator B Extension Register (DBGCBx)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-12. DBGCBX Field Descriptions

Field	Description
7 RWBEN	<b>Read/Write Comparator B Enable Bit</b> — The RWBEN bit controls whether read or write comparison is enabled for Comparator B. In full modes, RWAEN and RWA are used to control comparison of R/W and RWBEN is ignored. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
6 RWB	<b>Read/Write Comparator B Value Bit</b> — The RWB bit controls whether read or write is used in compare for Comparator B. The RWB bit is not used if RWBEN = 0. In full modes, RWAEN and RWA are used to control comparison of R/W and RWB is ignored. 0 Write cycle will be matched 1 Read cycle will be matched

### 24.3.2.11 Debug Comparator C Extension Register (DBGCCX)

Module Base + 0x000A

	7	6	5	4	3	2	1	0
R	RWCEN	RWC	0	0	0	0	0	0
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	U	0	0	0	0	U
= Unimplemented or Reserved								

Figure 24-12. Debug Comparator C Extension Register (DBGCCX)

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-13. DBGCCX Field Descriptions

Field	Description
7 RWCEN	<b>Read/Write Comparator C Enable Bit</b> — The RWCEN bit controls whether read or write comparison is enabled for Comparator C. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
6 RWC	<b>Read/Write Comparator C Value Bit</b> — The RWC bit controls whether read or write is used in compare for Comparator C. The RWC bit is not used if RWCEN = 0. 0 Write cycle will be matched 1 Read cycle will be matched

### 24.3.2.12 Debug Control Register (DBGC)

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R W	DBGEN	ARM	TAG	BRKEN	0	0	0	LOOP1
POR or non-end-run	1	1	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	0	U	0	0	0	0	U
= Unimplemented or Reserved								

**Figure 24-13. Debug Control Register (DBGC)**

- <sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the ARM and BRKEN bits are cleared but the remaining control bits in this register do not change after reset.

**Table 24-14. DBGC Field Descriptions**

Field	Description
7 DBGEN	<b>DBG Module Enable Bit</b> — The DBGEN bit enables the DBG module. The DBGEN bit is forced to zero and cannot be set if the MCU is secure. 0 DBG not enabled 1 DBG enabled
6 ARM	<b>Arm Bit</b> — The ARM bit controls whether the debugger is comparing and storing data in FIFO. See <a href="#">Section 24.4.4.2, “Arming the DBG Module”</a> for more information. 0 Debugger not armed 1 Debugger armed
5 TAG	<b>Tag or Force Bit</b> — The TAG bit controls whether a debugger or comparator C breakpoint will be requested as a tag or force breakpoint to the CPU. The TAG bit is not used if BRKEN = 0. 0 Force request selected 1 Tag request selected
4 BRKEN	<b>Break Enable Bit</b> — The BRKEN bit controls whether the debugger will request a breakpoint to the CPU at the end of a trace run, and whether comparator C will request a breakpoint to the CPU. 0 CPU break request not enabled 1 CPU break request enabled
0 LOOP1	<b>Select LOOP1 Capture Mode</b> — This bit selects either normal capture mode or LOOP1 capture mode. LOOP1 is not used in event-only modes. 0 Normal operation - capture COF events into the capture buffer FIFO 1 LOOP1 capture mode enabled. When the conditions are met to store a COF value into the FIFO, compare the current COF address with the address in comparator C. If these addresses match, override the FIFO capture and do not increment the FIFO count. If the address does not match comparator C, capture the COF address, including the PPACC indicator, into the FIFO and into comparator C.

### 24.3.2.13 Debug Trigger Register (DBGT)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R W <sup>2</sup>	TRGSEL	BEGIN	0	0			TRG	
POR or non-end-run	0	1	0	0	0	0	0	0
Reset end-run <sup>1</sup>	U	U	0	0	U	U	U	U

= Unimplemented or Reserved

**Figure 24-14. Debug Trigger Register (DBGT)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the control bits in this register do not change after reset.

<sup>2</sup> The DBG trigger register (DBGT) can not be changed unless ARM=0.

**Table 24-15. DBGT Field Descriptions**

Field	Description
7 TRGSEL	<b>Trigger Selection Bit</b> — The TRGSEL bit controls the triggering condition for the comparators. See <a href="#">Section 24.4.4, “Trigger Break Control (TBC)”</a> for more information. 0 Trigger on any compare address access 1 Trigger if opcode at compare address is executed
6 BEGIN	<b>Begin/End Trigger Bit</b> — The BEGIN bit controls whether the trigger begins or ends storing of data in FIFO. 0 Trigger at end of stored data 1 Trigger before storing data
3–0 TRG	<b>Trigger Mode Bits</b> — The TRG bits select the trigger mode of the DBG module as shown in <a href="#">Table 24-16</a> .

**Table 24-16. Trigger Mode Encoding**

TRG Value	Meaning
0000	A Only
0001	A Or B
0010	A Then B
0011	Event Only B
0100	A Then Event Only B
0101	A And B (Full Mode)
0110	A And Not B (Full mode)
0111	Inside Range
1000	Outside Range

**Table 24-16. Trigger Mode Encoding**

TRG Value	Meaning
1001 ↓ 1111	No Trigger

**NOTE**

The DBG trigger register (DBGT) can not be changed unless ARM=0.

**24.3.2.14 Debug Status Register (DBGS)**

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	AF	BF	CF	0	0	0	0	ARMF
W								
POR or non-end-run	0	0	0	0	0	0	0	1
Reset end-run <sup>1</sup>	U	U	U	0	0	0	0	0

= Unimplemented or Reserved

**Figure 24-15. Debug Status Register (DBGS)**

- <sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, ARMF gets cleared by reset but AF, BF, and CF do not change after reset.

**Table 24-17. DBGS Field Descriptions**

Field	Description
7 AF	<b>Trigger A Match Bit</b> — The AF bit indicates if Trigger A match condition was met since arming. 0 Comparator A did not match 1 Comparator A match
6 BF	<b>Trigger B Match Bit</b> — The BF bit indicates if Trigger B match condition was met since arming. 0 Comparator B did not match 1 Comparator B match
5 CF	<b>Trigger C Match Bit</b> — The CF bit indicates if Trigger C match condition was met since arming. 0 Comparator C did not match 1 Comparator C match
0 ARMF	<b>Arm Flag Bit</b> — The ARMF bit indicates whether the debugger is waiting for trigger or waiting for the FIFO to fill. While DBGEN = 1, this status bit is a read-only image of the ARM bit in DBGC. See <a href="#">Section 24.4.4.2, “Arming the DBG Module”</a> for more information. 0 Debugger not armed 1 Debugger armed

### 24.3.2.15 Debug Count Status Register (DBGCNT)

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	0	0	0	0	CNT			
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run <sup>1</sup>	0	0	0	0	U	U	U	U

= Unimplemented or Reserved

**Figure 24-16. Debug Count Status Register (DBGCNT)**

<sup>1</sup> In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the CNT[3:0] bits do not change after reset.

**Table 24-18. DBGS Field Descriptions**

Field	Description
3–0 CNT	<b>FIFO Valid Count Bits</b> — The CNT bits indicate the amount of valid data stored in the FIFO. <a href="#">Table 24-19</a> shows the correlation between the CNT bits and the amount of valid data in FIFO. The CNT will stop after a count to eight even if more data is being stored in the FIFO. The CNT bits are cleared when the DBG module is armed, and the count is incremented each time a new word is captured into the FIFO. The host development system is responsible for checking the value in CNT[3:0] and reading the correct number of words from the FIFO because the count does not decrement as data is read out of the FIFO at the end of a trace run.

**Table 24-19. CNT Bits**

CNT Value	Meaning
0000	No data valid
0001	1 word valid
0010	2 words valid
0011	3 words valid
0100	4 words valid
0101	5 words valid
0110	6 words valid
0111	7 words valid
1000	8 words valid

## 24.4 Functional Description

This section provides a complete functional description of the on-chip ICE system. The DBG module is enabled by setting the DBGEN bit in the DBGC register. Enabling the module allows the arming, triggering and storing of data in the FIFO. The DBG module is made up of three main blocks, the Comparators, Trigger Break Control logic and the FIFO.

### 24.4.1 Comparator

The DBG module contains three comparators, A, B, and C. Comparator A compares the core address bus with the address stored in the DBGCAH and DBGCAL registers. Comparator B compares the core address bus with the address stored in the DBGCBH and DBGCBL registers except in full mode, where it compares the data buses to the data stored in the DBGCBL register. Comparator C compares the core address bus with the address stored in the DBGCCH and DBGCCL registers. Matches on Comparators A, B, and C are signaled to the Trigger Break Control (TBC) block.

#### 24.4.1.1 RWA and RWAEN in Full Modes

In full modes ("A And B" and "A And Not B") RWAEN and RWA are used to select read or write comparisons for both comparators A and B. To select write comparisons and the write data bus in Full Modes set RWAEN=1 and RWA=0, otherwise read comparisons and the read data bus will be selected. The RWBEN and RWB bits are not used and will be ignored in Full Modes.

#### 24.4.1.2 Comparator C in LOOP1 Capture Mode

Normally comparator C is used as a third hardware breakpoint and is not involved in the trigger logic for the on-chip ICE system. In this mode, it compares the core address bus with the address stored in the DBGCCH and DBGCCL registers. However, in LOOP1 capture mode, comparator C is managed by logic in the DBG module to track the address of the most recent change-of-flow event that was captured into the FIFO buffer. In LOOP1 capture mode, comparator C is not available for use as a normal hardware breakpoint.

When the ARM and DBGEN bits are set to one in LOOP1 capture mode, comparator C value registers are cleared to prevent the previous contents of these registers from interfering with the LOOP1 capture mode operation. When a COF event is detected, the address of the event is compared to the contents of the DBGCCH and DBGCCL registers to determine whether it is the same as the previous COF entry in the capture FIFO. If the values match, the capture is inhibited to prevent the FIFO from filling up with duplicate entries. If the values do not match, the COF event is captured into the FIFO and the DBGCCH and DBGCCL registers are updated to reflect the address of the captured COF event.

### 24.4.2 Breakpoints

A breakpoint request to the CPU at the end of a trace run can be created if the BRKEN bit in the DBGC register is set. The value of the BEGIN bit in DBGT register determines when the breakpoint request to the CPU will occur. If the BEGIN bit is set, begin-trigger is selected and the breakpoint request will not occur until the FIFO is filled with 8 words. If the BEGIN bit is cleared, end-trigger is selected and the breakpoint request will occur immediately at the trigger cycle.

When traditional hardware breakpoints from comparators A or B are desired, set BEGIN=0 to select an end-trace run and set the trigger mode to either 0x0 (A-only) or 0x1 (A OR B) mode.

There are two types of breakpoint requests supported by the DBG module, tag-type and force-type. Tagged breakpoints are associated with opcode addresses and allow breaking just before a specific instruction executes. Force breakpoints are not associated with opcode addresses and allow breaking at the next instruction boundary. The TAG bit in the DBGC register determines whether CPU breakpoint requests will be a tag-type or force-type breakpoints. When TAG=0, a force-type breakpoint is requested and it will take effect at the next instruction boundary after the request. When TAG=1, a tag-type breakpoint is registered into the instruction queue and the CPU will break if/when this tag reaches the head of the instruction queue and the tagged instruction is about to be executed.

#### 24.4.2.1 Hardware Breakpoints

Comparators A, B, and C can be used as three traditional hardware breakpoints whether the on-chip ICE real-time capture function is required or not. To use any breakpoint or trace run capture functions set DBGEN=1. BRKEN and TAG affect all three comparators. When BRKEN=0, no CPU breakpoints are enabled. When BRKEN=1, CPU breakpoints are enabled and the TAG bit determines whether the breakpoints will be tag-type or force-type breakpoints. To use comparators A and B as hardware breakpoints, set DBGT=0x81 for tag-type breakpoints and 0x01 for force-type breakpoints. This sets up an end-type trace with trigger mode “A OR B”.

Comparator C is not involved in the trigger logic for the on-chip ICE system.

#### 24.4.3 Trigger Selection

The TRGSEL bit in the DBGT register is used to determine the triggering condition of the on-chip ICE system. TRGSEL applies to both trigger A and B except in the event only trigger modes. By setting the TRGSEL bit, the comparators will qualify a match with the output of opcode tracking logic. The opcode tracking logic is internal to each comparator and determines whether the CPU executed the opcode at the compare address. With the TRGSEL bit cleared a comparator match is all that is necessary for a trigger condition to be met.

##### NOTE

If the TRGSEL is set, the address stored in the comparator match address registers must be an opcode address for the trigger to occur.

#### 24.4.4 Trigger Break Control (TBC)

The TBC is the main controller for the DBG module. Its function is to decide whether data should be stored in the FIFO based on the trigger mode and the match signals from the comparator. The TBC also determines whether a request to break the CPU should occur.

The TAG bit in DBGC controls whether CPU breakpoints are treated as tag-type or force-type breakpoints. The TRGSEL bit in DBGT controls whether a comparator A or B match is further qualified by opcode tracking logic. Each comparator has a separate circuit to track opcodes because the comparators could

correspond to separate instructions that could be propagating through the instruction queue at the same time.

In end-type trace runs ( $BEGIN=0$ ), when the comparator registers match, including the optional R/W match, this signal goes to the CPU break logic where BRKEN determines whether a CPU break is requested and the TAG control bit determines whether the CPU break will be a tag-type or force-type breakpoint. When TRGSEL is set, the R/W qualified comparator match signal also passes through the opcode tracking logic. If/when it propagates through this logic, it will cause a trigger to the ICE logic to begin or end capturing information into the FIFO. In the case of an end-type ( $BEGIN=0$ ) trace run, the qualified comparator signal stops the FIFO from capturing any more information.

If a CPU breakpoint is also enabled, you would want TAG and TRGSEL to agree so that the CPU break occurs at the same place in the application program as the FIFO stopped capturing information. If TRGSEL was 0 and TAG was 1 in an end-type trace run, the FIFO would stop capturing as soon as the comparator address matched, but the CPU would continue running until a TAG signal could propagate through the CPUs instruction queue which could take a long time in the case where changes of flow caused the instruction queue to be flushed. If TRGSEL was one and TAG was zero in an end-type trace run, the CPU would break before the comparator match signal could propagate through the opcode tracking logic to end the trace run.

In begin-type trace runs ( $BEGIN=1$ ), the start of FIFO capturing is triggered by the qualified comparator signals, and the CPU breakpoint (if enabled by BRKEN=1) is triggered when the FIFO becomes full. Since this FIFO full condition does not correspond to the execution of a tagged instruction, it would not make sense to use TAG=1 for a begin-type trace run.

#### **24.4.4.1 Begin- and End-Trigger**

The definition of begin- and end-trigger as used in the DBG module are as follows:

- Begin-trigger: Storage in FIFO occurs after the trigger and continues until 8 locations are filled.
- End-trigger: Storage in FIFO occurs until the trigger with the least recent data falling out of the FIFO if more than 8 words are collected.

#### **24.4.4.2 Arming the DBG Module**

Arming occurs by enabling the DBG module by setting the DBGEN bit and by setting the ARM bit in the DBGC register. The ARM bit in the DBGC register and the ARMF bit in the DBGS register are cleared when the trigger condition is met in end-trigger mode or when the FIFO is filled in begin-trigger mode. In the case of an end-trace where DBGEN=1 and BEGIN=0, ARM and ARMF are cleared by any reset to end the trace run that was in progress. The ARMF bit is also cleared if ARM is written to zero or when the DBGEN bit is low. The TBC logic determines whether a trigger condition has been met based on the trigger mode and the trigger selection.

#### **24.4.4.3 Trigger Modes**

The on-chip ICE system supports nine trigger modes. The trigger modes are encoded as shown in [Table 24-16](#). The trigger mode is used as a qualifier for either starting or ending the storing of data in the FIFO. When the match condition is met, the appropriate flag AF or BF is set in DBGS register. Arming

the DBG module clears the AF, BF, and CF flags in the DBGS register. In all trigger modes except for the event only modes change of flow addresses are stored in the FIFO. In the event only modes only the value on the data bus at the trigger event B comparator match address will be stored.

#### **24.4.4.3.1 A Only**

In the A Only trigger mode, if the match condition for A is met, the AF flag in the DBGS register is set.

#### **24.4.4.3.2 A Or B**

In the A Or B trigger mode, if the match condition for A or B is met, the corresponding flag(s) in the DBGS register are set.

#### **24.4.4.3.3 A Then B**

In the A Then B trigger mode, the match condition for A must be met before the match condition for B is compared. When the match condition for A or B is met, the corresponding flag in the DBGS register is set.

#### **24.4.4.3.4 Event Only B**

In the Event Only B trigger mode, if the match condition for B is met, the BF flag in the DBGS register is set. The Event Only B trigger mode is considered a begin-trigger type and the BEGIN bit in the DBGT register is ignored.

#### **24.4.4.3.5 A Then Event Only B**

In the A Then Event Only B trigger mode, the match condition for A must be met before the match condition for B is compared. When the match condition for A or B is met, the corresponding flag in the DBGS register is set. The A Then Event Only B trigger mode is considered a begin-trigger type and the BEGIN bit in the DBGT register is ignored.

#### **24.4.4.3.6 A And B (Full Mode)**

In the A And B trigger mode, Comparator A compares to the address bus and Comparator B compares to the data bus. In the A and B trigger mode, if the match condition for A and B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only B happens, no flags are set.

For Breakpoint tagging operation with an end-trigger type trace, only matches from Comparator A will be used to determine if the Breakpoint conditions are met and Comparator B matches will be ignored.

#### **24.4.4.3.7 A And Not B (Full Mode)**

In the A And Not B trigger mode, comparator A compares to the address bus and comparator B compares to the data bus. In the A And Not B trigger mode, if the match condition for A and Not B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only Not B occur no flags are set.

For Breakpoint tagging operation with an end-trigger type trace, only matches from Comparator A will be used to determine if the Breakpoint conditions are met and Comparator B matches will be ignored.

#### 24.4.4.3.8 Inside Range, A ≤ address ≤ B

In the Inside Range trigger mode, if the match condition for A and B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only B occur no flags are set.

#### 24.4.4.3.9 Outside Range, address < A or address > B

In the Outside Range trigger mode, if the match condition for A or B is met, the corresponding flag in the DBGS register is set.

The four control bits BEGIN and TRGSEL in DBGT, and BRKEN and TAG in DBGC, determine the basic type of debug run as shown in Table 1.21. Some of the 16 possible combinations are not used (refer to the notes at the end of the table).

**Table 24-20. Basic Types of Debug Runs**

BEGIN	TRGSEL	BRKEN	TAG	Type of Debug Run
0	0	0	x <sup>(1)</sup>	Fill FIFO until trigger address (No CPU breakpoint - keep running)
0	0	1	0	Fill FIFO until trigger address, then force CPU breakpoint
0	0	1	1	Do not use <sup>(2)</sup>
0	1	0	x <sup>(1)</sup>	Fill FIFO until trigger opcode about to execute (No CPU breakpoint - keep running)
0	1	1	0	Do not use <sup>(3)</sup>
0	1	1	1	Fill FIFO until trigger opcode about to execute (trigger causes CPU breakpoint)
1	0	0	x <sup>(1)</sup>	Start FIFO at trigger address (No CPU breakpoint - keep running)
1	0	1	0	Start FIFO at trigger address, force CPU breakpoint when FIFO full
1	0	1	1	Do not use <sup>(4)</sup>
1	1	0	x <sup>(1)</sup>	Start FIFO at trigger opcode (No CPU breakpoint - keep running)
1	1	1	0	Start FIFO at trigger opcode, force CPU breakpoint when FIFO full
1	1	1	1	Do not use <sup>(4)</sup>

<sup>1</sup> When BRKEN = 0, TAG is do not care (x in the table).

<sup>2</sup> In end trace configurations (BEGIN = 0) where a CPU breakpoint is enabled (BRKEN = 1), TRGSEL should agree with TAG. In this case, where TRGSEL = 0 to select no opcode tracking qualification and TAG = 1 to specify a tag-type CPU breakpoint, the CPU breakpoint would not take effect until sometime after the FIFO stopped storing values. Depending on program loops or interrupts, the delay could be very long.

<sup>3</sup> In end trace configurations (BEGIN = 0) where a CPU breakpoint is enabled (BRKEN = 1), TRGSEL should agree with TAG. In this case, where TRGSEL = 1 to select opcode tracking qualification and TAG = 0 to specify a force-type CPU breakpoint, the CPU breakpoint would erroneously take effect before the FIFO stopped storing values and the debug run would not complete normally.

<sup>4</sup> In begin trace configurations (BEGIN = 1) where a CPU breakpoint is enabled (BRKEN = 1), TAG should not be set to 1. In begin trace debug runs, the CPU breakpoint corresponds to the FIFO full condition which does not correspond to a taggable instruction fetch.

## 24.4.5 FIFO

The FIFO is an eight word deep FIFO. In all trigger modes except for event only, the data stored in the FIFO will be change of flow addresses. In the event only trigger modes only the data bus value corresponding to the event is stored. In event only trigger modes, the high byte of the valid data from the FIFO will always read a 0x00.

### 24.4.5.1 Storing Data in FIFO

In all trigger modes except for the event only modes, the address stored in the FIFO will be determined by the change of flow indicators from the core. The signal core\_cof[1] indicates the current core address is the destination address of an indirect JSR or JMP instruction, or a RTS or RTI instruction or interrupt vector and the destination address should be stored. The signal core\_cof[0] indicates that a conditional branch was taken and that the source address of the conditional branch should be stored.

### 24.4.5.2 Storing with Begin-Trigger

Storing with Begin-Trigger can be used in all trigger modes. Once the DBG module is enabled and armed in the begin-trigger mode, data is not stored in the FIFO until the trigger condition is met. Once the trigger condition is met the DBG module will remain armed until 8 words are stored in the FIFO. If the core\_cof[1] signal becomes asserted, the current address is stored in the FIFO. If the core\_cof[0] signal becomes asserted, the address registered during the previous last cycle is decremented by two and stored in the FIFO.

### 24.4.5.3 Storing with End-Trigger

Storing with End-Trigger cannot be used in event-only trigger modes. Once the DBG module is enabled and armed in the end-trigger mode, data is stored in the FIFO until the trigger condition is met. If the core\_cof[1] signal becomes asserted, the current address is stored in the FIFO. If the core\_cof[0] signal becomes asserted, the address registered during the previous last cycle is decremented by two and stored in the FIFO. When the trigger condition is met, the ARM and ARMF will be cleared and no more data will be stored. In non-event only end-trigger modes, if the trigger is at a change of flow address the trigger event will be stored in the FIFO.

### 24.4.5.4 Reading Data from FIFO

The data stored in the FIFO can be read using BDM commands provided the DBG module is enabled and not armed (DBGEN=1 and ARM=0). The FIFO data is read out first-in-first-out. By reading the CNT bits in the DBGCNT register at the end of a trace run, the number of valid words can be determined. The FIFO data is read by optionally reading the DBGFH register followed by the DBGFL register. Each time the DBGFL register is read the FIFO is shifted to allow reading of the next word however the count does not decrement. In event-only trigger modes where the FIFO will contain only the data bus values stored, to read the FIFO only DBGFL needs to be accessed.

The FIFO is normally only read while ARM and ARMF=0, however reading the FIFO while the DBG module is armed will return the data value in the oldest location of the FIFO and the TBC will not allow

the FIFO to shift. This action could cause a valid entry to be lost because the unexpected read blocked the FIFO advance.

If the DBG module is not armed and the DBGFL register is read, the TBC will store the current opcode address. Through periodic reads of the DBGFH and DBGFL registers while the DBG module is not armed, host software can provide a histogram of program execution. This is called profile mode.

#### 24.4.6 Interrupt Priority

When TRGSEL is set and the DBG module is armed to trigger on begin- or end-trigger types, a trigger is not detected in the condition where a pending interrupt occurs at the same time that a target address reaches the top of the instruction pipe. In these conditions, the pending interrupt has higher priority and code execution switches to the interrupt service routine.

When TRGSEL is clear and the DBG module is armed to trigger on end-trigger types, the trigger event is detected on a program fetch of the target address, even when an interrupt becomes pending on the same cycle. In these conditions, the pending interrupt has higher priority, the exception is processed by the core and the interrupt vector is fetched. Code execution is halted before the first instruction of the interrupt service routine is executed. In this scenario, the DBG module will have cleared ARM without having recorded the change-of-flow that occurred as part of the interrupt exception. Note that the stack will hold the return addresses and can be used to reconstruct execution flow in this scenario.

When TRGSEL is clear and the DBG module is armed to trigger on begin-trigger types, the trigger event is detected on a program fetch of the target address, even when an interrupt becomes pending on the same cycle. In this scenario, the FIFO captures the change of flow event. Because the system is configured for begin-trigger, the DBG remains armed and does not break until the FIFO has been filled by subsequent change of flow events.

### 24.5 Resets

The DBG module cannot cause an MCU reset.

There are two different ways this module will respond to reset depending upon the conditions before the reset event. If the DBG module was setup for an end trace run with DBGEN=1 and BEGIN=0, ARM, ARMF, and BRKEN are cleared but the reset function on most DBG control and status bits is overridden so a host development system can read out the results of the trace run after the MCU has been reset. In all other cases including POR, the DBG module controls are initialized to start a begin trace run starting from when the reset vector is fetched. The conditions for the default begin trace run are:

- DBGCAH=0xFF, DBGCAL=0xFE so comparator A is set to match when the 16-bit CPU address 0xFFFF appears during the reset vector fetch
- DBGC=0xC0 to enable and arm the DBG module
- DBGT=0x40 to select a force-type trigger, a BEGIN trigger, and A-only trigger mode

### 24.6 Interrupts

The DBG contains no interrupt source.

## 24.7 Electrical Specifications

The DBG module contain no electrical specifications.