

# MC9S08GW64 MC9S08GW32

## Reference Manual

### ***HCS08 Microcontrollers***

#### **Related Documentation:**

---

- **MC9S08GW64 (Data Sheet)**  
Contains pin assignments and diagrams, all electrical specifications, and mechanical drawing outlines.

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

MC9S08GW64RM  
Rev.3  
10/2010



# MC9S08GW64 Features

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

---

- New version of S08 core with same performance as traditional S08 and lower power
- Up to 20 MHz CPU at 3.6 V to 2.15 V and up to 10MHz CPU at 3.6 V to 1.8 V, across temperature range of -40 °C to 85 °C
- HC08 instruction set with added BGND instruction
- Support for up to 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 and reduced power wait mode
- Low power run and wait modes allow peripherals to run while voltage regulator is in standby
- Peripheral clock gating register can disable clocks to unused modules, thereby reducing currents
- Very low power external oscillator that can be used in stop2 or stop3 modes to provide accurate clock source to real time counter
- 6 µs typical wakeup time from stop3 mode

## Clock Source Options

---

- Oscillator (XOSC1) — Loop-control Pierce oscillator; Crystal or ceramic resonator of 32.768 kHz; Dedicated clock source for iRTC and optional for ICS
- Oscillator (XOSC2) — Loop-control Pierce oscillator; Crystal or ceramic resonator range of 31.25 kHz to 38.4 kHz or 1 MHz to 16 MHz; optional clock source for ICS
- Internal Clock Source (ICS) — Internal clock source module containing a frequency-locked-loop (FLL) controlled by internal or external reference (XOSC1, XOSC2); precision trimming of internal reference allows 0.2% resolution and 2% deviation over temperature and voltage; supporting CPU/bus frequencies from 1 MHz to 20 MHz

## System Protection

---

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1 kHz internal clock source or bus clock
- Low-voltage warning with interrupt
- Low-voltage detection with reset or interrupt
- Illegal opcode and 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 3 more breakpoints in breakpoint unit)
- Breakpoint (BKPT) debug module containing three comparators (A, B, and C) with ability to match addresses in

64 KB space. Each comparator can be used as hardware breakpoint. Full mode, Comparator A compares address and Comparator B compares data. Supports both tag and force breakpoints

## Peripherals

---

- **LCD** — up to 4×40 or 8×36 LCD driver with internal charge pump and option to provide an internally regulated LCD reference that can be trimmed for contrast control
- **ADC16** — two analog-to-digital converters; 16-bit resolution; one dedicated differential per ADC; up to 16-ch; up to 2.5 µs conversion time for 12-bit mode; automatic compare function; hardware averaging; calibration registers; temperature sensor; internal bandgap reference channel; operation in stop3; fully functional from 3.6 V to 1.8 V
- **PRACMP** — three rail to rail programmable reference analog comparator; up to 8 inputs; on-chip programmable reference generator output; selectable interrupt on rising, falling, or either edge of comparator output; operation in stop3
- **SCI** — four full duplex non-return to zero (NRZ); LIN master extended break generation; LIN slave extended break detection; wakeup on active edge; SCI0 designed for AMR operation; TxD of SCI1 and SCI2 can be modulated with timers and RxD can be received through PRACMP;
- **SPI** — three full-duplex or single-wire bidirectional; double-buffered transmit and receive; master or slave mode; MSB-first or LSB-first shifting; SPI0 designed for AMR operation;
- **IIC** — up to 100 kbps with maximum bus loading; multi-master operation; programmable slave address; interrupt driven byte-by-byte data transfer; supporting broadcast mode and 10-bit addressing; supporting SM BUS functionality; can wake up from STOP3
- **FTM** — 2-channel flextimer module; selectable input capture, output compare, or buffered edge- or center-aligned PWM on each channel
- **IRTC** — independent real-time clock, independent power domain, 32 bytes RAM, 32.768 kHz input clock optional output to ICS, hardware calendar, hardware compensation due to crystal or temperature characteristics, tamper detection and indicator
- **PCRC** — 16/32 bit programmable cyclic redundancy check for high-speed CRC calculation
- **MTIM** — two 8-bit and one 16-bit timers; configurable clock inputs and interrupt generation on overflow
- **PDB** — programmable delay block; optimized for scheduling ADC conversions
- **PCNT** — position counter; working in stop3 mode without waking CPU; can be used to generate waveforms like timer

## Input/Output

---

- 57 GPIOs including one output-only pin
- Eight KBI interrupts with selectable polarity
- Hysteresis and configurable pullup device on all input pins; configurable slew rate and drive strength on all output pins.

## Package Options

---

- 80-pin LQFP; 64-pin LQFP



---

# MC9S08GW64 MCU Series Reference Manual

Covers: MC9S08GW64  
MC9S08GW32

MC9S08GW64  
Rev.3  
10/2010

# Revision History

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

<http://freescale.com>

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

Revision Number	Revision Date	Description of Changes
2	5/21/2010	Initial public release.
3	10/26/2010	Updated the block diagram. Corrected the CRCCTRL register, IRTC_TAMPER_SCR register, IRTC_STDBY_RAM register, changed the PCNT_PWM_CH1_VAL and PCNT_PWM_CH2_VAL to PCNT_PWM_CH0_VAL and PCNT_PWM_CH1_VAL in the <a href="#">Table 4-3</a> ; corrected PRACMPxCO register in the <a href="#">Table 4-5</a> . Updated <a href="#">Figure 4-2</a> . Updated <a href="#">Table 5-2</a> for FTM vectors. Updated <a href="#">Table 17-1</a> . Updated the <a href="#">Chapter 20, "Programmable Cyclic Redundancy Check (PCRCV1)"</a> . Updated the <a href="#">Chapter 16, "Independent Real Time Clock (IRTCV2)"</a> .

This product incorporates SuperFlash® technology licensed from SST.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc.

© Freescale Semiconductor, Inc., 2009-2010. All rights reserved.

# List of Chapters

Chapter 1 Device Overview .....	29
Chapter 2 Pins and Connections .....	35
Chapter 3 Modes of Operation .....	49
Chapter 4 Memory .....	59
Chapter 5 Resets, Interrupts, and General System Control .....	99
Chapter 6 Parallel Input/Output Control .....	129
Chapter 7 Processor Unit (S08CPUV6) .....	175
Chapter 8 Keyboard Interrupt (S08KBIV2) .....	197
Chapter 9 8-Bit Serial Peripheral Interface (S08SPIV4) .....	205
Chapter 10 Serial Communication Interface (S08SCIV4) .....	223
Chapter 11 LCD Module (S08LCDLPV1) .....	243
Chapter 12 Inter-Integrated Circuit (S08IICV5) .....	295
Chapter 13 Internal Clock Source (S08ICSV4) .....	327
Chapter 14 8-Bit Modulo Timer (S08MTIMV1) .....	343
Chapter 15 16-Bit Modulo Timer (S08MTIM16V1) .....	353
Chapter 16 Independent Real Time Clock (IRTCV2) .....	363
Chapter 17 Analog-to-Digital Converter (ADC16V1) .....	403
Chapter 18 FlexTimer Module (S08FTMV3) .....	449
Chapter 19 Programmable Reference Analog Comparator (S08PRACMPV1) .....	473
Chapter 20 Programmable Cyclic Redundancy Check (PCRCV1) .....	485

<b>Chapter 21 Position Counter (S08PCNTV1) .....</b>	<b>495</b>
<b>Chapter 22 Programmable Delay Block (S08PDBV1).....</b>	<b>529</b>
<b>Chapter 23 Voltage Reference (VREFV1) .....</b>	<b>541</b>
<b>Chapter 24 Development Support .....</b>	<b>547</b>
<b>Chapter 25 Break Point Unit (BKPT) (64K) .....</b>	<b>559</b>



# Contents

Section Number	Title	Page
<b>Chapter 1</b>		
<b>Device Overview</b>		
1.1	Devices in the MC9S08GW64 Series .....	29
1.2	MCU Block Diagram .....	30
1.3	System Clock Distribution .....	32
<b>Chapter 2</b>		
<b>Pins and Connections</b>		
2.1	Introduction .....	35
2.2	Device Pin Assignment .....	35
2.3	Recommended System Connections .....	37
2.3.1	Power .....	39
2.3.2	Oscillator .....	39
2.3.3	RESET .....	40
2.3.4	IRQ .....	40
2.3.5	Background/Mode Selection (BKGD/MS) .....	40
2.3.6	V <sub>REFO</sub> Pins .....	41
2.3.7	Dedicated ADC Pins .....	41
2.3.8	AMR Pins Compatible with 5 V Interface .....	41
2.3.9	LCD Pins .....	42
2.3.9.1	LCD Power Pins .....	42
2.3.9.2	LCD Driver Pins .....	42
2.3.10	Interfacing the SCIs to Off-Chip Circuits .....	43
2.3.11	PCNT[2:0] .....	44
2.3.12	General-Purpose I/O (GPIO) and Peripheral Ports .....	45
<b>Chapter 3</b>		
<b>Modes of Operation</b>		
3.1	Introduction .....	49
3.2	Features .....	49
3.3	Run Mode .....	49
3.3.1	Low-Power Run Mode (LPRun) .....	49
3.3.1.1	Interrupts in Low-Power Run Mode .....	50
3.3.1.2	Resets in Low-Power Run Mode .....	50
3.4	Active Background Mode .....	50
3.5	Wait Mode .....	51
3.5.1	Low-Power Wait Mode (LPWait) .....	51
3.5.1.1	Interrupts in Low-Power Wait Mode .....	52
3.5.1.2	Resets in Low-Power Wait Mode .....	52

3.6	Stop Modes .....	52
3.6.1	Stop2 Mode .....	53
3.6.2	Stop3 Mode .....	54
3.6.3	Active BDM Enabled in Stop Mode .....	54
3.6.4	LVD Enabled in Stop Mode .....	54
3.6.5	Stop Modes in Low Power Run Mode .....	54
3.7	Mode Selection .....	55
3.7.1	On-Chip Peripheral Modules in Stop and Low Power Modes .....	57

## Chapter 4 Memory

4.1	Introduction .....	59
4.2	MC9S08GW64 Series Memory Map .....	59
4.3	Reset and Interrupt Vector Assignments .....	61
4.4	Register Addresses and Bit Assignments .....	62
4.5	Memory Management Unit .....	79
4.5.1	Features .....	79
4.5.2	Register Definition .....	80
4.5.2.1	Program Page Register (PPAGE) .....	80
4.5.2.2	Linear Address Pointer Registers 2:0 (LAP2:LAP0) .....	80
4.5.2.3	Linear Word Post Increment Register (LWP) .....	81
4.5.2.4	Linear Byte Post Increment Register (LBP) .....	81
4.5.2.5	Linear Byte Register (LB) .....	82
4.5.2.6	Linear Address Pointer Add Byte Register (LAPAB) .....	82
4.5.3	Functional Description .....	83
4.5.3.1	Memory Expansion .....	83
4.6	RAM .....	85
4.7	Flash .....	85
4.7.1	Features .....	85
4.7.2	Program and Erase Times .....	86
4.7.3	Program and Erase Command Execution .....	86
4.7.4	Burst Program Execution .....	88
4.7.5	Access Errors .....	90
4.7.6	Flash Block Protection .....	90
4.7.7	Vector Redirection .....	91
4.8	Security .....	91
4.9	Flash Registers and Control Bits .....	93
4.9.1	Flash Clock Divider Register (FCDIV) .....	93
4.9.2	Flash Options Register (FOPT and NVOPT) .....	94
4.9.3	Flash Configuration Register (FCNFG) .....	95
4.9.4	Flash Protection Register (FPROT and NVPROT) .....	95
4.9.5	Flash Status Register (FSTAT) .....	96
4.9.6	Flash Command Register (FCMD) .....	97

## Chapter 5

### Resets, Interrupts, and General System Control

5.1	Introduction .....	99
5.2	Features .....	99
5.3	MCU Reset .....	99
5.4	Computer Operating Properly (COP) Watchdog .....	100
5.5	Interrupts .....	101
5.5.1	Interrupt Stack Frame .....	102
5.5.2	External Interrupt Request (IRQ) Pin .....	103
5.5.2.1	Pin Configuration Options .....	103
5.5.2.2	Edge and Level Sensitivity .....	103
5.5.3	Interrupt Vectors, Sources, and Local Masks .....	104
5.6	Low-Voltage Detect (LVD) System .....	106
5.6.1	Power-On Reset Operation .....	106
5.6.2	Low-Voltage Detection (LVD) Reset Operation .....	106
5.6.3	Low-Voltage Detection (LVD) Interrupt Operation .....	106
5.6.4	Low-Voltage Warning (LVW) Interrupt Operation .....	106
5.7	Peripheral Clock Gating .....	106
5.8	Reset, Interrupt, and System Control Registers and Control Bits .....	107
5.8.1	Interrupt Pin Request Status and Control Register (IRQSC) .....	107
5.8.2	System Reset Status Register (SRS) .....	109
5.8.3	System Background Debug Force Reset Register (SBDFR) .....	110
5.8.4	System Options Register 1 (SOPT1) .....	111
5.8.5	Internal Peripheral Select Register 1 (SIMIPS1) .....	112
5.8.6	Internal Peripheral Select Register 2 (SIMIPS2) .....	113
5.8.7	Internal Peripheral Select Register 3 (SIMIPS3) .....	114
5.8.8	SIM Clock Options Register (SIMCO) .....	115
5.8.9	Clock Check and Select Control Register (CCSCTRL) .....	116
5.8.10	CLKOUT Prescaler Register (CLK_PRSC_H, CLK_PRSC_L) .....	117
5.8.11	System Device Identification Register (SDIDH, SDIDL) .....	118
5.8.12	System Power Management Status and Control 1 Register (SPMSC1) .....	119
5.8.13	System Power Management Status and Control 2 Register (SPMSC2) .....	120
5.8.14	System Power Management Status and Control 3 Register (SPMSC3) .....	121
5.8.15	System Clock Gating Control 1 Register (SCGC1) .....	122
5.8.16	System Clock Gating Control 2 Register (SCGC2) .....	123
5.8.17	System Clock Gating Control 3 Register (SCGC3) .....	125
5.8.18	System Clock Gating Control 4 Register (SCGC4) .....	126
5.8.19	System Clock Gating Control 5 Register (SCGC5) .....	127

## Chapter 6

### Parallel Input/Output Control

6.1	Introduction .....	129
6.2	Port Data and Data Direction .....	129
6.3	Pullup, Slew Rate, and Drive Strength .....	130
6.3.1	Port Internal Pullup Enable .....	130

6.3.2	Port Slew Rate Enable .....	131
6.3.3	Port Drive Strength Select .....	131
6.4	Open Drain Operation .....	131
6.5	Pin Behavior in Stop Modes .....	131
6.6	Parallel I/O and Pin Control Registers .....	132
6.6.1	Port A Registers .....	133
6.6.1.1	Port A Data Register (PTAD) .....	133
6.6.1.2	Port A Data Direction Register (PTADD) .....	134
6.6.1.3	Port A Pull Enable Register (PTAPE) .....	135
6.6.1.4	Port A Slew Rate Enable Register (PTASE) .....	135
6.6.1.5	Port A Drive Strength Selection Register (PTADS) .....	136
6.6.1.6	Port A Input Filter Enable Register (PTAIFE) .....	136
6.6.2	Port B Registers .....	137
6.6.2.1	Port B Data Register (PTBD) .....	137
6.6.2.2	Port B Data Direction Register (PTBDD) .....	137
6.6.2.3	Port B Pull Enable Register (PTBPE) .....	138
6.6.2.4	Port B Slew Rate Enable Register (PTBSE) .....	138
6.6.2.5	Port B Drive Strength Selection Register (PTBDS) .....	139
6.6.2.6	Port B Input Filter Enable Register (PTBIFE) .....	139
6.6.3	Port C Registers .....	140
6.6.3.1	Port C Data Register (PTCD) .....	140
6.6.3.2	Port C Data Direction Register (PTCDD) .....	140
6.6.3.3	Port C Pull Enable Register (PTCPE) .....	141
6.6.3.4	Port C Slew Rate Enable Register (PTCSE) .....	141
6.6.3.5	Port C Drive Strength Selection Register (PTCDS) .....	142
6.6.3.6	Port C Input Filter Enable Register (PTCIFE) .....	142
6.6.4	Port D Registers .....	143
6.6.4.1	Port D Data Register (PTDD) .....	143
6.6.4.2	Port D Data Direction Register (PTDDD) .....	143
6.6.4.3	Port D Pull Enable Register (PTDPE) .....	144
6.6.4.4	Port D Slew Rate Enable Register (PTDSE) .....	144
6.6.4.5	Port D Drive Strength Selection Register (PTDDS) .....	145
6.6.4.6	Port D Input Filter Enable Register (PTDIFE) .....	145
6.6.5	Port E Registers .....	146
6.6.5.1	Port E Data Register (PTED) .....	146
6.6.5.2	Port E Data Direction Register (PTEDD) .....	146
6.6.5.3	Port E Pull Enable Register (PTEPE) .....	147
6.6.5.4	Port E Slew Rate Enable Register (PTESE) .....	147
6.6.5.5	Port E Input Filter Enable Register (PTEIFE) .....	147
6.6.6	Port F Registers .....	148
6.6.6.1	Port F Data Register (PTFD) .....	148
6.6.6.2	Port F Data Direction Register (PTFDD) .....	148
6.6.6.3	Port F Pull Enable Register (PTFPE) .....	149
6.6.6.4	Port F Slew Rate Enable Register (PTFSE) .....	149
6.6.6.5	Port F Input Filter Enable Register (PTFIFE) .....	149

6.6.7	Port G Registers .....	150
6.6.7.1	Port G Data Register (PTGD) .....	150
6.6.7.2	Port G Data Direction Register (PTGDD) .....	150
6.6.7.3	Port G Pull Enable Register (PTGPE) .....	151
6.6.7.4	Port G Slew Rate Enable Register (PTGSE) .....	151
6.6.7.5	Port G Input Filter Enable Register (PTGIFE) .....	151
6.6.8	Port H Registers .....	152
6.6.8.1	Port H Data Register (PTHD) .....	152
6.6.8.2	Port H Data Direction Register (PTHDD) .....	152
6.6.8.3	Port H Pull Enable Register (PTHPE) .....	153
6.6.8.4	Port H Slew Rate Enable Register (PTHSE) .....	153
6.6.8.5	Port H Input Filter Enable Register (PTHIFE) .....	153
6.7	Pin Mux Controls .....	154
6.7.1	Port A Pin Function Register 1 (PTAPF1) .....	156
6.7.2	Port A Pin Function Register 2 (PTAPF2) .....	157
6.7.3	Port A Pin Function Register 3 (PTAPF3) .....	158
6.7.4	Port A Pin Function Register 4 (PTAPF4) .....	158
6.7.5	Port B Pin Function Register 1 (PTBPF1) .....	159
6.7.6	Port B Pin Function Register 2 (PTBPF2) .....	160
6.7.7	Port B Pin Function Register 3 (PTBPF3) .....	161
6.7.8	Port B Pin Function Register 4 (PTBPF4) .....	161
6.7.9	Port C Pin Function Register 1 (PTCPF1) .....	162
6.7.10	Port C Pin Function Register 2 (PTCPF2) .....	162
6.7.11	Port C Pin Function Register 3 (PTCPF3) .....	163
6.7.12	Port C Pin Function Register 4 (PTCPF4) .....	163
6.7.13	Port D Pin Function Register 1 (PTDPF1) .....	164
6.7.14	Port D Pin Function Register 2 (PTDPF2) .....	164
6.7.15	Port D Pin Function Register 3 (PTDPF3) .....	165
6.7.16	Port D Pin Function Register 4 (PTDPF4) .....	165
6.7.17	Port E Pin Function Register 1 (PTEPF1) .....	166
6.7.18	Port E Pin Function Register 2 (PTEPF2) .....	166
6.7.19	Port E Pin Function Register 3 (PTEPF3) .....	167
6.7.20	Port E Pin Function Register 4 (PTEPF4) .....	167
6.7.21	Port F Pin Function Register 1 (PTFPF1) .....	168
6.7.22	Port F Pin Function Register 2 (PTFPF2) .....	168
6.7.23	Port F Pin Function Register 3 (PTFPF3) .....	169
6.7.24	Port F Pin Function Register 4 (PTFPF4) .....	169
6.7.25	Port G Pin Function Register 1 (PTGPF1) .....	170
6.7.26	Port G Pin Function Register 2 (PTGPF2) .....	170
6.7.27	Port G Pin Function Register 3 (PTGPF3) .....	171
6.7.28	Port G Pin Function Register 4 (PTGPF4) .....	172
6.7.29	Port H Pin Function Register 1 (PTHPF1) .....	173

## Chapter 7

### Central Processor Unit (S08CPUV6)

7.1	Introduction .....	175
7.1.1	Features .....	175
7.2	Programmer's Model and CPU Registers .....	176
7.2.1	Accumulator (A) .....	176
7.2.2	Index Register (H:X) .....	176
7.2.3	Stack Pointer (SP) .....	177
7.2.4	Program Counter (PC) .....	177
7.2.5	Condition Code Register (CCR) .....	177
7.3	Addressing Modes .....	179
7.3.1	Inherent Addressing Mode (INH) .....	179
7.3.2	Relative Addressing Mode (REL) .....	179
7.3.3	Immediate Addressing Mode (IMM) .....	179
7.3.4	Direct Addressing Mode (DIR) .....	180
7.3.5	Extended Addressing Mode (EXT) .....	180
7.3.6	Indexed Addressing Mode .....	180
7.3.6.1	Indexed, No Offset (IX) .....	180
7.3.6.2	Indexed, No Offset with Post Increment (IX+) .....	180
7.3.6.3	Indexed, 8-Bit Offset (IX1) .....	180
7.3.6.4	Indexed, 8-Bit Offset with Post Increment (IX1+) .....	180
7.3.6.5	Indexed, 16-Bit Offset (IX2) .....	181
7.3.6.6	SP-Relative, 8-Bit Offset (SP1) .....	181
7.3.6.7	SP-Relative, 16-Bit Offset (SP2) .....	181
7.4	Special Operations .....	181
7.4.1	Reset Sequence .....	181
7.4.2	Interrupt Sequence .....	181
7.4.3	Wait Mode Operation .....	182
7.4.4	Stop Mode Operation .....	182
7.4.5	BGND Instruction .....	183
7.5	HCS08 Instruction Set Summary .....	185

## Chapter 8

### Keyboard Interrupt (S08KBIV2)

8.1	Introduction .....	197
8.1.1	KBI Clock Gating .....	197
8.1.2	Features .....	199
8.1.3	Modes of Operation .....	199
8.1.3.1	KBI in Wait Mode .....	199
8.1.3.2	KBI in Stop Modes .....	199
8.1.3.3	KBI in Active Background Mode .....	199
8.1.4	Block Diagram .....	199
8.2	External Signal Description .....	200
8.3	Register Definition .....	200
8.3.1	KBI Status and Control Register (KBISC) .....	200

8.3.2	KBI Pin Enable Register (KBIPE)	201
8.3.3	KBI Edge Select Register (KBIES)	201
8.4	Functional Description	202
8.4.1	Edge Only Sensitivity	202
8.4.2	Edge and Level Sensitivity	202
8.4.3	KBI Pullup/Pulldown Resistors	203
8.4.4	KBI Initialization	203

## Chapter 9

### 8-Bit Serial Peripheral Interface (S08SPIV4)

9.1	Introduction	205
9.1.1	AMR SPI0	205
9.1.2	SPI Clock Gating	205
9.1.3	Features	207
9.1.4	Block Diagrams	207
9.1.4.1	SPI System Block Diagram	207
9.1.4.2	SPI Module Block Diagram	208
9.1.5	SPI Baud Rate Generation	209
9.2	External Signal Description	210
9.2.1	SPSCK — SPI Serial Clock	210
9.2.2	MOSI — Master Data Out, Slave Data In	210
9.2.3	MISO — Master Data In, Slave Data Out	210
9.2.4	$\overline{SS}$ — Slave Select	210
9.3	Modes of Operation	211
9.3.1	SPI in Stop Modes	211
9.4	Register Definition	211
9.4.1	SPI Control Register 1 (SPIC1)	211
9.4.2	SPI Control Register 2 (SPIC2)	212
9.4.3	SPI Baud Rate Register (SPIBR)	213
9.4.4	SPI Status Register (SPIS)	214
9.4.5	SPI Data Register (SPID)	215
9.5	Functional Description	216
9.5.1	General	216
9.5.2	Master Mode	216
9.5.3	Slave Mode	217
9.5.4	SPI Clock Formats	218
9.5.5	Special Features	220
9.5.5.1	SS Output	220
9.5.5.2	Bidirectional Mode (MOMI or SISO)	221
9.5.6	SPI Interrupts	222
9.5.7	Mode Fault Detection	222

## Chapter 10

### Serial Communication Interface (S08SCIV4)

10.1	Introduction	223
------	--------------	-----

10.1.1	AMR SCI0 .....	223
10.1.2	SCI1 and SCI2 Internal Connection .....	223
10.1.3	SCI Clock Gating .....	223
10.1.4	Features .....	225
10.1.5	Modes of Operation .....	225
10.1.6	Block Diagram .....	226
10.2	Register Definition .....	228
10.2.1	SCI Baud Rate Registers (SCIxBDH, SCIxBDL) .....	228
10.2.2	SCI Control Register 1 (SCIxC1) .....	229
10.2.3	SCI Control Register 2 (SCIxC2) .....	230
10.2.4	SCI Status Register 1 (SCIxS1) .....	231
10.2.5	SCI Status Register 2 (SCIxS2) .....	233
10.2.6	SCI Control Register 3 (SCIxC3) .....	234
10.2.7	SCI Data Register (SCIxD) .....	235
10.3	Functional Description .....	235
10.3.1	Baud Rate Generation .....	235
10.3.2	Transmitter Functional Description .....	236
10.3.2.1	Send Break and Queued Idle .....	237
10.3.3	Receiver Functional Description .....	237
10.3.3.1	Data Sampling Technique .....	238
10.3.3.2	Receiver Wakeup Operation .....	238
10.3.4	Interrupts and Status Flags .....	239
10.3.5	Additional SCI Functions .....	240
10.3.5.1	8- and 9-Bit Data Modes .....	240
10.3.5.2	Stop Mode Operation .....	241
10.3.5.3	Loop Mode .....	241
10.3.5.4	Single-Wire Operation .....	241

## Chapter 11

### LCD Module (S08LCDLPV1)

11.1	Introduction .....	243
11.1.1	LCD Clock Sources .....	243
11.1.2	LCD Modes of Operation .....	244
11.1.3	LCD Status after Stop2 Wakeup .....	244
11.1.4	LCD Clock Gating .....	244
11.1.5	Features .....	246
11.1.6	Modes of Operation .....	246
11.1.7	Block Diagram .....	248
11.2	External Signal Description .....	248
11.2.1	LCD[63:0] .....	249
11.2.2	V <sub>LCD</sub> .....	249
11.2.3	V <sub>LL1</sub> , V <sub>LL2</sub> , V <sub>LL3</sub> .....	249
11.2.4	V <sub>cap1</sub> , V <sub>cap2</sub> .....	249
11.3	Register Definition .....	249
11.3.1	LCD Control Register 0 (LCDC0) .....	249



11.3.2	LCD Control Register 1 (LCDC1)	250
11.3.3	LCD Voltage Supply Register (LCDSUPPLY)	251
11.3.4	LCD Regulated Voltage Control Register (LCDRVC)	252
11.3.5	LCD Blink Control Register (LCDBCTL)	253
11.3.6	LCD Status Register (LCDS)	254
11.3.7	LCD Pin Enable Registers 0–7 (LCDPEN0–LCDPEN7)	254
11.3.8	Backplane Enable Registers 0–7 (BPEN0–BPEN7)	255
11.3.9	LCD Waveform Registers (LCDWF[63:0])	257
11.4	Functional Description	262
11.4.1	LCD Driver Description	263
11.4.1.1	LCD Duty Cycle	263
11.4.1.2	LCD Bias	264
11.4.1.3	LCD Module Base Clock and Frame Frequency	264
11.4.1.4	LCD Waveform Examples	266
11.4.2	LCDWF Registers	271
11.4.3	LCD Display Modes	271
11.4.3.1	LCD Blink Modes	272
11.4.3.2	Blink Frequency	272
11.4.4	LCD Charge Pump, Voltage Divider, and Power Supply Operation	273
11.4.4.1	LCD Charge Pump and Voltage Divider	275
11.4.4.2	LCD Power Supply and Voltage Buffer Configuration	276
11.4.5	Resets	280
11.4.6	Interrupts	281
11.5	Initialization Section	281
11.5.1	Initialization Sequence	281
11.5.2	Initialization Examples	284
11.5.2.1	Initialization Example 1	284
11.5.2.2	Initialization Example 2	286
11.5.2.3	Initialization Example 3	287
11.6	Application Information	288
11.6.1	LCD Seven Segment Example Description	289
11.6.1.1	LCD Module Waveforms	291
11.6.1.2	Segment On Driving Waveform	292
11.6.1.3	Segment Off Driving Waveform	292
11.6.2	LCD Contrast Control	292
11.6.3	Stop Mode Recovery	293

## Chapter 12

### Inter-Integrated Circuit (S08IICV5)

12.1	Introduction	295
12.1.1	IIC Clock Gating	295
12.1.2	Features	297
12.1.3	Modes of Operation	297
12.1.4	Block Diagram	297
12.2	External Signal Description	298

12.2.1	SCL — Serial Clock Line .....	298
12.2.2	SDA — Serial Data Line .....	298
12.3	Register Definition .....	299
12.3.1	Module Memory Map .....	299
12.3.2	IIC Address Register 1 (IICA1) .....	299
12.3.3	IIC Frequency Divider Register (IICF) .....	300
12.3.4	IIC Control Register (IICC1) .....	303
12.3.5	IIC Status Register (IICS) .....	304
12.3.6	IIC Data I/O Register (IICD) .....	305
12.3.7	IIC Control Register 2 (IICC2) .....	306
12.3.8	IIC Programmable Input Glitch Filter (IICFLT) .....	306
12.3.9	IIC SMBus Control and Status Register (IICSMB) .....	308
12.3.10	IIC Address Register 2 (IICA2) .....	309
12.3.11	IIC SCL Low Time Out Register High (IICSLTH) .....	309
12.3.12	IIC SCL Low Time Out register Low (IICSLTL) .....	310
12.4	Functional Description .....	311
12.4.1	IIC Protocol .....	311
12.4.1.1	START Signal .....	311
12.4.1.2	Slave Address Transmission .....	312
12.4.1.3	Data Transfer .....	312
12.4.1.4	STOP Signal .....	312
12.4.1.5	Repeated START Signal .....	313
12.4.1.6	Arbitration Procedure .....	313
12.4.1.7	Clock Synchronization .....	313
12.4.1.8	Handshaking .....	314
12.4.1.9	Clock Stretching .....	314
12.4.2	10-bit Address .....	315
12.4.2.1	Master-Transmitter Addresses a Slave-Receiver .....	315
12.4.2.2	Master-Receiver Addresses a Slave-Transmitter .....	315
12.4.3	Address Matching .....	316
12.4.4	System Management Bus Specification .....	316
12.4.4.1	Timeouts .....	316
12.4.4.2	FAST ACK and NACK .....	318
12.5	Resets .....	318
12.6	Interrupts .....	318
12.6.1	Byte Transfer Interrupt .....	319
12.6.2	Address Detect Interrupt .....	319
12.6.3	Exit from Low-Power/Stop Modes .....	319
12.6.4	Arbitration Lost Interrupt .....	319
12.6.5	Timeouts Interrupt in SMBus .....	320
12.6.6	Programmable Input Glitch Filter .....	320
12.6.7	Address Matching Wakeup .....	320
12.7	Initialization/Application Information .....	321
12.8	SMBALERT# .....	325

## Chapter 13

### Internal Clock Source (S08ICSV4)

13.1	Introduction .....	327
13.1.1	Features .....	329
13.1.2	Block Diagram .....	329
13.1.3	Modes of Operation .....	330
13.1.3.1	FLL Engaged Internal (FEI) .....	330
13.1.3.2	FLL Engaged External (FEE) .....	330
13.1.3.3	FLL Bypassed Internal (FBI) .....	330
13.1.3.4	FLL Bypassed Internal Low Power (FBILP) .....	331
13.1.3.5	FLL Bypassed External (FBE) .....	331
13.1.3.6	FLL Bypassed External Low Power (FBELP) .....	331
13.1.3.7	Stop (STOP) .....	331
13.2	External Signal Description .....	331
13.3	Register Definition .....	331
13.3.1	ICS Control Register 1 (ICSC1) .....	332
13.3.2	ICS Control Register 2 (ICSC2) .....	334
13.3.3	ICS Trim Register (ICSTRM) .....	334
13.3.4	ICS Status and Control (ICSSC) .....	335
13.4	Functional Description .....	337
13.4.1	Operational Modes .....	337
13.4.1.1	FLL Engaged Internal (FEI) .....	337
13.4.1.2	FLL Engaged External (FEE) .....	338
13.4.1.3	FLL Bypassed Internal (FBI) .....	338
13.4.1.4	FLL Bypassed Internal Low Power (FBILP) .....	338
13.4.1.5	FLL Bypassed External (FBE) .....	338
13.4.1.6	FLL Bypassed External Low Power (FBELP) .....	339
13.4.1.7	Stop .....	339
13.4.2	Mode Switching .....	339
13.4.3	Bus Frequency Divider .....	340
13.4.4	Low Power Bit Usage .....	340
13.4.5	DCO Maximum Frequency with 32.768 kHz Oscillator .....	340
13.4.6	Internal Reference Clock .....	340
13.4.7	External Reference Clock .....	341
13.4.8	Fixed Frequency Clock .....	341
13.4.9	Local Clock .....	341

## Chapter 14

### 8-Bit Modulo Timer (S08MTIMV1)

14.1	Introduction .....	343
14.1.1	MTIM Clock Gating .....	343
14.1.2	Features .....	345
14.1.3	Modes of Operation .....	345
14.1.3.1	MTIM in Wait Mode .....	345
14.1.3.2	MTIM in Stop Modes .....	345

14.1.3.3	MTIM in Active Background Mode .....	345
14.1.4	Block Diagram .....	346
14.2	External Signal Description .....	346
14.3	Register Definition .....	346
14.3.1	MTIMx Status and Control Register (MTIMxSC) .....	348
14.3.2	MTIMx Clock Configuration Register (MTIMxCLK) .....	349
14.3.3	MTIMx Counter Register (MTIMxCNT) .....	350
14.3.4	MTIMx Modulo Register (MTIMxMOD) .....	350
14.4	Functional Description .....	351
14.4.1	MTIM Operation Example .....	352

## Chapter 15

### 16-Bit Modulo Timer (S08MTIM16V1)

15.1	Introduction .....	353
15.1.1	MTIM3 Clock Gating .....	353
15.2	Features .....	355
15.2.1	Block Diagram .....	355
15.2.2	Modes of Operation .....	355
15.2.2.1	MTIM16 in Wait Mode .....	355
15.2.2.2	MTIM16 in Stop Modes .....	356
15.2.2.3	MTIM16 in Active Background Mode .....	356
15.3	External Signal Description .....	356
15.3.1	TCLK — External Clock Source Input into MTIM16 .....	356
15.4	Register Definition .....	356
15.4.1	MTIMx16 Status and Control Register (MTIMxSC) .....	357
15.4.2	MTIM16 Clock Configuration Register (MTIMxCLK) .....	357
15.4.3	MTIM16 Counter Register High/Low (MTIMxCNTH:L) .....	358
15.4.4	MTIM16 Modulo Register High/Low (MTIMxMODH/MTIMxMODL) .....	359
15.5	Functional Description .....	360
15.5.1	MTIM16 Operation Example .....	361

## Chapter 16

### Independent Real Time Clock (IRTCV2)

16.1	Introduction .....	363
16.1.1	IRTC Power Supply Source .....	363
16.1.2	IRTC Clock Gating .....	364
16.2	Features .....	366
16.3	Modes of Operation .....	367
16.3.1	Low Power Modes .....	368
16.3.1.1	Run Mode .....	368
16.3.1.2	Wait Mode .....	368
16.3.1.3	Stop Mode .....	368
16.4	External Signal Description .....	368
16.5	Register Definitions .....	368
16.5.1	IRTC Year & Month Counters Register (IRTC_YEARMON) .....	369

16.5.2	IRTC Day & Day-of-Week Counters Register (IRTC_DAYS)	370
16.5.3	IRTC Hours and Minutes Counters Register (IRTC_HOURMIN)	371
16.5.4	IRTC Seconds Counter Register (IRTC_SECONDS)	372
16.5.5	IRTC Year and Month Alarm Register (IRTC_ALM_YRMON)	372
16.5.6	IRTC Days Alarm Register (IRTC_ALM_DAYS)	373
16.5.7	IRTC Hours and Minutes Alarm Register (IRTC_ALM_HM)	374
16.5.8	IRTC Seconds Alarm Register (IRTC_ALM_SECS)	375
16.5.9	IRTC Control Register (IRTC_CTRL)	376
16.5.10	IRTC Status Register (IRTC_STATUS)	377
16.5.11	IRTC Interrupt Status Register (IRTC_ISR)	378
16.5.12	IRTC Interrupt Enable Register (IRTC_IER)	380
16.5.13	IRTC Minutes Down Counter Register (IRTC_COUNT_DN)	382
16.5.14	IRTC Daylight Saving Hour Register (IRTC_DST_HOUR)	383
16.5.14.1	IRTC Daylight Saving Month Register (IRTC_DST_MNTH)	384
16.5.15	IRTC Daylight Saving Day Register (IRTC_DST_DAY)	385
16.5.16	IRTC Compensation (IRTC_COMPEN)	385
16.5.17	IRTC Tamper Time Stamp Year & Month Register (IRTC_TTSR_YM)	386
16.5.18	IRTC Tamper Time Stamp Day Register (IRTC_TTSR_DAY)	387
16.5.19	IRTC Tamper Time Stamp Hours & Minutes Register	388
16.5.20	IRTC Tamper Time Stamp Seconds Register (IRTC_TTSR_SEC)	388
16.5.21	IRTC Tamper Status & Control Register (IRTC_TAMPER_SCR)	389
16.5.22	IRTC Tamper Filter 0 & 1 Control Register (IRTC_FILTER01_CTRL)	390
16.5.23	IRTC Tamper Filter 2 & 3 Control Register (IRTC_FILTER23_CTRL)	391
16.5.24	IRTC Standby RAM (IRTC_STDBY_RAM)	392
16.6	Functional Description	393
16.6.1	Compensation Logic	393
16.6.2	Write Protection Logic	394
16.6.3	Tamper Detection Logic	395
16.6.4	IRTC Control Logic	398
16.6.5	IRTC Isolation Logic	398
16.6.6	Battery Power Architecture	399
16.7	Design Assumptions	400

## Chapter 17

### Analog-to-Digital Converter (ADC16V1)

17.1	Introduction	403
17.1.1	ADC Clock Gating	403
17.1.2	Hardware Trigger	403
17.1.3	ADC Alternate Clock	404
17.2	ADC Connections	404
17.2.1	Features	407
17.2.2	Block Diagram	407
17.3	External Signal Description	408
17.3.1	Analog Power ( $V_{DDA}$ )	409
17.3.2	Analog Ground ( $V_{SSA}$ )	409

17.3.3	Voltage Reference Select High ( $V_{REFSH}$ )	409
17.3.4	Voltage Reference Select Low ( $V_{REFL}$ )	409
17.3.5	Analog Channel Inputs (ADx)	410
17.3.6	Differential Analog Channel Inputs (DADx)	410
17.4	Register Definition	410
17.4.1	Status and Control Registers 1 (ADCSC1A:ADCSC1n)	410
17.4.2	Configuration Register 1 (ADCCFG1)	412
17.4.3	Configuration Register 2 (ADCCFG2)	414
17.4.4	Data Result Registers (ADCRHA:ADCRLA to ADCRHn:ADCRLn)	415
17.4.5	Compare Value Registers (ADCCV1H:ADCCV1L & ADCCV2H:ADCCV2L)	416
17.4.6	Status and Control Register 2 (ADCSC2)	417
17.4.7	Status and Control Register 3 (ADCSC3)	419
17.4.8	ADC Offset Correction Register (ADCOFSH:ADCOFSL)	419
17.4.9	ADC Plus-Side Gain Register (ADCPGH:ADCPGL)	420
17.4.10	ADC Minus-Side Gain Register (ADCMGH:ADCMGL)	420
17.4.11	ADC Plus-Side General Calibration Value Registers (ADCCLPx)	421
17.4.12	ADC Minus-Side General Calibration Value Registers (ADCCLMx)	423
17.4.13	Pin Control 1 Register (APCTL1)	424
17.4.14	Pin Control 2 Register (APCTL2)	425
17.4.15	Pin Control 3 Register (APCTL3)	426
17.4.16	Pin Control 4 Register (APCTL4)	427
17.5	Functional Description	428
17.5.1	Clock Select and Divide Control	429
17.5.2	Input Select and Pin Control	429
17.5.3	Voltage Reference Selection	429
17.5.4	Hardware Trigger and Channel Selects	430
17.5.5	Conversion Control	430
17.5.5.1	Initiating Conversions	430
17.5.5.2	Completing Conversions	431
17.5.5.3	Aborting Conversions	432
17.5.5.4	Power Control	432
17.5.5.5	Sample Time and Total Conversion Time	433
17.5.5.6	Conversion Time Examples	435
17.5.5.7	Hardware Average Function	436
17.5.6	Automatic Compare Function	436
17.5.7	Calibration Function	437
17.5.8	User Defined Offset Function	439
17.5.9	Temperature Sensor	440
17.5.10	MCU Wait Mode Operation	440
17.5.11	MCU Stop3 Mode Operation	440
17.5.11.1	Stop3 Mode With ADACK Disabled	441
17.5.11.2	Stop3 Mode With ADACK Enabled	441
17.5.12	MCU Stop2 Mode Operation	441
17.6	Initialization Information	441
17.6.1	ADC Module Initialization Example	442

17.6.1.1	Initialization Sequence .....	442
17.6.1.2	Pseudo-Code Example .....	442
17.7	Application Information .....	443
17.7.1	External Pins and Routing .....	444
17.7.1.1	Analog Supply Pins .....	444
17.7.1.2	Analog Voltage Reference Pins .....	444
17.7.1.3	Analog Input Pins .....	445
17.7.2	Sources of Error .....	445
17.7.2.1	Sampling Error .....	445
17.7.2.2	Pin Leakage Error .....	445
17.7.2.3	Noise-Induced Errors .....	446
17.7.2.4	Code Width and Quantization Error .....	446
17.7.2.5	Linearity Errors .....	447
17.7.2.6	Code Jitter, Non-Monotonicity, and Missing Codes .....	447

## Chapter 18

### FlexTimer Module (S08FTMV3)

18.1	Introduction .....	449
18.1.1	FTM External Clock Source .....	449
18.1.2	FTM Valid Function .....	449
18.1.3	FTM Clock Gating .....	449
18.1.4	Features .....	451
18.1.5	Modes of Operation .....	451
18.1.6	Block Diagram .....	451
18.2	Signal Description .....	452
18.2.1	EXTCLK — FTM External Clock .....	453
18.2.2	FTMCHn — FTM Channel (n) I/O Pin .....	453
18.3	Memory Map and Register Definition .....	454
18.3.1	Module Memory Map .....	454
18.3.2	Register Descriptions .....	455
18.3.3	FTM Status and Control Register (FTMSC) .....	455
18.3.4	FTM Counter Registers (FTMCNTH:FTMCNTL) .....	456
18.3.5	FTM Counter Modulo Registers (FTMMODH:FTMMODL) .....	457
18.3.6	FTM Channel (n) Status and Control Register (FTMCnSC) .....	458
18.3.7	FTM Channel Value Registers (FTMCnVH:FTMCnVL) .....	459
18.4	Functional Description .....	460
18.4.1	Clock Source .....	461
18.4.1.1	Counter Clock Source .....	461
18.4.2	Prescaler .....	462
18.4.3	Counter .....	462
18.4.3.1	Up Counting .....	462
18.4.3.2	Up-Down Counting .....	463
18.4.3.3	Free Running Counter .....	464
18.4.3.4	Counter Reset .....	464
18.4.4	Input Capture Mode .....	464

18.4.5	Output Compare Mode .....	465
18.4.6	Edge-Aligned PWM (EPWM) Mode .....	467
18.4.7	Center-Aligned PWM (CPWM) Mode .....	468
18.4.8	Load of the Registers With Write Buffers .....	470
18.4.9	BDM Mode .....	471
18.5	Reset Overview .....	471
18.6	FTM Interrupts .....	471
18.6.1	Timer Overflow Interrupt .....	471
18.6.2	Channel (n) Interrupt .....	471

## Chapter 19

### Programmable Reference Analog Comparator (S08PRACMPV1)

19.1	Introduction .....	473
19.1.1	PRACMP Input Pins CMPPx and Output pin CMPOUTx .....	473
19.1.2	PRACMP Internal Connection .....	473
19.1.3	PRACMP Clock Gating .....	474
19.1.4	Features .....	476
19.1.5	Modes of Operation .....	476
19.1.5.1	Operation in Wait Mode .....	476
19.1.5.2	Operation in Stop Mode .....	476
19.1.5.3	Operation in Background Mode .....	476
19.1.6	Block Diagram .....	477
19.2	External Signal Description .....	478
19.3	Memory Map and Register Definition .....	478
19.3.1	PRACMP Control and Status Register (PRACMPxCS) .....	478
19.3.2	PRACMP Control Register 0 (PRACMPxC0) .....	479
19.3.3	PRACMP Control Register 1 (PRACMPxC1) .....	480
19.3.4	PRACMP Control Register 2 (PRACMPxC2) .....	482
19.4	Functional Description .....	482
19.5	Setup and Operation of PRACMP .....	483
19.6	Resets .....	483
19.7	Interrupts .....	483

## Chapter 20

### Programmable Cyclic Redundancy Check (PCRCV1)

20.1	Introduction .....	485
20.1.1	PCRC Clock Gating .....	485
20.1.2	Features .....	487
20.1.3	Block Diagram .....	487
20.1.4	Modes of Operation .....	487
20.1.4.1	Run Mode .....	487
20.1.4.2	Low Power Modes (Wait or Stop) .....	487
20.2	External Signals Description .....	488
20.3	Memory Map and Register Definition .....	488
20.3.1	CRC Data Register (CRCDH1:CRCDH0:CRCDL1:CRCDL0) .....	488



20.3.2	CRC Polynomial Register (CRCPH1:CRCPH0:CRCPL1:CRCPL0)	489
20.3.3	CRC Control Register (CRCCTL)	490
20.4	Functional Description	491
20.4.1	CRC Initialization/Re-Initialization	491
20.4.2	CRC Calculations	491
20.4.2.1	16-bit CRC	491
20.4.2.2	32-bit CRC	492
20.4.3	Transpose Feature	492
20.4.3.1	Types of Transpose	492
20.4.4	CRC Result Complement	493

## Chapter 21

### Position Counter (S08PCNTV1)

21.1	Introduction	495
21.1.1	PCNT Internal Connection	495
21.1.2	PCNT Clock Gating	495
21.1.3	Features	498
21.1.4	Modes of Operation	498
21.1.5	Low Power Modes	499
21.1.5.1	Run Mode	499
21.1.5.2	Wait Mode	499
21.1.5.3	Stop Mode	499
21.1.6	Block Diagram	499
21.2	External Signal Description	500
21.3	Register Definition	502
21.3.1	PCounter Status Register (PCNT_STATUS)	502
21.3.2	PCounter Control Register (PCNT_CTRL)	503
21.3.3	PCounter Forward Counter Modulus Register (PCNT_FCMOD)	505
21.3.4	PCounter Forward Counter Register (PCNT_FCNT)	506
21.3.5	PCounter Reverse Counter Modulus Register (PCNT_RCMOD)	506
21.3.6	PCounter Reverse Counter Register (PCNT_RCNT)	507
21.3.7	PCounter PWM Modulus Register (PCNT_PWM_MOD)	507
21.3.8	PCounter PWM Channel 0 Value Register (PCNT_PWM_CH0_VAL)	508
21.3.9	PCounter PWM Channel 1 Value Register (PCNT_PWM_CH1_VAL)	508
21.3.10	PCounter State Register (PCNT_STATE)	509
21.4	Functional Description	509
21.4.1	Modes of Operation	509
21.4.1.1	Single Direct Counter Mode	509
21.4.1.2	180 Degree Dual Sensor Mode	510
21.4.1.3	Two-Signal Binary Mode	511
21.4.1.4	Two-Signal Gray Mode	513
21.4.1.5	Three-Signal Binary Mode	515
21.4.1.6	Three-Signal Gray Mode	517
21.4.1.7	Atomic Counter/PWM Mode (Edge- or Centre-Aligned)	519
21.4.2	Noise Filtering	519

21.4.3	Synchronous and Asynchronous Interrupt of PCounter .....	519
21.4.3.1	Synchronous Interrupt .....	519
21.4.3.2	Asynchronous Interrupt .....	520
21.4.4	Sampling vs Level Sensing .....	520
21.4.5	PCounter Internal PWM .....	520
21.4.5.1	Edge-Aligned PWM .....	521
21.4.5.2	Centre-Aligned PWM .....	522
21.4.6	Sensor Activation Signal & Internal Sampling Signal Generation .....	523
21.4.7	Initialization Sequence .....	525
21.5	Typical Use Case .....	526

## Chapter 22

### Programmable Delay Block (S08PDBV1)

22.1	Introduction .....	529
22.1.1	PDB Clock Gating .....	530
22.1.2	Features .....	532
22.1.3	Modes of Operation .....	532
22.1.4	Block Diagram .....	532
22.2	Memory Map and Registers .....	535
22.2.1	Memory Map .....	535
22.2.2	Registers Descriptions .....	535
22.2.2.1	PDB Status and Control Register (PDBSC) .....	535
22.2.2.2	PDB Modulus Register (PDBMOD) .....	537
22.2.2.3	PDB Counter Register (PDBCNT) .....	537
22.2.2.4	PDB Interrupt Delay Register (PDBIDLY) .....	537
22.2.2.5	PDB Channel n Control Register (PDBCHnCR) .....	538
22.2.2.6	PDB Channel n Delay A & Delay B Registers (PDBCHnDLYA & PDBCHnDLYB) .....	539
22.2.3	Functional Description .....	539

## Chapter 23

### Voltage Reference (VREFV1)

23.1	Introduction .....	541
23.1.1	Overview .....	541
23.1.2	Features .....	542
23.1.3	Modes of Operation .....	542
23.1.4	External Signal Description .....	542
23.2	Memory Map and Register Definition .....	543
23.2.1	VREF Trim Register (VREFTRM) .....	543
23.2.2	VREF Status and Control Register (VREFSC) .....	544
23.3	Functional Description .....	544
23.3.1	Voltage Reference Disabled, VREFEN=0 .....	545
23.3.2	Voltage Reference Enabled, VREFEN=1 .....	545
23.3.2.1	Mode[1:0]=00 .....	545
23.3.2.2	Mode[1:0]=01 .....	545


23.3.2.3 Mode[1:0]=10 .....	545
23.3.2.4 Mode[1:0]=11 .....	545
23.4 Initialization Information .....	545

## Chapter 24 Development Support

24.1 Introduction .....	547
24.1.1 Features .....	547
24.2 Background Debug Controller (BDC) .....	547
24.2.1 BKGD Pin Description .....	548
24.2.2 Communication Details .....	549
24.2.3 BDC Commands .....	552
24.2.4 BDC Hardware Breakpoint .....	555
24.3 Register Definition .....	555
24.3.1 BDC Registers and Control Bits .....	555
24.3.1.1 BDC Status and Control Register (BDCSCR) .....	556
24.3.1.2 BDC Breakpoint Match Register (BDCBKPT) .....	557
24.3.2 System Background Debug Force Reset Register (SBD FR) .....	558

## Chapter 25 Break Point Unit (BKPT) (64K)

25.1 Introduction .....	559
25.1.1 Features .....	559
25.1.2 Modes of Operation .....	559
25.1.3 Block Diagram .....	559
25.2 Signal Description .....	560
25.3 Memory Map and Registers .....	561
25.3.1 Module Memory Map .....	561
25.3.2 Register Bit Summary- .....	562
25.3.3 Register Descriptions .....	563
25.3.3.1 Breakpoint Comparator A High Register (BKPTCAH) .....	563
25.3.3.2 Breakpoint Comparator A Low Register (BKPTCAL) .....	563
25.3.3.3 Breakpoint Comparator B High Register (BKPTCBH) .....	564
25.3.3.4 Breakpoint Comparator B Low Register (BKPTCBL) .....	564
25.3.3.5 Breakpoint Comparator C High Register (BKPTCCH) .....	565
25.3.3.6 Breakpoint Comparator C Low Register (BKPTCCL) .....	566
25.3.3.7 Breakpoint Comparator A Control Register (BKPTAC) .....	566
25.3.3.8 Breakpoint Comparator B Control Register (BKPTBC) .....	567
25.3.3.9 Breakpoint Comparator C Control Register (BKPTCC) .....	569
25.3.3.10 Breakpoint Status Register (BKPTS) .....	570
25.4 Functional Description .....	571
25.4.1 Comparator .....	571
25.4.1.1 RWA and RWAEN in Full Modes .....	571
25.4.2 Breakpoint Control Logic (BCL) .....	571
25.4.2.1 Breakpoint types .....	571



25.4.2.2 Full Mode .....	572
25.5 Resets .....	572
25.6 Interrupts .....	572

# Chapter 1

## Device Overview

The MC9S08GW64 and MC9S08GW32 are members of the low-cost, low-power, 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 MC9S08GW64 Series

Table 1-1 summarizes the feature sets available in the MC9S08GW64 series of MCUs.

#### NOTE

For FlexTimer module, MC9S08GW64 series only support TPM function which is the basic function of FlexTimer. The other function of FlexTimer is not available.

**Table 1-1. MC9S08GW64 Series Features by MCU and Package**

Feature	MC9S08GW64		MC9S08GW32	
Package	80-pin LQFP	64-pin LQFP	80-pin LQFP	64-pin LQFP
FLASH	65,536 Bytes		32,768 Bytes	
RAM	4,032 Bytes		2,048 Bytes	
ADC0 <sup>1</sup> Single-ended Channels	7-ch	7-ch	7-ch	7-ch
ADC0 Differential Channels <sup>2</sup>	1	0	1	0
ADC1 Single-ended Channels	7-ch	7-ch	7-ch	7-ch
ADC1 Differential Channels	1	1	1	1
BKPT	yes		yes	
ICS	yes		yes	
IIC	yes		yes	
IRQ	yes		yes	
IRTC	yes		yes	

Table 1-1. MC9S08GW64 Series Features by MCU and Package (continued)

Feature	MC9S08GW64		MC9S08GW32	
Package	80-pin LQFP	64-pin LQFP	80-pin LQFP	64-pin LQFP
KBI	8-ch		8-ch	
MTIM8	2		2	
MTIM16	yes		yes	
PCNT	yes		yes	
PCRC	yes		yes	
PDB	yes		yes	
PRACMP	3		3	
SCI	4		4	
SPI	3		3	
FTM	2-ch		2-ch	
LCD	8x36 4x40	8x24 4x28	8x36 4x40	8x24 4x28
VREFO	yes	yes	yes	yes
XOSC	2		2	
I/O pins <sup>3</sup>	57	45	57	45

<sup>1</sup> There are two 16-bit ADC modules, so two parallel conversions at two channels can be made simultaneously.

<sup>2</sup> Each differential channel consists of two pins (DADPx and DADMx).

<sup>3</sup> The I/O pins include one output-only pin.

## 1.2 MCU Block Diagram

The block diagram in [Figure 1-1](#) shows the structure of the MC9S08GW64 series MCUs.

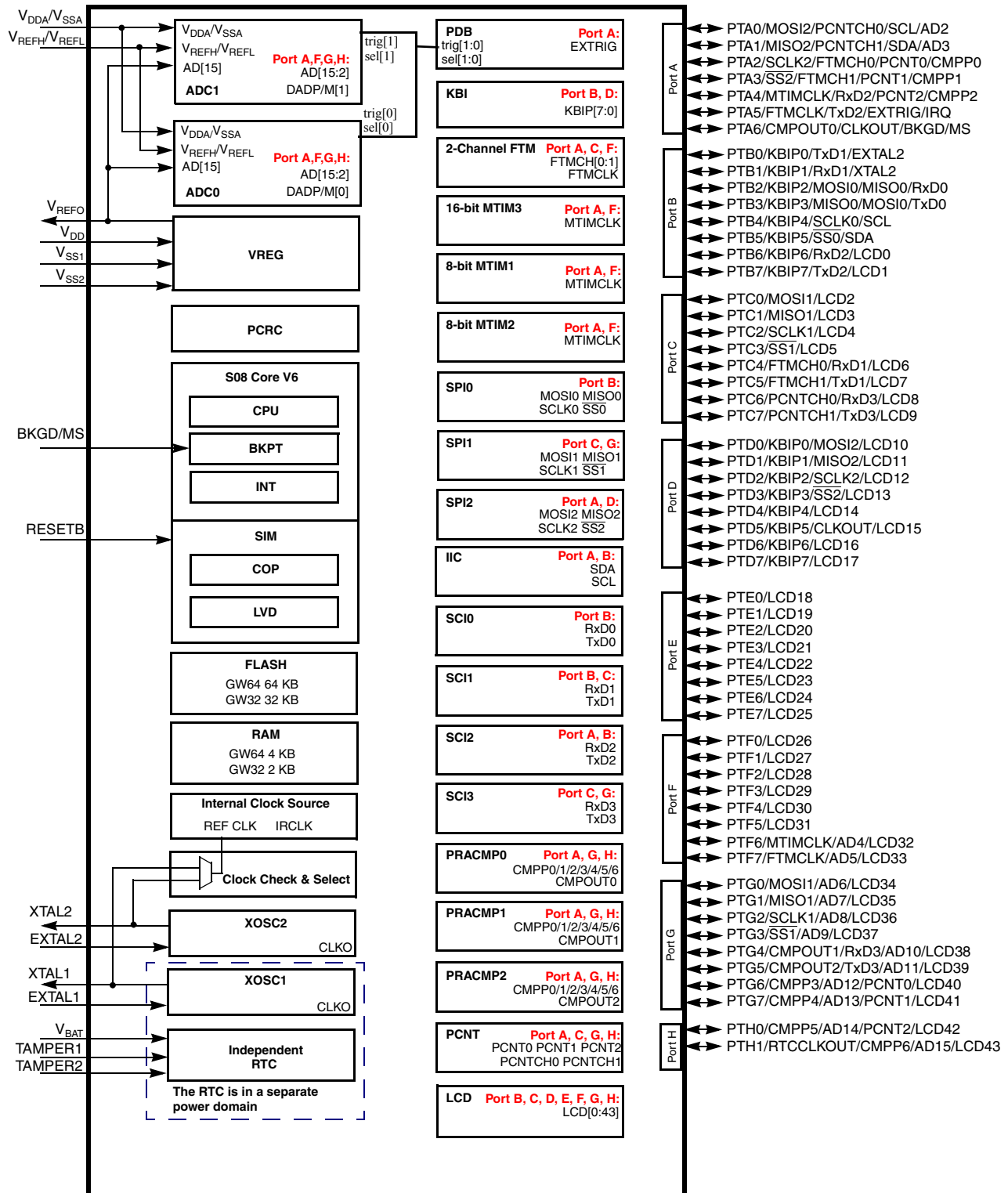


Figure 1-1. MC9S08GW64 Series Block Diagram

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

**Table 1-2. Module Versions**

Module	Version
Central Processing Unit (CPU)	6
Internal Clock Source (ICS)	4
Low Power Oscillator (XOSC)	1
Independent Real-Time Clock (IRTC)	2
Pulse Counter (PCNT)	1
Modulo Timer 8 bits (MTIM8)	1
Modulo Timer 16 bits (MTIM16)	1
FlexTimer Module (FTM)	3
Programmable Cyclic Redundancy Check (PCRC)	1
Programmable Delay Block (PDB)	1
Analog-to-Digital Converter (ADC16)	1
Programmable Reference Analog Comparator (PRACMP)	1
Liquid Crystal Display Driver (LCD)	1
Inter-Integrated Circuit (IIC)	5
Serial Communications Interface (SCI)	4
Serial Peripheral Interface (SPI)	4
Keyboard Interrupt (KBI)	2
Breakpoint Unit (BKPT)	1

### 1.3 System Clock Distribution

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

A unique feature of this product family is the independent real time clock (IRTC) and secondary 32 kHz crystal oscillator operating on an independent power domain. The IRTC can be powered by an independent coin battery, and continue to count even when the main CPU supply is down. The “Clock Check & Select” function allows the CPU to check whether the oscillator outputs are active prior to switching to one of the two as the reference for the ICS. The ICS supplies the clock sources:

- ICSOUT — This clock source is used as the CPU clock and the peripheral bus clock, BUSCLK. Control bits in the ICS control registers determine which of the three clock sources is connected:
  - Internal reference clock
  - External reference clock
  - Frequency-locked loop (FLL) output

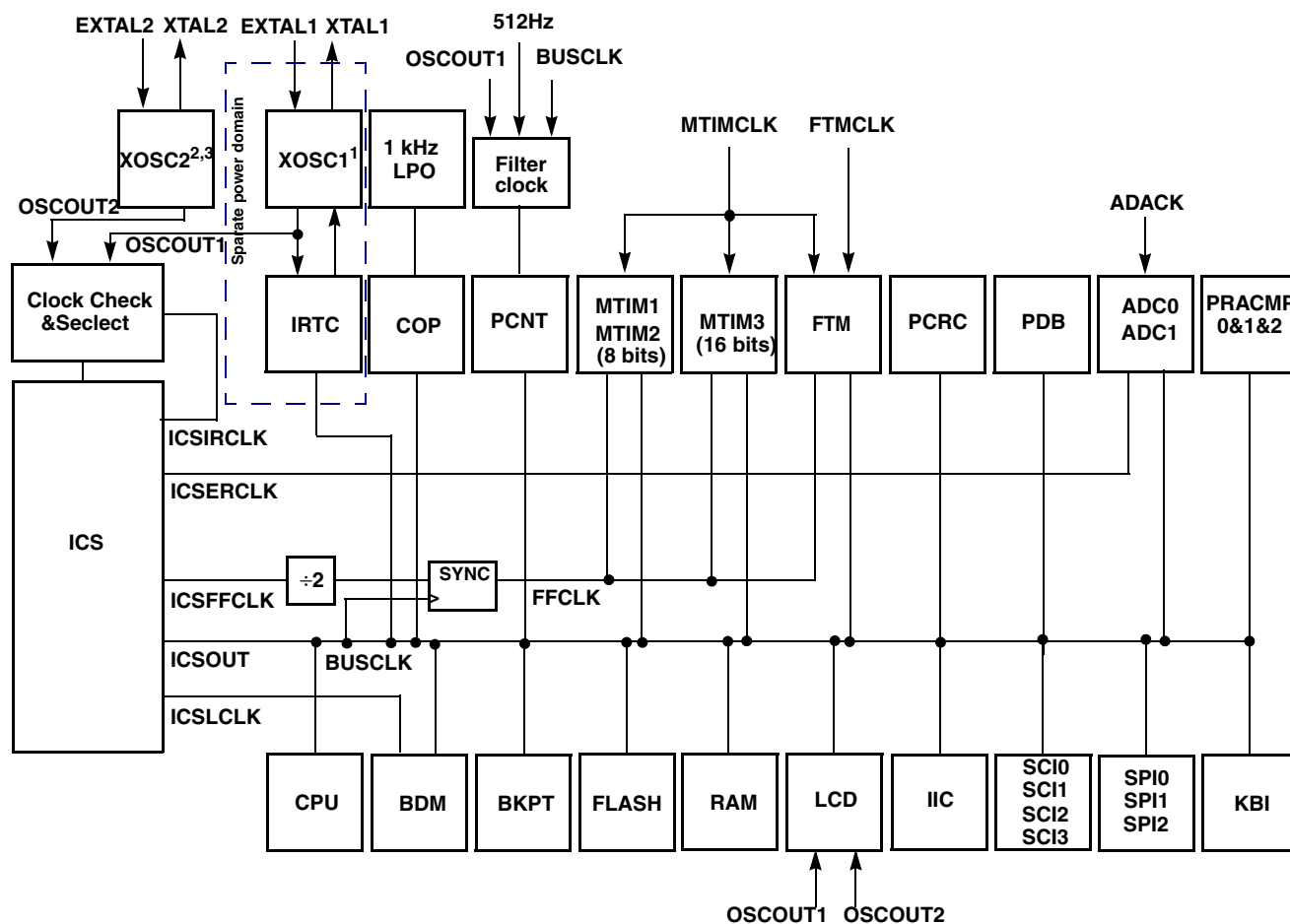
See Chapter 13, “Internal Clock Source (S08ICSV4),” for details on configuring the ICSOUT clock.

- ICSLCLK — This clock source is derived from the digitally-controlled oscillator, DCO, of the ICS when the ICS is configured to use the internal or external reference clock as the reference clock.



Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.

- **ICSERCLK** — This is the external reference clock and can be selected as the alternate clock for the ADC module. The “Optional External Reference Clock” section in [Chapter 13, “Internal Clock Source \(S08ICSV4\)”](#), explains the IC SERCLK in more detail. See [Chapter 17, “Analog-to-Digital Converter \(ADC16V1\)”](#), for more information regarding the use of IC SERCLK with these modules.
- **ICSIRCLK** — This is the internal reference clock and it is not used outside of the ICS module. [Chapter 13, “Internal Clock Source \(S08ICSV4\)”](#) explains the IC SIRCLK in more detail.
- **ICSFFCLK** — This generates the fixed-frequency clock (FFCLK) after being synchronized to the bus clock. It can be selected as the clock source for the MTIM and FTM modules. The frequency of the ICSFFCLK is determined by the settings of the ICS. See the “Fixed-Frequency Clock” section in [Chapter 13, “Internal Clock Source \(S08ICSV4\)”](#), for details.
- **LPOCLK** — This clock is generated from an internal low-power oscillator that is completely independent of the ICS module. The LPOCLK can be selected as the clock source to the COP. See [Section 5.4, “Computer Operating Properly \(COP\) Watchdog”](#) for details on using the LPOCLK with COP.
- **OSCOUT1** — This is a direct output of the XOSC1 and is used as the clock source for the independent real time clock (IRTC) and LCD modules. This signal can also be consumed by the ICS. See [Chapter 13, “Internal Clock Source \(S08ICSV4\)”](#), or [Chapter 16, “Independent Real Time Clock \(IRTCV2\)”](#), for details.
- **OSCOUT2** — This is a direct output of the XOSC2 and is normally consumed by ICS. It can also be consumed by the LCD module. See [Chapter 13, “Internal Clock Source \(S08ICSV4\)”](#), or [Chapter 11, “LCD Module \(S08LCDLPV1\)”](#), for details.
- **FTMCLK** — FTMCLK is the optional external clock source for the FTM module. The FTMCLK must be limited to 1/4th the frequency of the bus clock for synchronization. See [Section 18.2.1, “EXTCLK — FTM External Clock”](#), for more details.
- **MTIMCLK** — MTIMCLK is the optional external clock source for the MTIM and FTM modules. The MTIMCLK must be limited to 1/4th the frequency of the bus clock for synchronization. See the “External MTIM Clock Sources” section in [Chapter 15, “16-Bit Modulo Timer \(S08MTIM16V1\)”](#), [Chapter 14, “8-Bit Modulo Timer \(S08MTIMV1\)”](#) and [Chapter 18, “FlexTimer Module \(S08FTMV3\)”](#), for more details.
- **ADACK** — Each ADC module has an internally generated asynchronous clock which allows it to run in STOP mode. This signal is not available externally. See [Chapter 17, “Analog-to-Digital Converter \(ADC16V1\)”](#), for details.



- <sup>1</sup> OSC1 must work in 32 kHz mode and can never be disabled. The only bit REFS selected by the IRTC configuration register is used for OSC1 to bypass the oscillator when external clock is applied. The ICS module is valid only for OSC2 and cannot control OSC1.
- <sup>2</sup> In the external clock mode, ICS is fed from OSC2 after reset by default, though it can be fed from OSC1 or OSC2.
- <sup>3</sup> If ICS is working on the clock from external oscillator (XOSC2), then accidental write to PTBPF1 could hang the system because the clock to the entire system would be gated. Even external reset is not able to wake it up

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

## Chapter 2

# Pins and Connections

### 2.1 Introduction

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

### 2.2 Device Pin Assignment

This section shows the pin assignments for MC9S08GW64 series devices.

#### NOTE

$V_{BAT}$ , EXTAL1, XTAL1, TAMPER1, and TAMPER2 pins are in a different power domain.

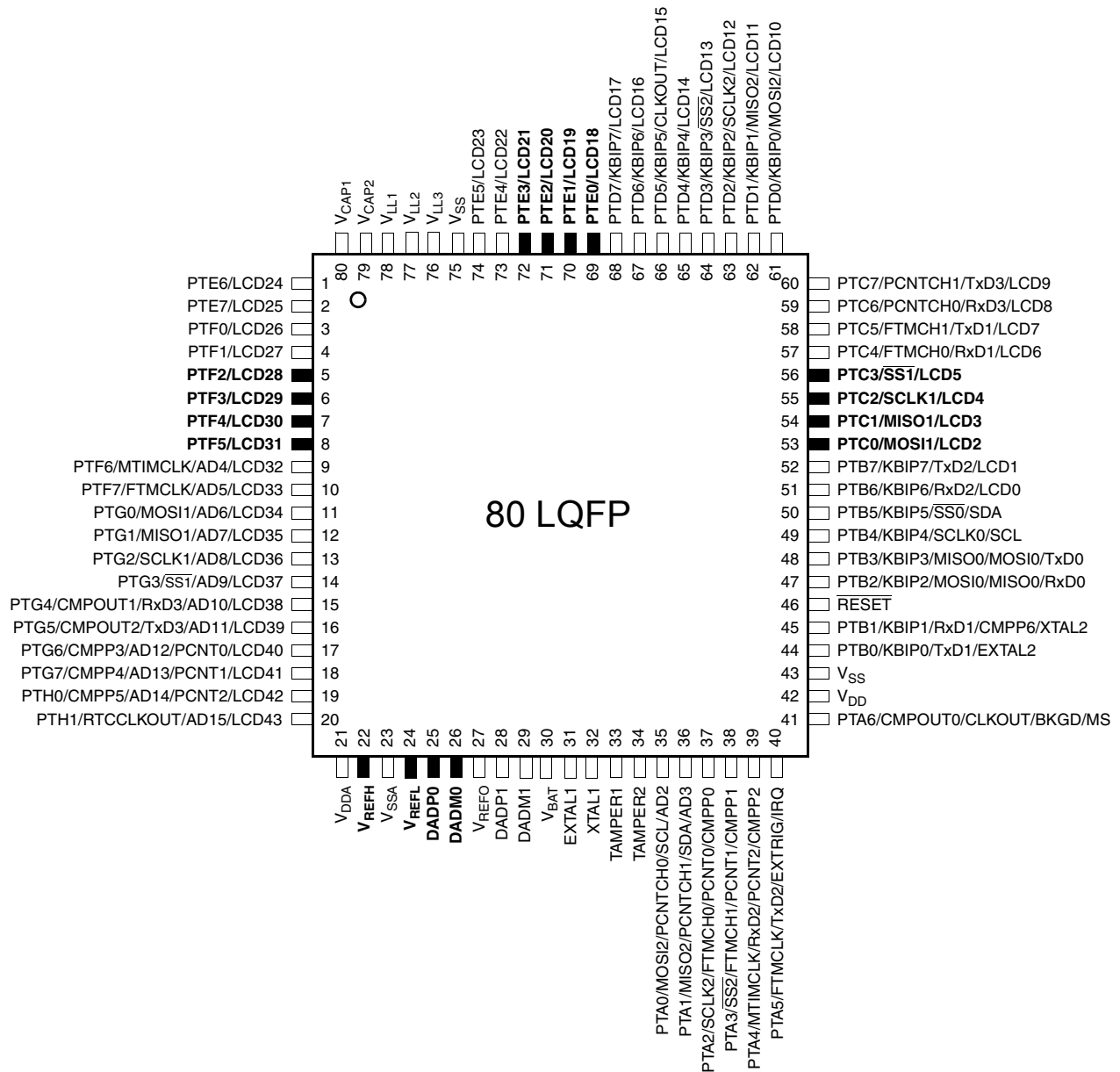


Figure 2-1. MC9S08GW64 Series in 80-Pin LQFP Package

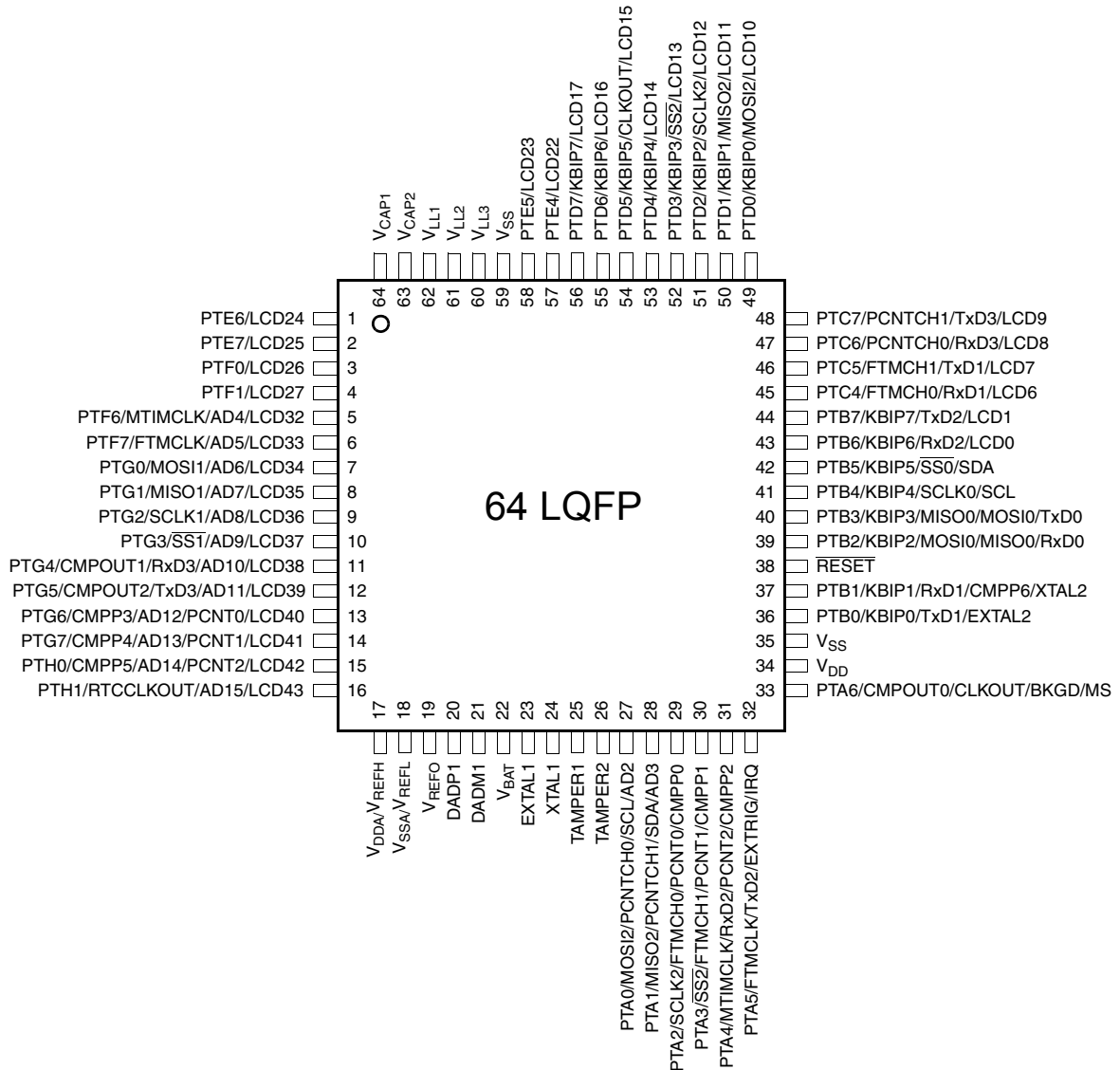


Figure 2-2. MC9S08GW64 Series in 64-Pin LQFP Package

## 2.3 Recommended System Connections

Figure 2-3 shows pin connections that are common to MC9S08GW64 series application systems.

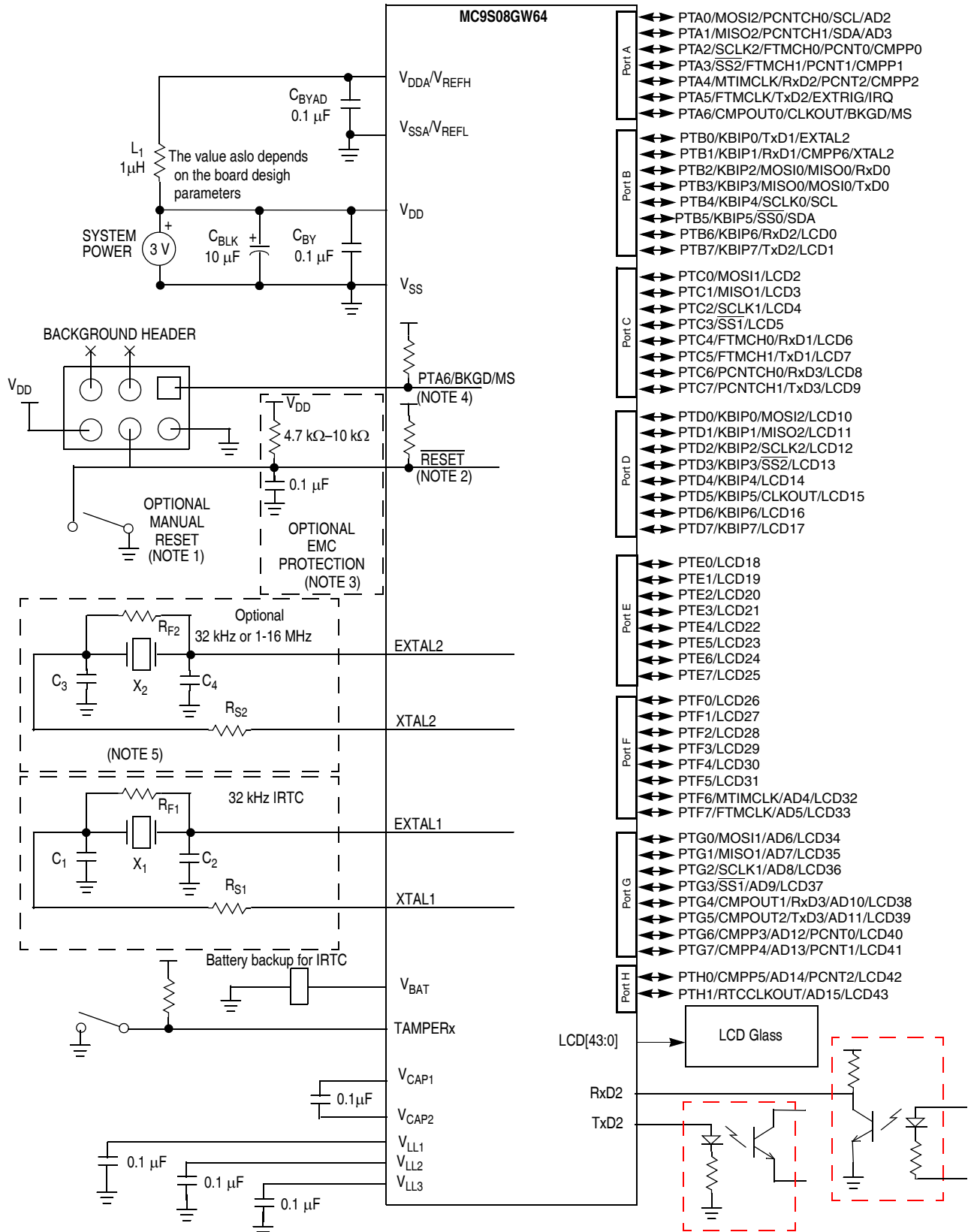


Figure 2-3. Basic System Connections

## NOTES:

1.  $\overline{\text{RESET}}$  pin can only be used to reset into user mode, you can not enter BDM using  $\overline{\text{RESET}}$  pin. BDM can be entered by holding MS low during POR or writing a 1 to BDFR in SBDFFR with MS low after issuing BDM command.
2.  $\overline{\text{RESET}}$  and IRQ features have optional internal pullup device.
3. RC filter on  $\overline{\text{RESET}}$  pin recommended for noisy environments.
4. PTA6 is an output only GPIO, when PTA6 is configured as BKGD, pin becomes bi-directional.
5. When using the XOSC2 module in low range and low power mode, the external components  $R_F$ ,  $R_S$ ,  $C_1$ , and  $C_2$  are not required.

### 2.3.1 Power

$V_{DD}$ ,  $V_{BAT}$  and  $V_{SS1}$ ,  $V_{SS2}$  are the primary power supply pins for the MCU.  $V_{DD}$ ,  $V_{SS1}$  and  $V_{SS2}$  supply power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides a regulated lower-voltage source for the CPU and other internal circuitry of the MCU.  $V_{BAT}$  supplies the power only to IRTC and OSC1.

LCD/GPIO can be powered differently, refer to [Chapter 6](#), “Parallel Input/Output Control” for additional information.

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

$V_{DDA}$  and  $V_{SSA}$  are the analog power supply pins for the MCU. This voltage source supplies power to the ADC and PRACMP modules. A 0.1  $\mu\text{F}$  ceramic bypass capacitor should be located as near to the MCU power pins as practical to suppress high-frequency noise. And a inductor (i.e. BRL2518T1R0M, also inductor value depends on the board design parameters) should be added between  $V_{DD}$  and  $V_{DDA}$  pins to restrain the potential glitch on PRACMP.

The  $V_{REFH}$  and  $V_{REFL}$  pins are the voltage reference high and voltage reference low inputs, respectively for the ADC module. A 0.1  $\mu\text{F}$  ceramic bypass capacitor must be located as near to the  $V_{REFx}$  pins as practical to suppress high-frequency noise.  $V_{REFH}$  and  $V_{REFL}$  are available only on the 80-pin LQFP package. On the 64 pin LQFP package they are connected to  $V_{DDA}$  and  $V_{SSA}$  respectively.

### 2.3.2 Oscillator

Oscillator1 is in the power domain of  $V_{BAT}$  while oscillator2 in the power domain of  $V_{DD}$ .

Immediately after reset, the MCU uses an internally-generated clock provided by the internal clock source (ICS) module.

The oscillator (XOSC) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Optionally, an external clock source can optionally be connected to the EXTAL input pin. The oscillator can be configured to run in stop2 or stop3 modes. For more information on the ICS, see [Chapter 13](#), “Internal Clock Source (S08ICSV4)”.

Refer to [Figure 2-3](#) for the following discussion.  $R_S$  (when used) and  $R_F$  must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors and some metal film resistors have too much inductance.  $C_1$  and  $C_2$  normally must 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.

$C_1$  and  $C_2$  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  $C_1$  and  $C_2$ . The crystal manufacturer typically specifies a load capacitance which is the series combination of  $C_1$  and  $C_2$  (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).

When using the oscillator in low-range and low-gain mode, the external components  $R_S$ ,  $R_F$ ,  $C_1$ , and  $C_2$  are not required.

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

The  $\overline{\text{RESET}}$  pin is a dedicated hardware reset pin. During stop2, the  $\overline{\text{RESET}}$  pin can be used to wake the device from that state. There is a direct analog connection from this pad to the power management controller wakeup pin.

The internal pullup on this pin is enabled upon any device reset.

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 non-POR reset is initiated (whether from an external signal or from an internal system), the  $\overline{\text{RESET}}$  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

- The voltage on the internally pulled up  $\overline{\text{RESET}}$  pin must be below  $V_{DD}$ . The internal gates connected to this pin are pulled to  $V_{DD}$ . If the  $\overline{\text{RESET}}$  pin is required to drive to a  $V_{DD}$  level, an external pullup must be used.
- In EMC-sensitive applications, an external RC filter is recommended on the  $\overline{\text{RESET}}$  pin if enabled. See [Figure 2-3](#) for an example.
- Reset should be asserted for minimum of 4 bus clock cycles.

### 2.3.4 IRQ

The IRQ pin is the input source for the IRQ interrupt and it is also the input for the BIH and BIL instructions. For IRQ, there is also a direct analog connection from pad to power management controller input. It acts as a non-maskable interrupt to the CPU in the stop2 mode. After reset, PTA5/FTMCLK/TxD2/EXTRIG/IRQ pin serves as GPIO.

### 2.3.5 Background/Mode Selection (BKGD/MS)

During a power-on reset (POR) or background debug force reset (see [Section 5.8.3, “System Background Debug Force Reset Register \(SBD FR\)”](#) for more information), the



PTA6/CMPOUT0/CLKOUT/BKGD/MS pin functions as a mode selection pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin (BKGDPE = 1), an internal pullup device is automatically enabled.

The background debug communication function is enabled when BKGDPE in SOPT1 is set. BKGDPE is set following any reset of the MCU and must be cleared to use the alternative PTA6/CMPOUT0/CLKOUT/BKGD/MS pin functions.

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

The BKGD/MS pin is used primarily with BDC communications, and features a custom protocol that uses 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/MS 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/MS pin.

The BKGD/MS selection pin can be reprogrammed to operate as PTA6 on this device. It must only be programmed for use as an output, as an external signal driving this pin during startup may cause the device to boot into debug mode.

### 2.3.6 V<sub>REF0</sub> Pins

The V<sub>REF0</sub> is the voltage reference output pins. A 0.1µF bypass capacitor must be used on these pins.

### 2.3.7 Dedicated ADC Pins

The DADP0, DADM0 and DADP1, DADM1 pins are dedicated differential ADC pins. DADP0 and DADM0 pins are not available on the 64-pin package.

### 2.3.8 AMR Pins Compatible with 5 V Interface

PTB3, PTB4 and PTB5 can be used as automatic meter reading (AMR) pins which are compatible with 5 V interface as shown in [Figure 2-5](#).

#### NOTE

As shown in [Figure 2-5](#), V<sub>DD</sub> is 3 V power supply while V<sub>DD1</sub> is 5 V power supply.

In SPI communication interface, PTB3 must be configured as MOSI0 if serving as a master and be configured as MISO0 if serving as a slaver.

In SCI communication interface, only PTB3 need to be involved in the communication and is served as TxD0.

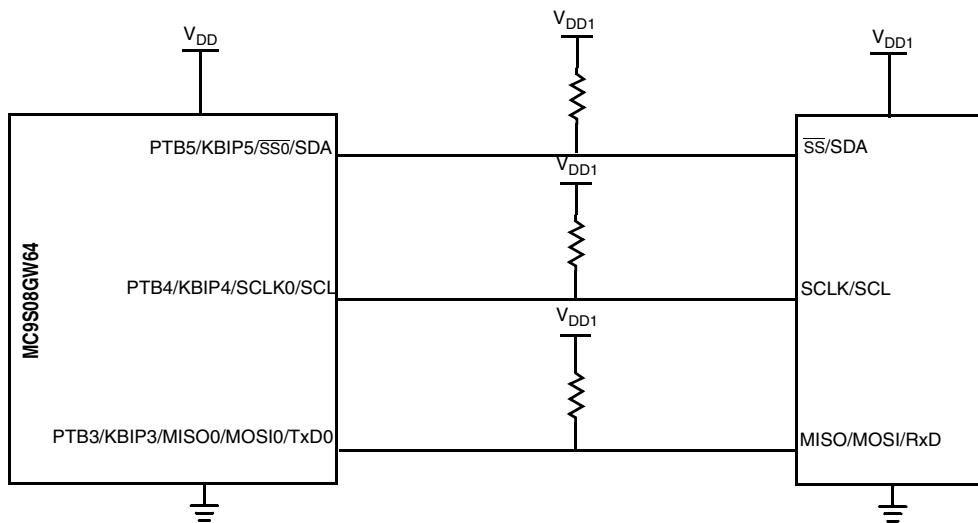


Figure 2-4. Connection with 5 V Interface

## 2.3.9 LCD Pins

### 2.3.9.1 LCD Power Pins

The  $V_{LL1}$ ,  $V_{LL2}$ ,  $V_{LL3}$ ,  $V_{cap1}$ , and  $V_{cap2}$  pins are dedicated to providing power to the LCD module. For detailed information about LCD pins, see [Chapter 11, “LCD Module \(S08LCDLPV1\)”](#).

### 2.3.9.2 LCD Driver Pins

The MC9S08GW64 series of MCUs contain 44 LCD driver pins on the 80-pin package, and 32 LCD driver pins on the 64-pin package. For both the 80-pin and 64-pin packages, all of the LCD driver pins are multiplexed with GPIO. If the LCD module is disabled, the LCD driver pins are high-impedance and the pins are configured as GPIO. The LCD module is automatically disabled after resets except for stop2 wakeup. For detailed information about LCD driver pins, see [Chapter 11, “LCD Module \(S08LCDLPV1\)”](#).

#### NOTE

MC9S08GW64 series support only up to 44 LCD pins (LCD[43:0]), please ignore the other LCD pins in the following sections of this chapter.

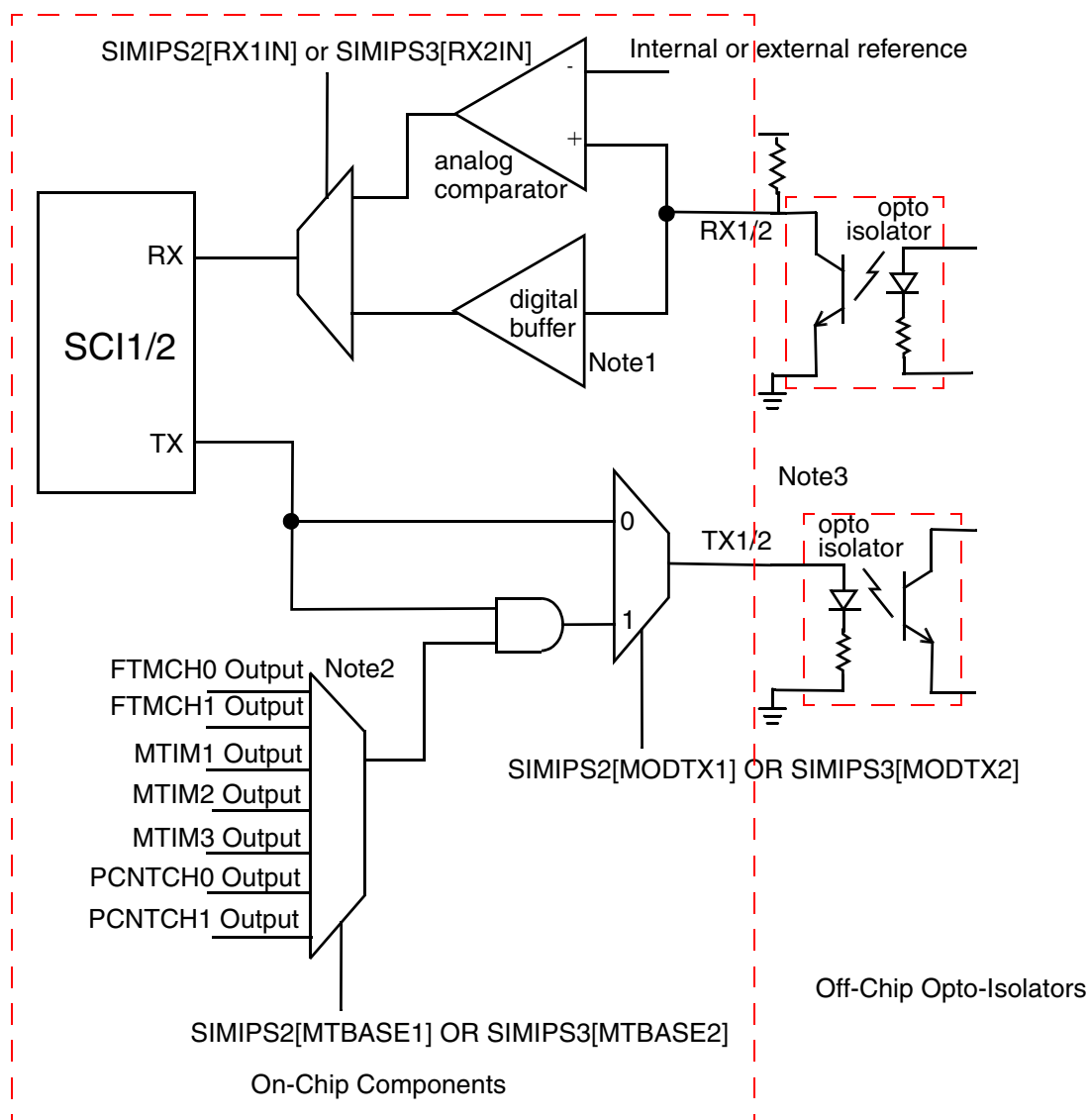
For GPIO muxed with LCD pins, full complimentary or open drain drive is controlled by the LCD controller. When LCD pins are configured as open drain GPIOs, then the internal pullup is not disabled in output mode and is controlled by the GPIO pull control register. This can cause some leakage from the pads if a pullup is enabled and a zero is being driven.

For some peripheral ports shared with LCD pad such as SPI, SCI and CLKOUT, in some cases, such as VLL3 powered by charge pump, it may not reach a high frequency i.e. 20MHz.

In the Charge Pump mode when the VLL3 is being generated internally. No digital I/O functionality should be present. If the VLL3 is being provided externally I/O can be used to drive respective loads.

### 2.3.10 Interfacing the SCIs to Off-Chip Circuits

The Rx pins of SCI1 and SCI2 can be fed directly from the digital I/O buffer, or signals be pre-conditioned using comparators 0 and 1 as shown in [Figure 2-5](#). Similarly, the Tx outputs of SCI1 and SCI2 can be modulated with the output of one of the timers before being passed off chip. SCI2 is twice the normal I/O drive capability on the Tx pin while SCI1 Tx pin is normal I/O drive capability.



### Figure 2-5. On-Chip Signal Conditioning Associated with SCI RX and TX Pins

Controls for the circuitry shown in Figure 2-5 are discussed in Section 5.8.6, “Internal Peripheral Select Register 2 (SIMIPS2),” and Section 5.8.7, “Internal Peripheral Select Register 3 (SIMIPS3).”

### 2.3.11 PCNT[2:0]

The PCNT input can be fed directly from PCNT pins or those signals can be pre-conditioned using analogue comparators PRACMP for interfacing to flow sensors that give analog output as shown in [Figure 2-6](#). This connection options provide the user more flexibility on using various of sensors.

Controls for the circuitry shown in Figure 2-6 are discussed in Section 5.8.5, “Internal Peripheral Select Register 1 (SIMIPS1).”

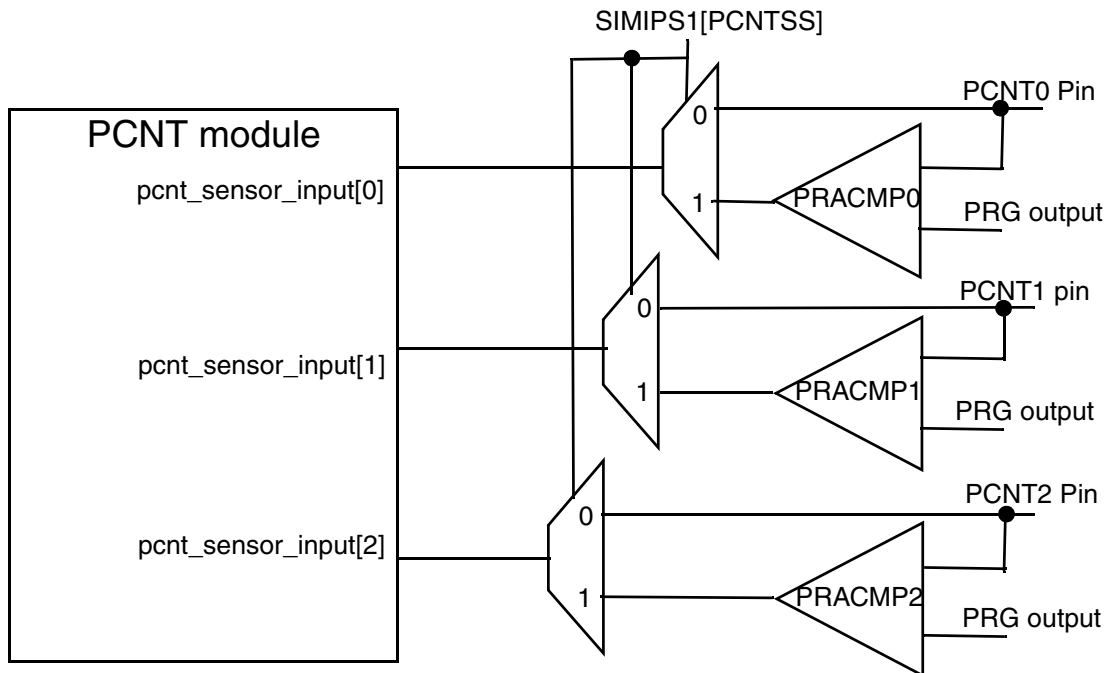


Figure 2-6. On-Chip Signal Conditioning Associated with PCNT[0:2] Pins

### 2.3.12 General-Purpose I/O (GPIO) and Peripheral Ports

The MC9S08GW64 series of MCUs support up to 57 general-purpose I/O pins, including one output-only pins, which are shared with on-chip peripheral functions (timers, serial I/O, LCD, ADC, PRACMP, etc.). The output-only pin (PTA6/CMPOUT0/CLKOUT/BKGD/MS) is bi-directional when configured as BKGD.

GPIO that is multiplexed with LCD pins can be configured to reference  $V_{DD}$  or  $V_{LL3}$ . See Section 6.1, “Introduction” for more information.

All the GPIO and peripheral ports are enabled and disabled via Pin Mux Controls logic only. Please refer to Section 6.7, “Pin Mux Controls,” for more information.

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

When a port pin is configured as a general-purpose input or when a peripheral uses the port pin as an input, software can enable a pullup device. Pad cells on these devices also include an optional low pass filter in the inputs. As well, this can be enabled via software control.

Immediately after reset, all of these pins (excluding the BKGD/MS pin) are configured as high impedance general purpose inputs with internal pullup devices disabled.

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 must either enable on-chip pullup devices or change the direction of unused or non-bonded pins to outputs so they do not float.

When using the 64-pin device, either enable on-chip pullup devices or change the direction of non-bonded pins to output so the pins do not float.

#### NOTE

Do not set LP bit in ICSC2 when XOSC2 is not used.

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

80	64	Port Pin	Default func	Alt 1	Alt 2	Alt3	Alt4
1	1	PTE6	PTE6		LCD24		
2	2	PTE7	PTE7		LCD25		
3	3	PTF0	PTF0	LCD26			
4	4	PTF1	PTF1	LCD27			
5		PTF2	PTF2	LCD28			
6		PTF3	PTF3	LCD29			
7		PTF4	PTF4	LCD30			
8		PTF5	PTF5	LCD31			
9	5	PTF6	PTF6	MTIMCLK	AD4	LCD32	
10	6	PTF7	PTF7	FTMCLK	AD5	LCD33	
11	7	PTG0	PTG0	MOSI1	AD6	LCD34	
12	8	PTG1	PTG1	MISO1	AD7	LCD35	
13	9	PTG2	PTG2	SCLK1	AD8	LCD36	
14	10	PTG3	PTG3	$\overline{SS1}$	AD9	LCD37	
15	11	PTG4	PTG4	CMPOUT1	RxD3	AD10	LCD38
16	12	PTG5	PTG5	CMPOUT2	TxD3	AD11	LCD39
17	13	PTG6	PTG6	CMPP3	AD12	PCNT0	LCD40
18	14	PTG7	PTG7	CMPP4	AD13	PCNT1	LCD41
19	15	PTH0	PTH0	CMPP5	AD14	PCNT2	LCD42

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

80	64	Port Pin	Default func	Alt 1	Alt 2	Alt3	Alt4
20	16	PTH1	PTH1	RTCCLKOUT	AD15	LCD43	
21	17	V <sub>DDA</sub>	V <sub>DDA</sub>				
22		V <sub>REFH</sub>	V <sub>REFH</sub>				
23	18	V <sub>SSA</sub>	V <sub>SSA</sub>				
24		V <sub>REFL</sub>	V <sub>REFL</sub>				
25		DADP0	DADP0				
26		DADM0	DADM0				
27	19	V <sub>REFO</sub>	V <sub>REFO</sub>				
28	20	DADP1	DADP1				
29	21	DADM1	DADM1				
30	22	V <sub>BAT</sub>	V <sub>BAT</sub>				
31	23	EXTAL1	EXTAL1				
32	24	XTAL1	XTAL1				
33	25	TAMPER1 <sup>1</sup>	TAMPER1				
34	26	TAMPER2	TAMPER2				
35	27	PTA0	PTA0	MOSI2	PCNTCH0	SCL	AD2
36	28	PTA1	PTA1	MISO2	PCNTCH1	SDA	AD3
37	29	PTA2	PTA2	SCLK2	FTMCH0	PCNT0	CMPP0
38	30	PTA3	PTA3	SS2	FTMCH1	PCNT1	CMPP1
39	31	PTA4	PTA4	MTIMCLK	RxD2	PCNT2	CMPP2
40	32	PTA5 <sup>2</sup>	PTA5	FTMCLK	TxD2	EXTRIG	IRQ
41	33	PTA6 <sup>3</sup>	BKGD/MS	CMPOUT0	CLKOUT	BKGD/MS	
42	34	V <sub>DD</sub>	V <sub>DD</sub>				
43	35	V <sub>SS</sub>	V <sub>SS</sub>				
44	36	PTB0	PTB0	KBIP0	TxD1	EXTAL2	
45	37	PTB1 <sup>1</sup>	PTB1	KBIP1	RxD1	CMPP6	XTAL2
46	38	RESET	RESET				
47	39	PTB2	PTB2	KBIP2	MOSI0	MISO0	RxD0
48	40	PTB3 <sup>4</sup>	PTB3	KBIP3	MISO0	MOSI0	TxD0
49	41	PTB4 <sup>3</sup>	PTB4	KBIP4	SCLK0	SCL	
50	42	PTB5 <sup>3</sup>	PTB5	KBIP5	SS0	SDA	
51	43	PTB6	PTB6	KBIP6	RxD2	LCD0	
52	44	PTB7	PTB7	KBIP7	TxD2	LCD1	
53		PTC0	PTC0	MOSI1	LCD2		
54		PTC1	PTC1	MISO1	LCD3		
55		PTC2	PTC2	SCLK1	LCD4		
56		PTC3	PTC3	SS1	LCD5		
57	45	PTC4	PTC4	FTMCH0	RxD1	LCD6	
58	46	PTC5	PTC5	FTMCH1	TxD1	LCD7	
59	47	PTC6	PTC6	PCNTCH0	RxD3	LCD8	

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

80	64	Port Pin	Default func	Alt 1	Alt 2	Alt3	Alt4
60	48	PTC7	PTC7	PCNTCH1	TxD3	LCD9	
61	49	PTD0	PTD0	KBIP0	MOSI2	LCD10	
62	50	PTD1	PTD1	KBIP1	MISO2	LCD11	
63	51	PTD2	PTD2	KBIP2	SCLK2	LCD12	
64	52	PTD3	PTD3	KBIP3	SS2	LCD13	
65	53	PTD4	PTD4	KBIP4	LCD14		
66	54	PTD5	PTD5	KBIP5	CLKOUT	LCD15	
67	55	PTD6	PTD6	KBIP6	LCD16		
68	56	PTD7	PTD7	KBIP7	LCD17		
69		PTE0	PTE0	LCD18			
70		PTE1	PTE1	LCD19			
71		PTE2	PTE2	LCD20			
72		PTE3	PTE3	LCD21			
73	57	PTE4	PTE4		LCD22		
74	58	PTE5	PTE5		LCD23		
75	59	V <sub>SS</sub>	V <sub>SS</sub>				
76	60	V <sub>LL3</sub>	V <sub>LL3</sub>				
77	61	V <sub>LL2</sub>	V <sub>LL2</sub>				
78	62	V <sub>LL1</sub>	V <sub>LL1</sub>				
79	63	V <sub>CAP2</sub>	V <sub>CAP2</sub>				
80	64	V <sub>CAP1</sub>	V <sub>CAP1</sub>				

<sup>1</sup> TAMPER0 pin is dedicatedly used for Battery Removal Tamper and not exposed on any pins.

<sup>2</sup> PTA5 is with double the normal I/O drive capability.

<sup>3</sup> PTA6 is an output-only pin when it is configured as GPIO.

<sup>4</sup> PTB2, PTB3 and PTB4 are compatible with 5 V devices with a pullup device.



## Chapter 3

# Modes of Operation

### 3.1 Introduction

The operating modes of the MC9S08GW64 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.
- Run mode — CPU clocks can run at full speed and the internal supply is fully regulated.
- LPRUN mode — CPU and peripheral clocks are restricted to 250 kHz maximum and the internal voltage regulator is in standby.
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained.
- LPWAIT mode — CPU shuts down to conserve power; peripheral clocks are restricted to 250 kHz maximum and the internal voltage regulator is in standby.
- 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, LCDPENx, LCDBPENx, and LCDWFX registers' content is retained, and I/O states held.

### 3.3 Run Mode

This is the normal operating mode for the MC9S08GW64 series. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE–0xFFFFF after reset.

#### 3.3.1 Low-Power Run Mode (LPRun)

In the low-power run mode, the on-chip voltage regulator is put into its standby state. This state uses the minimum power consumption necessary for CPU functionality. To reduce power consumption further, clear the appropriate bits in the SCGC registers to disable any unused peripheral clocks.

Before entering this mode, the following conditions must be met:

- FBELP is the selected clock mode for the ICS.
- The HGO bit in the ICSC2 register is clear.
- The bus frequency is less than 250 kHz.

- If enabled, configure the ADC to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements. The bandgap channel can be converted if the VREF module is enabled in low-power run mode.
- The LVDE or LVDSE bit in SPMSC1 register must be clear. LVD and LVW will automatically be disabled.
- Flash programming/erasing is not allowed.
- The VREF module must be enabled to supply the reference voltage for the PRACMP option to compare to internal reference voltage in LPRUN and LPWAIT.

Once these conditions are met, low power run mode can be entered by setting the LPR bit in the SPMSC2 register.

To re-enter standard run mode, clear the LPR bit. The LPRS bit in the SPMSC2 register is a read-only status bit that can be used to determine if the regulator is in full regulation mode or not. When LPRS is '0', the regulator is in full-regulation mode and the MCU can run at full speed in any clock mode.

### 3.3.1.1 Interrupts in Low-Power Run Mode

Low power run mode provides the option to return to full regulation if any interrupt occurs. This is done by setting the LPWUI bit in the SPMSC2 register. The ICS can then be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is clear, interrupts will be serviced in low power run mode.

If the LPWUI bit is set, LPR and LPRS bits will be cleared and interrupts will be serviced with the regulator in full regulation.

### 3.3.1.2 Resets in Low-Power Run Mode

Any reset will exit low power run mode, clear the LPR and LPRS bits, and return the device to normal run mode.

## 3.4 Active Background Mode

The active background mode functions are managed through the BDC in the HCS08 core. The BDC and the on-chip Breakpoint module (BKPT), provide the means for analyzing MCU operation during software development.

Active background mode is entered in any of six ways:

- When the BKGD/MS pin is low during POR
- When the BKGD/MS pin is low immediately after issuing a background debug force reset (see [Section 5.8.3, “System Background Debug Force Reset Register \(SBD FR\).”](#))
- 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 BKPT breakpoint

After entering active background mode, the CPU is held in a suspended state while it waits for serial background commands instead of executing instructions from the user application program.

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory-access-with-status commands
  - BDC register access commands
  - The BACKGROUND command
- Active background commands, which can be executed only 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 MC9S08GW64 series are shipped from the Freescale Semiconductor factory, the flash program memory is erased by default unless specifically noted. As a result, no program can be executed in run mode until the flash memory is initially programmed. The active background mode can also be used to erase and reprogram the flash memory after it has been previously programmed.

For additional information about the active background mode, refer to the [Chapter 24, “Development Support.”](#)

## 3.5 Wait Mode

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

### 3.5.1 Low-Power Wait Mode (LPWait)

Low power wait mode is entered by executing a WAIT instruction while the MCU is in low power run mode. In the low power wait mode, the on-chip voltage regulator remains in its standby state (as in the low

power run mode). This state uses the minimum power consumption necessary for most modules to maintain functionality. Power consumption is most reduced by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC register.

The same restrictions on the low power run mode apply to low power wait mode.

### 3.5.1.1 Interrupts in Low-Power Wait Mode

If the LPWUI bit is set when the WAIT instruction is executed, then the voltage regulator will return to full regulation when wait mode is exited. The ICS can be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is clear when the WAIT instruction is executed, an interrupt will return the device to low power run mode.

If the LPWUI bit is set when the WAIT instruction is executed, an interrupt will return the device to normal run mode with full regulation and the LPR and LPRS bits will be cleared.

### 3.5.1.2 Resets in Low-Power Wait Mode

Any reset will exit low power wait mode, clear the LPR and LPRS bits, and return the device to normal run mode.

## 3.6 Stop Modes

One of the two stop modes (stop2 or stop3) is entered upon execution of a STOP instruction when the STOPE bit in the system option 1 register (SOPT1) is set. In both stop modes, the bus and CPU clocks are halted. In stop3 the voltage regulator is in standby. In stop2 the voltage regulator is in partial powerdown. The ICS module can be configured to leave the reference clocks running. See [Chapter 13, “Internal Clock Source \(S08ICSV4\)”](#) 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**

Register	SOPT1	BDCSCR	SPMSC1		SPMSC2	Stop Mode
Bit name	STOPE	ENBDM <sup>1</sup>	LVDE	LVDSE	PPDC	
0	x	x	x	x	x	Stop modes disabled; illegal opcode reset if STOP instruction executed
1	1	x	x	x	x	Stop3 with BDM enabled <sup>2</sup>
1	0	Both bits must be 1	x	x	x	Stop3 with voltage regulator active
1	0	Either bit a 0	0	0	0	Stop3
1	0	Either bit a 0	1	1	1	Stop2

- <sup>1</sup> ENBDM is located in the BDCSCR which is accessible only through BDC commands, see the [Chapter 24, “Development Support.”](#)
- <sup>2</sup> When in Stop3 mode with BDM enabled, The  $SI_{DD}$  will be near  $RI_{DD}$  levels because internal clocks are enabled.

### 3.6.1 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, optionally the low power oscillator, voltage reference and the LCD module. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2. LCD driver pins continue to drive the signals necessary to display the LCD data.

Exit from stop2 is performed by asserting the wakeup pin ( $\overline{RESET}$ , PTA5/IRQ, or IRTC interrupt) on the MCU.

#### NOTE

When PTA5/IRQ is used as an active low wakeup source, it must be configured as an input prior to executing a STOP instruction to avoid an immediate exit from stop2. PTA5/IRQ can be disabled as a wakeup if it is configured as a high driven output. For lowest power consumption in stop2, this pin must not be left open if configured as input (enable the internal pullup, tie an external pullup device, or set pin as output).

In addition, the Independent Real Time Clock module (IRTC) can wake the MCU from stop2, if enabled.

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

- All module control and status registers are reset, except for SPMSC1–SPMSC3, LCDPEN<sub>x</sub>, LCDBPEN<sub>x</sub> and LCDWFR<sub>x</sub>.
- The LVD reset function is enabled and the MCU remains in the reset state if  $V_{DD}$  is below the LVD trip point
- The CPU takes the reset vector

In addition to the above, upon waking 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.

If using the low power oscillator during stop2, the user must reconfigure the ICSC2 register which contains oscillator control bits before PPDACK is written.

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 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.

If enabled, LCD functionality continues in stop2 mode and upon stop2 recovery the LCD control registers (LDC0, LDC1, LCDSUPPLY, LCDRVC, LCDBCTL, and LCDS) must be re-initialized before writing the PPDACK.

### 3.6.2 Stop3 Mode

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

Stop3 can be exited by asserting  $\overline{\text{RESET}}$ , or by an interrupt from one of the following sources: the IRTC, LVD, LVW, ADC, ACMP, IRQ, SCI, LCD, KBI, IIC or PCNT.

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

### 3.6.3 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. This register is described in [Chapter 24, “Development Support.”](#) If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If the user attempts to enter stop2 with ENBDM set, the MCU will instead enter stop3.

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

### 3.6.4 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (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.

### 3.6.5 Stop Modes in Low Power Run Mode

Stop2 mode cannot be entered from low power run mode. If the PPDC bit is set, then the LPR bit cannot be set. Likewise, if the LPR bit is set, the PPDC bit cannot be set.

Stop3 mode can be entered from low power run mode by executing the STOP instruction while in low power run. Exiting stop3 with a reset will put the device back into normal run mode. If LPWUI is clear, interrupts will exit stop3 mode, return the device to low power run mode, and then service the interrupt. If

LPWUI is set, interrupts will exit stop3 mode, put the device into normal run mode, clear LPR and LPRS bits, and then service the interrupt.

### 3.7 Mode Selection

Several control signals are used to determine the current operating mode of the device. Table 3-2 shows the conditions for each of the device's operating modes.

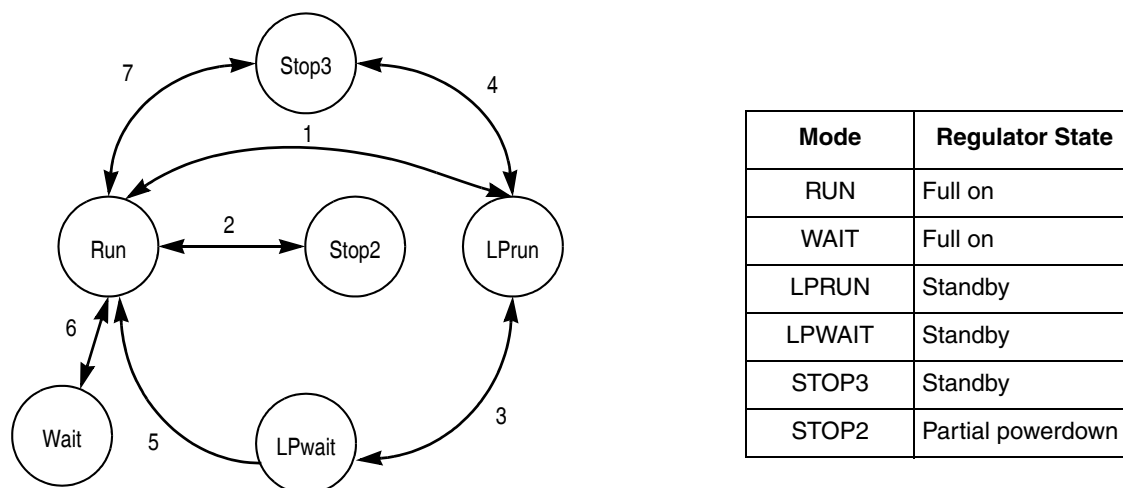
**Table 3-2. Power Mode Selections**

Mode of Operation	BDCSCR BDM	SPMSC1 PMC		SPMSC2 PMC		CPU & Periph CLKs	Affects on Sub-System	
	ENBDM <sup>1</sup>	LVDE	LVDSE	LPR	PPDC		BDM Clock	Voltage Regulator
RUN mode	0	x	x	0	x	on. ICS in any mode.	off	on
		1	1	1			on	
	1	x	x	x			on	
LPRUN mode	0	0	x	1	0	Low freq required. ICS in FBELP mode only.	off	standby
		1	0				on	
WAIT mode — (Assumes WAIT instruction executed.)	0	x	x	0	x	CPU clock is off; peripheral clocks on. ICS state same as RUN mode.	off	on
		1	1	1			on	
	1	x	x	x			on	
LPWAIT mode — (Assumes WAIT instruction executed.)	0	0	x	1	0	CPU clock is off; peripheral clocks at low speed. ICS in FBELP mode.	off	standby
		1	0				on	
Stop3 — (Assumes STOPE bit is set and STOP instruction executed.) Note that stop3 is used in place of stop2 if the BDM or LVD is enabled.	0	0	x	x	0	ICS in STOP. IC SERCLK, ICSIRCLK, and OSCOUT optionally on <sup>2</sup>	off	standby
	0	1	0	x	0		off	
	0	1	1	x	x		off	
	1	x	x	x	x	ICSLCLK still active.	on	on - stop currents will be increased
Stop2 — (Assumes STOPE bit is set and STOP instruction executed.) If BDM or LVD is enabled, stop3 will be invoked rather than stop2.	0	0	x	0	1	OSCOUT optionally on <sup>2,3</sup>	off	partial powerdown
		1	0					

<sup>1</sup> ENBDM is located in the BDC status and control register (BDCSCR) which is write accessible only through BDC commands.

<sup>2</sup> Configured within the ICS module based on the settings of IREFSTEN, EFRESTEN, IRCLKEN and ERCLKEN.

<sup>3</sup> In stop2, CPU, flash, ICS and all peripheral modules are powered down except for LCD.



**Figure 3-1. Allowable Power Mode Transitions for the MC9S08GW64 Series**

Figure 3-1 illustrates mode state transitions allowed between the legal states shown in Table 3-1.  $\overline{\text{RESET}}$  or PTA5/IRQ must be asserted low or a IRTC interrupt must occur to exit stop2. Interrupts suffice for the other stop and wait modes.

Table 3-3 defines triggers for the various state transitions shown in Figure 3-1.

**Table 3-3. Triggers for Transitions Shown in Figure 3-1.**

Transition #	From	To	Trigger
1	RUN	LPRUN	Configure settings shown in Table 3-1, switch LPR =1 last
	LPRUN	RUN	Clear LPR
			Interrupt when LPWUI=1
2	RUN	STOP2	Pre-configure settings shown in Table 3-1, issue STOP instruction
	STOP2	RUN	Assert zero on $\overline{\text{RESET}}$ <sup>1</sup> or PTA5/IRQ, assert an IRTC interrupt, or reload environment from RAM
3	LPRUN	LPWAIT	WAIT instruction
	LPWAIT	LPRUN	Interrupt when LPWUI = 0
4	LPRUN	STOP3	STOP instruction
	STOP3	LPRUN	Interrupt when LPWUI = 0
5	LPWAIT	RUN	Interrupt when LPWUI = 1
	RUN	LPWAIT	NOT SUPPORTED



Table 3-3. Triggers for Transitions Shown in Figure 3-1. (continued)

Transition #	From	To	Trigger
6	RUN	WAIT	WAIT instruction
	WAIT	RUN	Interrupt or reset
7	STOP3	RUN	Interrupt (if LPR = 0, or LPR = 1 and LPWUI = 1) or reset
	RUN	STOP3	STOP instruction

<sup>1</sup> An analog connection from this pin to the on-chip regulator wakes the regulator, which then initiates a power-on-reset sequence.

### 3.7.1 On-Chip Peripheral Modules in Stop and Low Power 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 3.6.1, “Stop2 Mode,”](#) and [Section 3.6.2, “Stop3 Mode,”](#) for specific information on system behavior in stop modes.

When the MCU enters LPWait or LPRun modes, system clocks to the internal peripheral modules continue based on the settings of the clock gating control registers (SCGC).

Table 3-4. Stop and Low Power Mode Behavior

Peripheral	Mode			
	Stop2	Stop3	LPWait	LPRun
CPU	Off	Standby	Standby	On
RAM	Standby	Standby	Standby	On
FLASH	Off	Standby	Standby	On
Port I/O Registers	Off	Standby	Standby	On
ADC16	Off	Optionally On <sup>1</sup>	Optionally On <sup>1</sup>	Optionally On <sup>1</sup>
PRACMPx	Off	Optionally On <sup>2</sup>	Optionally On	Optionally On
BDM	Off <sup>3</sup>	Optionally On	Off <sup>4</sup>	Off <sup>4</sup>
COP	Off	Off	Optionally On	Optionally On
ICS	Off	Optionally On <sup>5</sup>	On <sup>6</sup>	On <sup>6</sup>
IIC	Off	Optionally On	Optionally On	Optionally On
IRQ	Wake Up	Optionally On	Optionally On	Optionally On
KBI	Off	Optionally On	Optionally On	Optionally On
LVD/LVW	Off <sup>7</sup>	Optionally On	Off <sup>8</sup>	Off <sup>8</sup>
MTIMx	Off	Standby	Optionally On	Optionally On
LCD	Optionally On	Optionally On	Optionally On <sup>9</sup>	Optionally On <sup>9</sup>
SCIx	Off	Standby	Optionally On	Optionally On
SPI	Off	Standby	Optionally On	Optionally On
FTM	Off	Standby	Optionally On	Optionally On
PDB	Off	Standby	Optionally On	Optionally On

Table 3-4. Stop and Low Power Mode Behavior (continued)

Peripheral	Mode			
	Stop2	Stop3	LPWait	LPRun
PCRC	Off	Standby	Optionally On	Optionally On
PCNT	Off	Optionally On	Optionally On	Optionally On
VREF	Off	Optionally On	Optionally On	Optionally On
Voltage Regulator	Partial Powerdown	Optionally On <sup>10</sup>	Standby	Standby
XOSC2	Optionally On <sup>11</sup>	Optionally On <sup>11</sup>	Optionally On	Optionally On
XOSC1	On <sup>12</sup>	On <sup>12</sup>	On	On
IRTC	On <sup>12</sup>	On <sup>12</sup>	On	On
I/O Pins	States Held	Peripheral Control	Peripheral Control	On

<sup>1</sup> Requires the asynchronous ADC clock. For stop3, The VREF must be enabled to run in stop if converting the bandgap channel.

<sup>2</sup> VREF must be enabled to run in stop if using the bandgap as a reference.

<sup>3</sup> If ENBDM is set when entering stop2, the MCU will actually enter stop3.

<sup>4</sup> If ENBDM is set when entering LPRun or LPWait, the MCU will actually stay in run mode or enter wait mode, respectively.

<sup>5</sup> IRCLKEN and IREFSTEN set in ICSC1, else in standby.

<sup>6</sup> ICS must be configured for FBELP, bus frequency limited to 250 kHz, the external crystal limited to 2MHz (Bus Divider is 8) in LPRUN or LPWAIT.

<sup>7</sup> If LVDSE is set when entering stop2, the MCU will actually enter stop3.

<sup>8</sup> If LVDSE is set when entering LPRun or LPWait, the MCU will actually enter run or wait mode, respectively.

<sup>9</sup> Requires OSCOUT1 or OSCOUT2.

<sup>10</sup> Requires the LVD to be enabled, else in standby. See [Section 3.6.4, "LVD Enabled in Stop Mode"](#).

<sup>11</sup> ERCLKEN and EREFSTEN set in ICSC2, else in standby.

<sup>12</sup> When XOSCI and RTC are powered by battery in Stop2 and Stop3 modes, recommend to disable LVD.

# Chapter 4

## Memory

### 4.1 Introduction

This chapter describes the MC9S08GW64 series on-chip memory. It details the memory map, various vector and bit assignments, registers and control bits, and other RAM and flash features.

### 4.2 MC9S08GW64 Series Memory Map

As shown in [Figure 4-3](#), on-chip memory in the MC9S08GW64 series of MCUs consists of RAM, flash program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into several groups:

**Table 4-1. System Memory Map**

Start	End	Size	Name	Comments
0x0000	0x00BF	192	Direct Page Registers	
0x00C0	0x107F	4032	RAM	GW32 0x00C0 to 0x08BF 2048 RAM <sup>1</sup>
0x1080	0x10BF	64	LCD RAM	
0x10E0	0x17FF		Unimplementaed	
0x1800	0x195F	352	High Page Registers	
0xFFB0	0xFFBF	16	Nonvolatile Registers	

<sup>1</sup> In GW32, if RAM is accessed beyond 2K boundary then the functionality can not be guaranteed and unexpected behavior will occur.

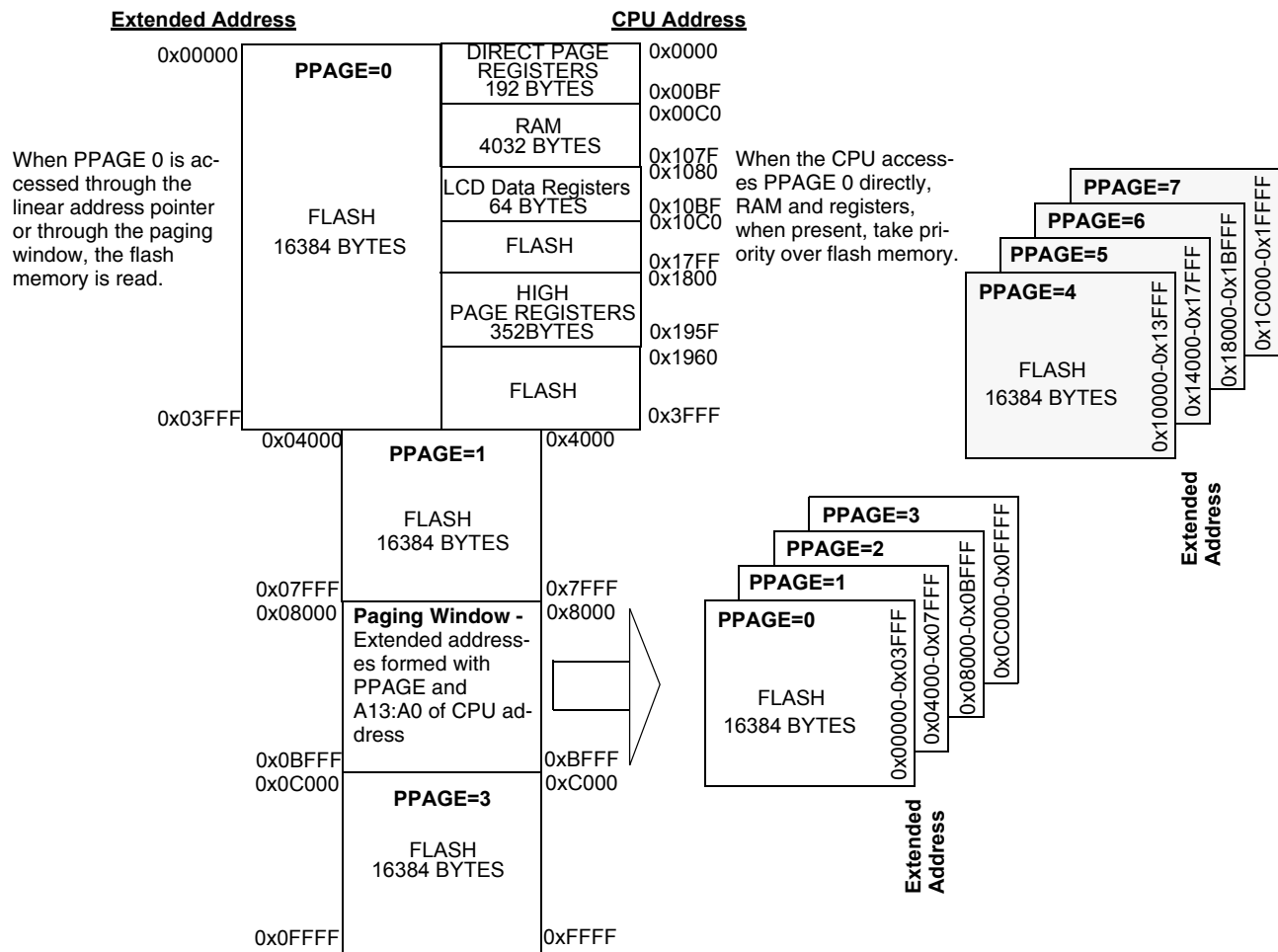


Figure 4-1. MC9S08GW64 Memory Map

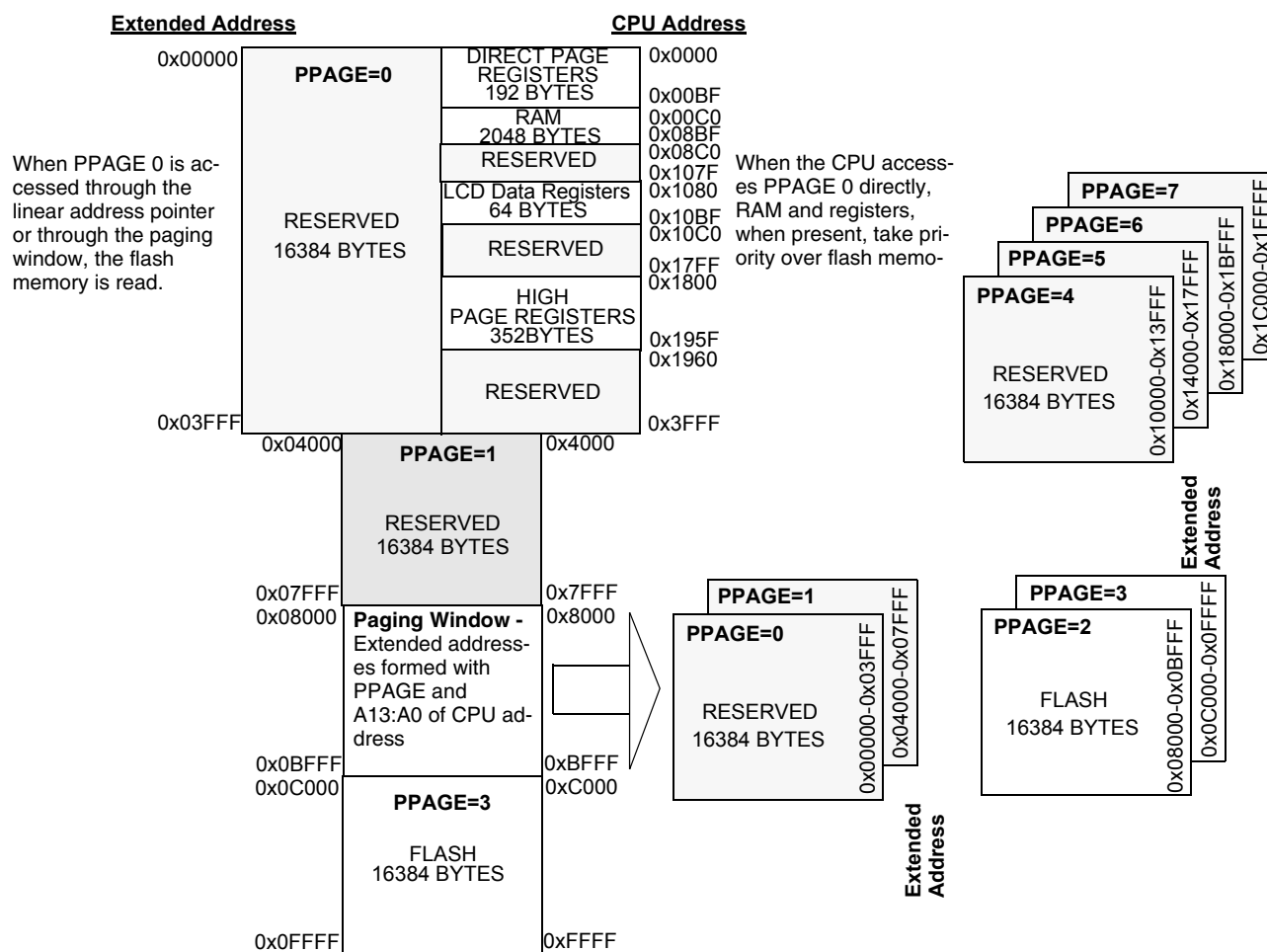


Figure 4-2. MC9S08GW32 Memory Map

### 4.3 Reset and Interrupt Vector Assignments

Table 4-2 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 MC9S08GW64 series.

Table 4-2. Reset and Interrupt Vectors

Address (High/Low)	Vector	Vector Name
0xFF80:FF81 ↕ 0xFF94:FF95	Unused Vector Space (available for user program)	
0xFF96:FF97	PDB	V <sub>PDB</sub>
0xFF98:FF99	PDB Sequence Error	V <sub>PDBERR</sub>
0xFF9A:FF9B	LCD	V <sub>LCD</sub>
0xFF9C:FF9D	IRTC	V <sub>IRTC</sub>

Table 4-2. Reset and Interrupt Vectors (continued)

Address (High/Low)	Vector	Vector Name
0xFF9E:FF9F	KBI	V <sub>KBI</sub>
0xFFC0:FFC1	PCNT	V <sub>PCNT</sub>
0xFFC2:FFC3	PRACMP2	V <sub>PRCMP2</sub>
0xFFC4:FFC5	PRACMP1	V <sub>PRCMP1</sub>
0xFFC6:FFC7	PRACMP0	V <sub>PRCMP0</sub>
0xFFC8:FFC9	ADC1	V <sub>ADC1</sub>
0xFFCA:FFCB	ADC0	V <sub>ADC0</sub>
0xFFCC:FFCD	MTIM2	V <sub>MTIM2</sub>
0xFFCE:FFCF	MTIM1	V <sub>MTIM1</sub>
0xFFD0:FFD1	MTIM0	V <sub>MTIM0</sub>
0xFFD2:FFD3	FTM Overflow	V <sub>FTMOF</sub>
0xFFD4:FFD5	FTM Channel 0	V <sub>FTMCH0</sub>
0xFFD6:FFD7	FTM Channel 1	V <sub>FTMCH1</sub>
0xFFD8:FFD9	IIC	V <sub>IIC</sub>
0xFFDA:FFDB	SPI2	V <sub>SPI2</sub>
0xFFDC:FFDD	SPI1	V <sub>SPI1</sub>
0xFFDE:FFDF	SPI0	V <sub>SPI0</sub>
0xFFE0:FFE1	SCI3 Transmit	V <sub>SCI3TX</sub>
0xFFE2:FFE3	SCI3 Receive	V <sub>SCI3RX</sub>
0xFFE4:FFE5	SCI3 Error	V <sub>SCI3ERR</sub>
0xFFE6:FFE7	SCI2 Transmit	V <sub>SCI2TX</sub>
0xFFE8:FFE9	SCI2 Receive	V <sub>SCI2RX</sub>
0xFFEA:FFEB	SCI2 Error	V <sub>SCI2ERR</sub>
0xFFEC:FFED	SCI1 Transmit	V <sub>SCI1TX</sub>
0xFFEE:FFEF	SCI1 Receive	V <sub>SCI1RX</sub>
0xFFF0:FFF1	SCI1 Error	V <sub>SCI1ERR</sub>
0xFFF2:FFF3	SCI0 Transmit	V <sub>SCI0TX</sub>
0xFFF4:FFF5	SCI0 Receive	V <sub>SCI0RX</sub>
0xFFF6:FFF7	SCI0 Error	V <sub>SCI0ERR</sub>
0xFFF8:FFF9	LVD	V <sub>LVD</sub>
0xFFFA:FFFB	IRQ	V <sub>IRQ</sub>
0xFFFC:FFFD	SWI	V <sub>SWI</sub>
0xFFFE:FFFF	RESET	V <sub>RESET</sub>

## 4.4 Register Addresses and Bit Assignments

The registers in the MC9S08GW64 series are divided into these groups:

- Direct-page registers are the ones located in the first 96 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- LCD data registers, LCDPENx, LCDBPENx, LCDWFX are located at the end of the RAM module.
- 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 a block of 16 locations in flash memory at 0xFFB0–0xFFBF. Nonvolatile register locations include:
  - 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 4-3](#) is a summary of all user-accessible direct-page registers and control bits.

The direct-page registers in [Table 4-3](#) 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 4-5](#) and [Table 4-6](#), the whole address in column one is shown in bold. In [Table 4-3](#), [Table 4-5](#), and [Table 4-6](#), the register names in column two are shown in bold to set them apart from the bit names to the right. Cells that are not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s. When writing to these bits, write a 0 unless otherwise specified.

Table 4-3. Direct-Page Register Summary (Sheet 1 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	0	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	0	0	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	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x0009	PTEDD	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x000A	PTFD	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
0x000B	PTFDD	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
0x000C	PTGD	PTGD7	PTGD6	PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
0x000D	PTGDD	PTGDD7	PTGDD6	PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
0x000E	PTHD	—	—	—	—	—	—	PTHD1	PTHD0
0x000F	PTHDD	—	—	—	—	—	—	PTHDD1	PTHDD0
0x0010	MTIM1SC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x0011	MTIM1CLK	0	0	CLKS		PS			
0x0012	MTIM1CNT	COUNT							
0x0013	MTIM1MOD	MOD							
0x0014	MTIM2SC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x0015	MTIM2CLK	0	0	CLKS		PS			
0x0016	MTIM2CNT	COUNT							
0x0017	MTIM2MOD	MOD							
0x0018	MTIM3SC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x0019	MTIM3CLK	0	0	CLKS		PS			
0x001A	MTIM3CNTH	CNTH							
0x001B	MTIM3CNTL	CNTL							
0x001C	MTIM3MODH	MODH							
0x001D	MTIM3MODL	MODL							
0x001E- 0x001F	Reserved	—	—	—	—	—	—	—	—
0x0020	PDBSC	PRESCALER			0	0	0	0	0
0x0021		IF	CONT	SWTRIG	TRIGSEL			IE	EN
0x0022	PDBMOD	MOD[15:8]							
0x0023		MOD[7:0]							
0x0024	PDBCNT	COUNT[15:8]							
0x0025		COUNT[7:0]							
0x0026	PDBIDLY	IDELAY[15:8]							
0x0027		IDELAY[7:0]							



Table 4-3. Direct-Page Register Summary (Sheet 2 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	PDBCH1CR	ERRA	ERRB	0	0	0	0	0	0
0x0029		0	0	AOS		BOS		ENA	ENB
0x002A		PDBCH1DLYA							
0x002B		DELAY[15:8]							
		DELAY[7:0]							
0x002C	PDBCH1DLYB	DELAY[15:8]							
0x002D		DELAY[7:0]							
0x002E		Reserved	RESERVED						
0x002F		RESERVED							
0x0030	PDBCH2CR	ERRA	ERRB	0	0	0	0	0	0
0x0031		0	0	AOS		BOS		ENA	ENB
0x0032		PDBCH2DLYA							
0x0033		DELAY[15:8]							
		DELAY[7:0]							
0x0034	PDBCH2DLYB	DELAY[15:8]							
0x0035		DELAY[7:0]							
0x0036- 0x0037		Reserved	—	—	—	—	—	—	—
0x0038	PPAGE	0	0	0	0	0	XA16	XA15	XA14
0x0039	LAP2	0	0	0	0	0	0	0	LA16
0x003A	LAP1	LA15	LA14	LA13	LA12	LA11	LA10	LA9	LA8
0x003B	LAP0	LA7	LA6	LA5	LA4	LA3	LA2	LA1	LA0
0x003C	LWP	D7	D6	D5	D4	D3	D2	D1	D0
0x003D	LBP	D7	D6	D5	D4	D3	D2	D1	D0
0x003E	LB	D7	D6	D5	D4	D3	D2	D1	D0
0x003F	LAPAB	D7	D6	D5	D4	D3	D2	D1	D0
0x0040	ADC0SC1A	COCOA	AIENA	DIFFA	ADCHA				
0x0041	ADC0SC1B	COCOB	AIENB	DIFFB	ADCHB				
0x0042	ADC0CFG1	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0043	ADC0CFG2	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x0044	ADC0RHA	D[15:8]							
0x0045	ADC0RLA	D[7:0]							
0x0046	ADC0RHB	D[15:8]							
0x0047	ADC0RLB	D[7:0]							
0x0048	SPI0C1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0049	SPI0C2	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x004A	SPI0BR	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x004B	SPI0S	SPRF	0	SPTEF	MODF	0	0	0	0
0x004C	Reserved	—	—	—	—	—	—	—	—
0x004D	SPI0D	Bit 7	6	5	4	3	2	1	Bit 0
0x004E- 0x004F	Reserved	—	—	—	—	—	—	—	—

Table 4-3. Direct-Page Register Summary (Sheet 3 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0050	SPI1C1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0051	SPI1C2	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x0052	SPI1BR	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x0053	SPI1S	SPRF	0	SPTEF	MODF	0	0	0	0
0x0054	Reserved	—	—	—	—	—	—	—	—
0x0055	SPI1D	Bit 7	6	5	4	3	2	1	Bit 0
0x0056- 0x0057	Reserved	—	—	—	—	—	—	—	—
0x0058	SPI2C1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0059	SPI2C2	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x005A	SPI2BR	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x005B	SPI2S	SPRF	0	SPTEF	MODF	0	0	0	0
0x005C	Reserved	—	—	—	—	—	—	—	—
0x005D	SPI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x005E- 0x005F	Reserved	—	—	—	—	—	—	—	—
0x0060	PCNT_STATUS	0	PSTATE_INV			0	CURR_INV		
0x0061		0	0	0	0	0	SINVF	RCOVF	FCOVF
0x0062	PCNT_CTRL	SINVIE	RCOVFIE	FCOVFIE	MODE		CHANNEL_SEL		
0x0063		PCNT_EN	DIR	POL	CPWMS	FILTER VALUE			
0x0064	PCNT_FCMOD	PCNT_FCMOD_H							
0x0065		PCNT_FCMOD_L							
0x0066	PCNT_FCNTNTR	PCNT_FCNTNTR_H							
0x0067		PCNT_FCNTNTR_L							
0x0068	PCNT_RCMOD	PCNT_RCMOD_H							
0x0069		PCNT_RCMOD_L							
0x006A	PCNT_RCNTNTR	PCNT_RCNTNTR							
0x006B		PCNT_RCNTNTR							
0x006C	PCNT_PWM_MOD	PCNT_PWM_MOD_H							
0x006D		PCNT_PWM_MOD_L							
0x006E	PCNT_PWM_CH0_	PCNT_PWM_CH0_H							
0x006F	VAL	PCNT_PWM_CH0_L							
0x0070	PCNT_PWM_CH1_	PCNT_PWM_CH1_H							
0x0071	VAL	PCNT_PWM_CH1_L							
0x0072	PCNT_STATE	0	0	0	0	0	0	0	0
0x0073		0	0	PCOUNTER_STATE			CURR_STATE		
0x0074	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x0075	KBIPE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x0076	KBIES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x0077	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD

Table 4-3. Direct-Page Register Summary (Sheet 4 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0078	SCI0BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0079	SCI0BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x007A	SCI0C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x007B	SCI0C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x007C	SCI0S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x007D	SCI0S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x007E	SCI0C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x007F	SCI0D	Bit 7	6	5	4	3	2	1	Bit 0
0x0080	ADC1SC1A	COCOA	AIENA	DIFFA	ADCHA				
0x0081	ADC1SC1B	COCOB	AIENB	DIFFB	ADCHB				
0x0082	ADC1CFG1	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0083	ADC1CFG2	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x0084	ADC1RAH	D[15:8]							
0x0085	ADC1RAL	D[7:0]							
0x0086	ADC1RBH	D[15:8]							
0x0087	ADC1RBL	D[7:0]							
0x0088	SCI1BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0089	SCI1BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x008A	SCI1C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x008B	SCI1C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x008C	SCI1S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x008D	SCI1S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x008E	SCI1C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x008F	SCI1D	Bit 7	6	5	4	3	2	1	Bit 0
0x0090	IICA1	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x0091	IICF	MULT		ICR					
0x0092	IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	WUEN	0
0x0093	IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x0094	IICD	DATA							
0x0095	IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x0096	IICFLT	0	0	0	0	FLT3	FLT2	FLT1	FLT0
0x0097	IICSMB	FAK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
0x0098	IICA2	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0
0x0099	IICSLTH	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
0x009A	IICSLTL	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0
0x009B- 0x009F	Reserved	—	—	—	—	—	—	—	—
0x00A0	FTMSC	TOF	TOIE	CPWMS	CLKS		PS		
0x00A1	FTMCNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x00A2	FTMCNTL	Bit 7	6	5	4	3	2	1	Bit 0

Table 4-3. Direct-Page Register Summary (Sheet 5 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x00A3	FTMMODH	Bit 15	14	13	12	11	10	9	Bit 8
0x00A4	FTMMODL	Bit 7	6	5	4	3	2	1	Bit 0
0x00A5	FTMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x00A6	FTMC0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x00A7	FTMC0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x00A8	FTMC1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x00A9	FTMC1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x00AA	FTMC1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x00AB- 0x00AF	Reserved	—	—	—	—	—	—	—	—
0x00B0	CRCDH1	CRC Data Bit 31:24							
0x00B1	CRCDH0	CRC Data Bit 23:16							
0x00B2	CRCDL1	CRC Data Bit 15:8							
0x00B3	CRCDL0	CRC Data Bit 7:0							
0x00B4	CRCPH1	CRC Polynomial Bit 31:24							
0x00B5	CRCPH0	CRC Polynomial Bit 23:16							
0x00B6	CRCPL1	CRC Polynomial Bit 15:8							
0x00B7	CRCPL0	CRC Polynomial Bit 7:0							
0x00B8	CRCCTL	ToTW		ToTR		0	FXOR	SEED	CRCW
0x00B9- 0x00BF	Reserved	—	—	—	—	—	—	—	—

Use the LCD registers shown in table below to enable LCD functionality and display the LCD data.

Table 4-4. LCD Registers (Sheet 1 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1080	LCDPEN0	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
0x1081	LCDPEN1	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
0x1082	LCDPEN2	PEN23	PEN22	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
0x1083	LCDPEN3	PEN31	PEN30	PEN29	PEN28	PEN27	PEN26	PEN25	PEN24
0x1084	LCDPEN4	PEN39	PEN38	PEN37	PEN36	PEN35	PEN34	PEN33	PEN32
0x1085	LCDPEN5	—	—	—	—	PEN43	PEN42	PEN41	PEN40
0x1086- 0x1087	Reserved	—	—	—	—	—	—	—	—
0x1088	LCDBPEN0	BPEN7	BPEN6	BPEN5	BPEN4	BPEN3	BPEN2	BPEN1	BPEN0
0x1089	LCDBPEN1	BPEN15	BPEN14	BPEN13	BPEN12	BPEN11	BPEN10	BPEN9	BPEN8
0x108A	LCDBPEN2	BPEN23	BPEN22	BPEN21	BPEN20	BPEN19	BPEN18	BPEN17	BPEN16
0x108B	LCDBPEN3	BPEN31	BPEN30	BPEN29	BPEN28	BPEN27	BPEN26	BPEN25	BPEN24
0x108C	LCDBPEN4	BPEN39	BPEN38	BPEN37	BPEN36	BPEN35	BPEN34	BPEN33	BPEN32
0x108D	LCDBPEN5	—	—	—	—	BPEN43	BPEN42	BPEN41	BPEN40

Table 4-4. LCD Registers (Sheet 2 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x108E-0x108F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x1090	LCDWF0	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
0x1091	LCDWF1	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
0x1092	LCDWF2	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
0x1093	LCDWF3	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
0x1094	LCDWF4	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
0x1095	LCDWF5	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
0x1096	LCDWF6	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
0x1097	LCDWF7	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
0x1098	LCDWF8	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
0x1099	LCDWF9	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
0x109A	LCDWF10	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
0x109B	LCDWF11	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
0x109C	LCDWF12	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
0x109D	LCDWF13	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
0x109E	LCDWF14	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
0x109F	LCDWF15	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
0x10A0	LCDWF16	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
0x10A1	LCDWF17	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
0x10A2	LCDWF18	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
0x10A3	LCDWF19	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
0x10A4	LCDWF20	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
0x10A5	LCDWF21	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21
0x10A6	LCDWF22	BPHLCD22	BPGLCD22	BPFLCD22	BPELCD22	BPDLCD22	BPCLCD22	BPBLCD22	BPALCD22
0x10A7	LCDWF23	BPHLCD23	BPGLCD23	BPFLCD23	BPELCD23	BPDLCD23	BPCLCD23	BPBLCD23	BPALCD23
0x10A8	LCDWF24	BPHLCD24	BPGLCD24	BPFLCD24	BPELCD24	BPDLCD24	BPCLCD24	BPBLCD24	BPALCD24
0x10A9	LCDWF25	BPHLCD25	BPGLCD25	BPFLCD25	BPELCD25	BPDLCD25	BPCLCD25	BPBLCD25	BPALCD25
0x10AA	LCDWF26	BPHLCD26	BPGLCD26	BPFLCD26	BPELCD26	BPDLCD26	BPCLCD26	BPBLCD26	BPALCD26
0x10AB	LCDWF27	BPHLCD27	BPGLCD27	BPFLCD27	BPELCD27	BPDLCD27	BPCLCD27	BPBLCD27	BPALCD27
0x10AC	LCDWF28	BPHLCD28	BPGLCD28	BPFLCD28	BPELCD28	BPDLCD28	BPCLCD28	BPBLCD28	BPALCD28
0x10AD	LCDWF29	BPHLCD29	BPGLCD29	BPFLCD29	BPELCD29	BPDLCD29	BPCLCD29	BPBLCD29	BPALCD29
0x10AE	LCDWF30	BPHLCD30	BPGLCD30	BPFLCD30	BPELCD30	BPDLCD30	BPCLCD30	BPBLCD30	BPALCD30
0x10AF	LCDWF31	BPHLCD31	BPGLCD31	BPFLCD31	BPELCD31	BPDLCD31	BPCLCD31	BPBLCD31	BPALCD31
0x10B0	LCDWF32	BPHLCD32	BPGLCD32	BPFLCD32	BPELCD32	BPDLCD32	BPCLCD32	BPBLCD32	BPALCD32
0x10B1	LCDWF33	BPHLCD33	BPGLCD33	BPFLCD33	BPELCD33	BPDLCD33	BPCLCD33	BPBLCD33	BPALCD33
0x10B2	LCDWF34	BPHLCD34	BPGLCD34	BPFLCD34	BPELCD34	BPDLCD34	BPCLCD34	BPBLCD34	BPALCD34
0x10B3	LCDWF35	BPHLCD35	BPGLCD35	BPFLCD35	BPELCD35	BPDLCD35	BPCLCD35	BPBLCD35	BPALCD35
0x10B4	LCDWF36	BPHLCD36	BPGLCD36	BPFLCD36	BPELCD36	BPDLCD36	BPCLCD36	BPBLCD36	BPALCD36

Table 4-4. LCD Registers (Sheet 3 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x10B5	LCDWF37	BPHLCD37	BPGLCD37	BPFLCD37	BPELCD37	BPDLCD37	BPCLCD37	BPBLCD37	BPALCD37
0x10B6	LCDWF38	BPHLCD38	BPGLCD38	BPFLCD38	BPELCD38	BPDLCD38	BPCLCD38	BPBLCD38	BPALCD38
0x10B7	LCDWF39	BPHLCD39	BPGLCD39	BPFLCD39	BPELCD39	BPDLCD39	BPCLCD39	BPBLCD39	BPALCD39
0x10B8	LCDWF40	BPHLCD40	BPGLCD40	BPFLCD40	BPELCD40	BPDLCD40	BPCLCD40	BPBLCD40	BPALCD40
0x10B9	LCDWF41	BPHLCD41	BPGLCD41	BPFLCD41	BPELCD41	BPDLCD41	BPCLCD41	BPBLCD41	BPALCD41
0x10BA	LCDWF42	BPHLCD42	BPGLCD42	BPFLCD42	BPELCD42	BPDLCD42	BPCLCD42	BPBLCD42	BPALCD42
0x10BB	LCDWF43	BPHLCD43	BPGLCD43	BPFLCD43	BPELCD43	BPDLCD43	BPCLCD43	BPBLCD43	BPALCD43
0x10BC- 0x10BF	Reserved	—	—	—	—	—	—	—	—

High-page registers, shown in Table 4-5, 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 4-5. High-Page Register Summary (Sheet 1 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBD FR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	BKGDPE	1	STOPE	1	COPT		COPCLKS	COPW
0x1803- 0x1804	Reserved	—	—	—	—	—	—	—	—
0x1805	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1806	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1807	Reserved	—	—	—	—	—	—	—	—
0x1808	SCGC1	ADC1	ADC0	KBI	IIC	SCI3	SCI2	SCI1	SCI0
0x1809	SCGC2	CLKPRE	0	VREF	IRQ	LCD	SPI2	SPI1	SPI0
0x180A	SCGC3	PTH	PTG	PTF	PTE	PTD	PTC	PTB	PTA
0x180B	SCGC4	0	MUXCTRL	CRC	FTM	PDB	MTIM1	MTIM2	MTIM3
0x180C	SCGC5	PRACMP2	PRACMP1	PRACMP0	BKPT	IRTC	PCNT	0	FLS
0x180D	SIMIPS1	FTMCS		MTIM3CS	0	PCNTSS	MTIM2CS	0	0
0x180E	SIMIPS2	0	0	RX1IN	MTBASE1			0	MODTX1
0x180F	SIMIPS3	PCNTFCS		RX2IN	MTBASE2			DDRIVE	MODTX2
0x1810- 0x1812	Reserved	—	—	—	—	—	—	—	—
0x1813	SIMCO	0	0	0	0	CS			
0x1814	CCSCTRL	0	0	0	0	0	0	0	ICS_EXT_CLK_SEL
0x1815- 0x181B	Reserved	—	—	—	—	—	—	—	—
0x181C	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	BGBDS	BGBE
0x181D	SPMSC2	LPR	LPRS	LPWUI	0	PPDF	PPDACK	PPDE	PPDC

Table 4-5. High-Page Register Summary (Sheet 2 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x181E	Reserved	—	—	—	—	—	—	—	—
0x181F	SPMSC3	LVWF	LVWACK	LVDV	LVWV	LVWIE	0	0	0
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
0x1826	FCMD	FCMD							
0x1827- 0x182F	Reserved	—	—	—	—	—	—	—	—
0x1830	BKPTCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1831	BKPTCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1832	BKPTCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1833	BKPTCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1834	BKPTCCH	Bit 15	14	13	12	11	10	9	Bit 8
0x1835	BKPTCCL	Bit 7	6	5	4	3	2	1	Bit 0
0x1836	BKPTAC	BKPTAEN	RWAEN	RWA	TAGA	PAGESEL A	0	FMEN	FMDC
0x1837	BKPTBC	BKPTBEN	RWBEN	RWB	TAGB	PAGESEL B	0	0	0
0x1838	BKPTCC	BKPTCEN	RWCEN	RWC	TAGC	PAGESEL C	0	0	0
0x1839	BKPTS	0	0	0	0	0	AF	BF	CF
0x183A- 0x183B	Reserved	—	—	—	—	—	—	—	—
0x183C	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTE N
0x183D	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSSTE N
0x183E	ICSTRM	TRIM							
0x183F	ICSSC	DRST/DRS		DMX32	IREFST	CLKST		OSCINIT	FTRIM
0x1840	ADC0CV1H	CV1[15:8]							
0x1841	ADC0CV1L	CV1[7:0]							
0x1842	ADC0CV2H	CV2[15:8]							
0x1843	ADC0CV2L	CV2[7:0]							
0x1844	ADC0SC2	ADACT	ADTRG	ACFE	ACFGT	ACREN	0	REFSEL	
0x1845	ADC0SC3	CAL	CALF	0	0	ADCO	AVGE	AVGS	
0x1846	ADC0OFSH	OFS[15:8]							
0x1847	ADC0OFSL	OFS[7:0]							
0x1848	ADC0PGH	PG[15:8]							
0x1849	ADC0PGL	PG[7:0]							

Table 4-5. High-Page Register Summary (Sheet 3 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x184A	ADC0MGH	MG[15:8]							
0x184B	ADC0MGL	MG[7:0]							
0x184C	ADC0CLPD	0	0	CLPD					
0x184D	ADC0CLPS	0	0	CLPS					
0x184E	ADC0CLP4H	0	0	0	0	0	0	CLP4[9:8]	
0x184F	ADC0CLP4L	CLP4[7:0]							
0x1850	ADC0CLP3H	0	0	0	0	0	0	0	CLP3[8]
0x1851	ADC0CLP3L	CLP3[7:0]							
0x1852	ADC0CLP2	CLP2							
0x1853	ADC0CLP1	0	CLP1						
0x1854	ADC0CLP0	0	0	CLP0					
0x1855	Reserved	—	—	—	—	—	—	—	—
0x1856	ADC0CLMD	0	0	CLMD					
0x1857	ADC0CLMS	0	0	CLMS					
0x1858	ADC0CLM4H	0	0	0	0	0	0	CLM4[9:8]	
0x1859	ADC0CLM4L	CLM4[7:0]							
0x185A	ADC0CLM3H	0	0	0	0	0	0	0	CLM3[8]
0x185B	ADC0CLM3L	CLM3[7:0]							
0x185C	ADC0CLM2	CLM2							
0x185D	ADC0CLM1	0	CLM1						
0x185E	ADC0CLM0	0	0	CLM0					
0x185F	APC0TL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x1860	APC0TL2	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x1861- 0x1863	Reserved	—	—	—	—	—	—	—	—
0x1864	CLK_PRSC_H	0	0	0	0	0	Bit 10	Bit 9	Bit 8
0x1865	CLK_PRSC_L	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1866- 0x1867	Reserved	—	—	—	—	—	—	—	—
0x1868	SCI2BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x1869	SCI2BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x186A	SCI2C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x186B	SCI2C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x186C	SCI2S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x186D	SCI2S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x186E	SCI2C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x186F	SCI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x1870	SCI3BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x1871	SCI3BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x1872	SCI3C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT



Table 4-5. High-Page Register Summary (Sheet 4 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1873	SCI3C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x1874	SCI3S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x1875	SCI3S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x1876	SCI3C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x1877	SCI3D	Bit 7	6	5	4	3	2	1	Bit 0
0x1878	LCDC0	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
0x1879	LCDC1	LCDIEN	0	0	0	0	FCDEN	LCDWAI	LCDSTP
0x187A	LCDSUPPLY	CPSEL	HREFSEL	LADJ1	LADJ0	0	BBYPASS	VSUPPLY1	VSUPPLY0
0x187B	LCDRVC	RVEN	0	0	0	RVTRIM3	RVTRIM2	RVTRIM1	RVTRIM0
0x187C	LCDBCTL	BLINK	ALT	BLANK	0	BMODE	BRATE2	BRATE1	BRATE0
0x187D	LCDS	LCDIF	0	0	0	0	0	0	0
0x187E- 0x187F	Reserved	—	—	—	—	—	—	—	—
0x1880	IRTC_YEARMON	YEAR							
0x1881		0	0	0	0	MONTH			
0x1882	IRTC_DAYS	0	0	0	0	0	DAY_OF_WEEK		
0x1883		0	0	0	DAYS				
0x1884	IRTC_HOURMIN	0	0	0	HOURS				
0x1885		0	0	MINUTES					
0x1886	IRTC_SECONDS	0	0	0	0	0	0	0	0
0x1887		0	0	SECONDS					
0x1888	IRTC_ALM_YRMON	ALM_YEAR							
0x1889		0	0	0	0	ALM_MONTH			
0x188A	IRTC_ALM_DAYS	0	0	0	0	0	0	0	0
0x188B		0	0	0	ALM_DAYS				
0x188C	IRTC_ALM_HM	0	0	0	ALM_HOURS				
0x188D		0	0	ALM_MINUTES					
0x188E	IRTC_ALM_SEC	0	0	0	0	0	0	INC_S	DEC_S
0x188F		0	0	ALM_SECONDS					
0x1890	IRTC_CTRL	0	RTC_CLKOUT		0	0	0	0	SWR
0x1891		0	DSTEN	0	0	ALARM MATCH		WE1	WE0
0x1892	IRTC_STATUS	0	0	0	0	0	0	0	0
0x1893		0	PORB	CLV	WPE	OCAL	BERR	C_DON	INVAL
0x1894	IRTC_ISR	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0
0x1895		2Hz	1Hz	MIN	HR	DAY	ALM	CDT	TMPR
0x1896	IRTC_IER	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0
0x1897		2Hz	1Hz	MIN	HR	DAY	ALM	CDT	TMPR
0x1898	IRTC_COUNT_DN	0	0	0	0	0	0	0	0
0x1899		COUNTDOWN_COUNT							

Table 4-5. High-Page Register Summary (Sheet 5 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x189A–0x18A1	Reserved	—	—	—	—	—	—	—	—
0x18A2	IRTC_DST_HOUR	0	0	0	DST_START_HOUR				
0x18A3		0	0	0	DST_END_HOUR				
0x18A4	IRTC_DST_MNTH	0	0	0	0	DST_START_MONTH			
0x18A5		0	0	0	0	DST_END_MONTH			
0x18A6	IRTC_DST_DAY	0	0	0	DST_START_DAY				
0x18A7		0	0	0	DST_END_DAY				
0x18A8	IRTC_COMPEN	COMPENSATION_INTERVAL							
0x18A9		COMPENSATION_VALUE							
0x18AA	IRTC_TTSR_YM	TIME STAMP YEAR							
0x18AB		0	0	0	0	TIME STAMP MONTHS			
0x18AC	IRTC_TTSR_DAY	0	0	0	0	0	0	0	0
0x18AD		0	0	0	TIME STAMP DAY				
0x18AE	IRTC_TTSR_HM	0	0	0	TIME STAMP HOURS				
0x18AF		0	0	TIME STAMP MINUTES					
0x18B0	IRTC_TTSR_SEC	0	0	0	0	0	0	0	0
0x18B1		0	0	TIME STAMP SECONDS					
0x18B2	IRTC_TAMPER_S	0	0	0	0	0	TAMPER2	TAMPER1	BAT_RAMPER
0x18B3	CR	0	0	0	0	0	TAMPER2_CTRL	TAMPER1_CTRL	BAT_RAMPER_CTRL
0x18B4	IRTC_FILTER01_C	0	0	0	0	0	0	0	0
0x18B5	TRL	POL	CLKSEL	FILTER1 DURATION					
0x18B6	IRTC_FILTER23_C	POL	CLKSEL	FILTER2 DURATION					
0x18B7	TRL	0	0	0	0	0	0	0	0
0x18B8–0x18BF	Reserved	—	—	—	—	—	—	—	—
0x18C0	IRTC_STDBY_RAM0	MSB							
0x18C1		LSB							
0x18C2	IRTC_STDBY_RAM1	MSB							
0x18C3		LSB							
0x18C4	IRTC_STDBY_RAM2	MSB							
0x18C5		LSB							
0x18C6	IRTC_STDBY_RAM3	MSB							
0x18C7		LSB							
0x18C8	IRTC_STDBY_RAM4	MSB							
0x18C9		LSB							

Table 4-5. High-Page Register Summary (Sheet 6 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x18CA	IRTC_STDBY_RA M5	MSB							
0x18CB		LSB							
0x18CC	IRTC_STDBY_RA M6	MSB							
0x18CD		LSB							
0x18CE	IRTC_STDBY_RA M7	MSB							
0x18CF		LSB							
0x18D0	IRTC_STDBY_RA M8	MSB							
0x18D1		LSB							
0x18D2	IRTC_STDBY_RA M9	MSB							
0x18D3		LSB							
0x18D4	IRTC_STDBY_RA M10	MSB							
0x18D5		LSB							
0x18D6	IRTC_STDBY_RA M11	MSB							
0x18D7		LSB							
0x18D8	IRTC_STDBY_RA M12	MSB							
0x18D9		LSB							
0x18DA	IRTC_STDBY_RA M13	MSB							
0x18DB		LSB							
0x18DC	IRTC_STDBY_RA M14	MSB							
0x18DD		LSB							
0x18DE	IRTC_STDBY_RA M15	MSB							
0x18DF		LSB							
0x18E0	PTAPE	0	0	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x18E1	PTASE	0	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x18E2	PTADS	0	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x18E3	PTAIFE	0	0	PTAIFE5	PTAIFE4	PTAIFE3	PTAIFE2	PTAIFE1	PTAIFE0
0x18E4	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x18E5	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x18E6	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x18E7	PTBIFE	PTBIFE7	PTBIFE6	PTBIFE5	PTBIFE4	PTBIFE3	PTBIFE2	PTBIFE1	PTBIFE0
0x18E8	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0

Table 4-5. High-Page Register Summary (Sheet 7 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x18E9	PTCSE	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x18EA	PTCDS	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x18EB	PTCIFE	PTCIFE7	PTCIFE6	PTCIFE5	PTCIFE4	PTCIFE3	PTCIFE2	PTCIFE1	PTCIFE0
0x18EC	PTDPE	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x18ED	PTDSE	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x18EE	PTDDS	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x18EF	PTDIFE	PTDIFE7	PTDIFE6	PTDIFE5	PTDIFE4	PTDIFE3	PTDIFE2	PTDIFE1	PTDIFE0
0x18F0	PTEPE	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x18F1	PTESE	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x18F2	PTEDS	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x18F3	PTEIFE	PTEIFE7	PTEIFE6	PTEIFE5	PTEIFE4	PTEIFE3	PTEIFE2	PTEIFE1	PTEIFE0
0x18F4	PTFPE	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
0x18F5	PTFSE	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
0x18F6	PTFDS	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0
0x18F7	PTFIFE	PTFIFE7	PTFIFE6	PTFIFE5	PTFIFE4	PTFIFE3	PTFIFE2	PTFIFE1	PTFIFE0
0x18F8	PTGPE	PTGPE7	PTGPE6	PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
0x18F9	PTGSE	PTGSE7	PTGSE6	PTGSE5	PTGSE4	PTGSE3	PTGSE2	PTGSE1	PTGSE0
0x18FA	PTGDS	PTGDS7	PTGDS6	PTGDS5	PTGDS4	PTGDS3	PTGDS2	PTGDS1	PTGDS0
0x18FB	PTGIFE	PTGIFE7	PTGIFE6	PTGIFE5	PTGIFE4	PTGIFE3	PTGIFE2	PTGIFE1	PTGIFE0
0x18FC	PTHPE	—	—	—	—	—	—	PTHPE1	PTHPE0
0x18FD	PTHSE	—	—	—	—	—	—	PTHSE1	PTHSE0
0x18FE	PTHDS	—	—	—	—	—	—	PTHDS1	PTHDS0
0x18FF	PTHIFE	—	—	—	—	—	—	PTHIFE1	PTHIFE0
0x1900	ADC1CV1H	CV1[15:8]							
0x1901	ADC1CV1L	CV1[7:0]							
0x1902	ADC1CV2H	CV2[15:8]							
0x1903	ADC1CV2L	CV2[7:0]							
0x1904	ADC1SC2	ADACT	ADTRG	ACFE	ACFGT	ACREN	0	REFSEL	
0x1905	ADC1SC3	CAL	CALF	0	0	ADCO	AVGE	AVGS	
0x1906	ADC1OFSH	OFS[15:8]							
0x1907	ADC1OFSL	OFS[7:0]							
0x1908	ADC1PGH	PG[15:8]							
0x1909	ADC1PGL	PG[7:0]							
0x190A	ADC1MGH	MG[15:8]							
0x190B	ADC1MGL	MG[7:0]							
0x190C	ADC1CLPD	0	0	CLPD					
0x190D	ADC1CLPS	0	0	CLPS					
0x190E	ADC1CLP4H	0	0	0	0	0	0	CLP4[9:8]	
0x190F	ADC1CLP4L	CLP4[7:0]							
0x1910	ADC1CLP3H	0	0	0	0	0	0	0	CLP3[8]

Table 4-5. High-Page Register Summary (Sheet 8 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1911	ADC1CLP3L	CLP3[7:0]							
0x1912	ADC1CLP2	CLP2							
0x1913	ADC1CLP1	0	CLP1						
0x1914	ADC1CLP0	0	0	CLP0					
0x1915	Reserved	—	—	—	—	—	—	—	—
0x1916	ADC1CLMD	0	0	CLMD					
0x1917	ADC1CLMS	0	0	CLMS					
0x1918	ADC1CLM4H	0	0	0	0	0	0	CLM4[9:8]	
0x1919	ADC1CLM4L	CLM4[7:0]							
0x191A	ADC1CLM3H	0	0	0	0	0	0	0	CLM3[8]
0x191B	ADC1CLM3L	CLM3[7:0]							
0x191C	ADC1CLM2	CLM2							
0x191D	ADC1CLM1	0	CLM1						
0x191E	ADC1CLM0	0	0	CLM0					
0x191F	APC1TL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x1920	APC1TL2	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x1921- 0x1923	Reserved	—	—	—	—	—	—	—	—
0x1924	PRACMP0CS	ACEN	ACMPF	0	ACOPE	ACMPO	ACINTS		ACIEN
0x1925	PRACMP0C0	CMPHWT	ACPSEL			PRGHWT	ACNSEL		
0x1926	PRACMP0C1	PRGEN	PRGINS	0	PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
0x1927	PRACMP0C2	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
0x1928	PRACMP1CS	ACEN	ACMPF	0	ACOPE	ACMPO	ACINTS		ACIEN
0x1929	PRACMP1C0	CMPHWT	ACPSEL			PRGHWT	ACNSEL		
0x192A	PRACMP1C1	PRGEN	PRGINS	0	PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
0x192B	PRACMP1C2	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
0x192C	PRACMP2CS	ACEN	ACMPF	0	ACOPE	ACMPO	ACINTS		ACIEN
0x192D	PRACMP2C0	CMPHWT	ACPSEL			PRGHWT	ACNSEL		
0x192E	PRACMP2C1	PRGEN	PRGINS	0	PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
0x192F	PRACMP2C2	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
0x1930	VREFTRM	TRM							
0x1931	VREFSC	VREFEN	0	0	0	0	VREFST	MODE	
0x1932- 0x193F	Reserved	—	—	—	—	—	—	—	—
0x1940	PTAPF1	0	A1			0	A0		
0x1941	PTAPF2	0	A3			0	A2		
0x1942	PTAPF3	0	A5			0	A4		
0x1943	PTAPF4	0	0	0	0	0	A6		
0x1944	PTBPF1	0	B1			0	B0		
0x1945	PTBPF2	0	B3			0	B2		

Table 4-5. High-Page Register Summary (Sheet 9 of 9)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1946	PTBPF3	0		B5		0		B4	
0x1947	PTBPF4	0		B7		0		B6	
0x1948	PTCPF1	0		C1		0		C0	
0x1949	PTCPF2	0		C3		0		C2	
0x194A	PTCPF3	0		C5		0		C4	
0x194B	PTCPF4	0		C7		0		C6	
0x194C	PTDPF1	0		D1		0		D0	
0x194D	PTDPF2	0		D3		0		D2	
0x194E	PTDPF3	0		D5		0		D4	
0x194F	PTDPF4	0		D7		0		D6	
0x1950	PTEPF1	0		E1		0		E0	
0x1951	PTEPF2	0		E3		0		E2	
0x1952	PTEPF3	0		E5		0		E4	
0x1953	PTEPF4	0		E7		0		E6	
0x1954	PTFPF1	0		F1		0		F0	
0x1955	PTFPF2	0		F3		0		F2	
0x1956	PTFPF3	0		F5		0		F4	
0x1957	PTFPF4	0		F7		0		F6	
0x1958	PTGPF1	0		G1		0		G0	
0x1959	PTGPF2	0		G3		0		G2	
0x195A	PTGPF3	0		G5		0		G4	
0x195B	PTGPF4	0		G7		0		G6	
0x195C	PTHPF1	0		H1		0		H0	

Several reserved flash memory locations, shown in [Table 4-6](#), 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 IFR 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.

Table 4-6. Reserved Flash Memory Addresses

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAD	Reserved for Storage of VREF2 TRM	VREF TRIM							
0xFFAE	Reserved for Storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Reserved for Storage of ICSTRM	TRIM							
0xFFB0 – 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8 – 0xFFBC	Reserved	—	—	—	—	—	—	—	—
0xFFBD	NVPROT	FPS							
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).

## 4.5 Memory Management Unit

The memory management unit (MMU) allows the program and data space for the HCS08 family of microcontrollers to be extended beyond the 64 KB CPU addressable memory map. The MMU uses a paging scheme similar to that used on other MCU architectures, such as HCS12. The extended memory when used for data can also be accessed linearly using a linear address pointer and data access registers.

### 4.5.1 Features

Key features of the MMU module are:

- Memory management unit extends the HCS08 memory space
  - up to 4 MB for program and data space (Maximum)
- Extended program space using paging scheme
  - PPAGE register used for page selection
  - fixed 16 KB memory window
  - architecture supports up to 256 pages at 16 KB each
- Extended data space using linear address pointer

- up to 22-bit linear address pointer (Maximum)
- linear address pointer and data register provided in direct page allows access of complete flash memory map using direct-page instructions
- optional auto increment of pointer when data accessed
- supporting a 2s compliment addition/subtraction to address pointer without using any math instructions or memory resources
- supporting word accesses to any address specified by the linear address pointer when using LDHX, STHX instructions

## 4.5.2 Register Definition

### 4.5.2.1 Program Page Register (PPAGE)

The HCS08 core architecture limits the CPU addressable space available to 64 KB. The address space can be extended to 128 KB using a paging window scheme. The program page (PPAGE) allows for selecting one of the 16 KB blocks to be accessed through the program page window located at 0x8000–0xBFFF. The CALL and RTC instructions can load or store the value of PPAGE onto or from the stack during program execution. After any reset, PPAGE is set to PAGE 2.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	XA16	XA15	XA14
W								
Reset:	0	0	0	0	0	0	1	0

Figure 4-3. Program Page Register (PPAGE)

Table 4-7. Program Page Register Field Descriptions

Field	Description
2:0 XA16:XA14	When the CPU addresses the paging window, 0x8000–0xBFFF, the value in the PPAGE register along with the CPU addresses A13:A0 are used to create a 17-bit extended address.

### 4.5.2.2 Linear Address Pointer Registers 2:0 (LAP2:LAP0)

These three registers, LAP2:LAP0, contain the 17-bit linear address that allows the user to access any flash location in the extended address map. This register is used in conjunction with the data registers, linear byte (LB), linear byte post increment (LBP) and linear word post increment (LWP). The contents of LAP2:LAP0 will auto-increment when accessing data using the LBP and LWP registers. The contents of LAP2:LAP0 can be increased by writing an 8-bit value to LAPAB.



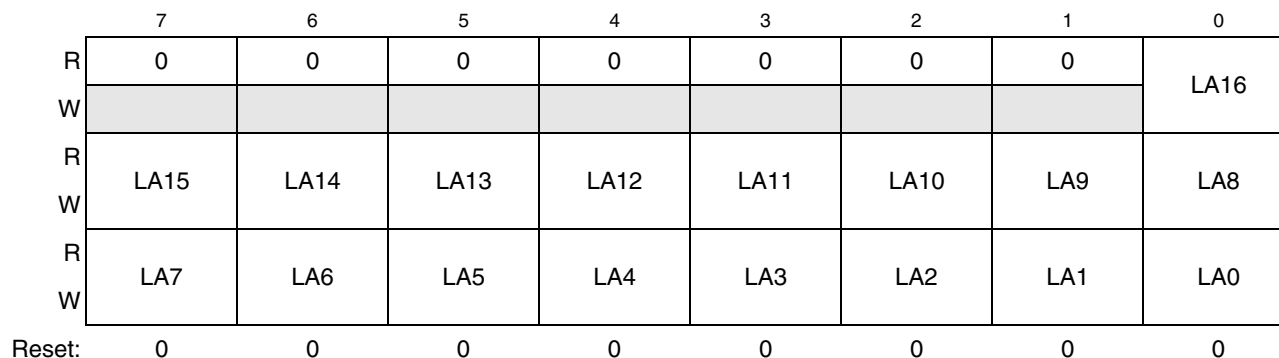


Figure 4-4. Linear Address Pointer Registers 2:0 (LAP2:LAP0)

Table 4-8. Linear Address Pointer Registers 2:0 Field Descriptions

Field	Description
16:0 LA16:LA0	The values in LAP2:LAP0 are used to create a 17-bit linear address pointer. The value in these registers are used as the extended address when accessing any of the data registers LB, LBP and LWP.

### 4.5.2.3 Linear Word Post Increment Register (LWP)

This register is one of three data registers that the user can use to access any flash memory location in the extended address map. When LWP is accessed, the contents of LAP2:LAP0 make up the extended address of the flash memory location to be addressed. When accessing data using LWP, the contents of LAP2:LAP0 will increment after the read or write is complete.

Accessing LWP does the same thing as accessing LBP. The MMU register ordering of LWP followed by LBP, allow the user to access data by words using the LDHX or STHX instructions of the LWP register.

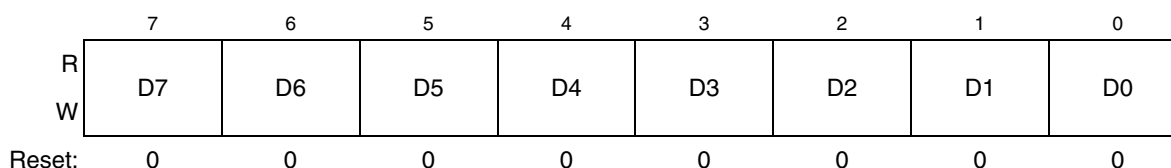


Figure 4-5. Linear Word Post Increment Register (LWP)

Table 4-9. Linear Word Post Increment Register Field Descriptions

Field	Description
7:0 D7:D0	Reads of this register first return the data value pointed to by the linear address pointer, LAP2:LAP0 and then increment LAP2:LAP0. Writes to this register first write the data value to the memory location specified by the linear address pointer and then increment LAP2:LAP0. Writes to this register are most commonly used when writing to the flash block(s) during programming.

### 4.5.2.4 Linear Byte Post Increment Register (LBP)

This register is one of three data registers that the user can use to access any flash memory location in the extended address map. When LBP is accessed, the contents of LAP2:LAP0 make up the extended address

of the flash memory location to be addressed. When accessing data using LBP, the contents of LAP2:LAP0 will increment after the read or write is complete.

Accessing LBP does the same thing as accessing LWP. The MMU register ordering of LWP followed by LBP, allow the user to access data by words using the LDHX or STHX instructions with the address of the LWP register.

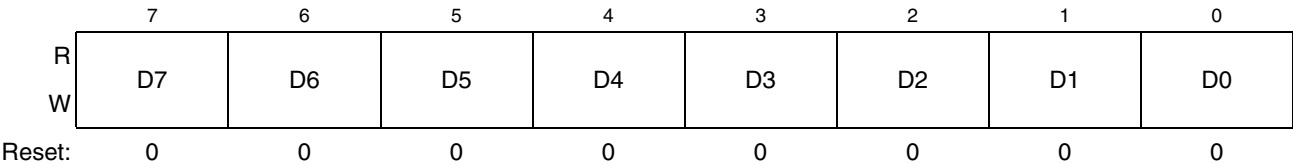


Figure 4-6. Linear Byte Post Increment Register (LBP)

Table 4-10. Linear Byte Post Increment Register Field Descriptions

Field	Description
7:0 D7:D0	Reading of this register first return the data value pointed to by the linear address pointer, LAP2:LAP0 and then increment LAP2:LAP0. Writes to this register first write the data value to the memory location specified by the linear address pointer and then increment LAP2:LAP0. Writes to this register are most commonly used when writing to the flash block(s) during programming.

4.5.2.5 Linear Byte Register (LB)

This register is one of three data registers that the user can use to access any flash memory location in the extended address map. When LB is accessed, the contents of LAP2:LAP0 make up the extended address of the flash memory location to be addressed.

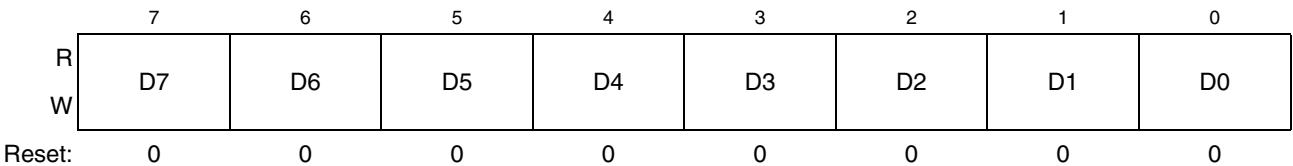


Figure 4-7. Linear Byte Register (LB)

Table 4-11. Linear Data Register Field Descriptions

Field	Description
7:0 D7:D0	Reading of this register returns the data value pointed to by the linear address pointer, LAP2:LAP0. Writes to this register will write the data value to the memory location specified by the linear address pointer. Writes to this register are most commonly used when writing to the flash block(s) during programming.

4.5.2.6 Linear Address Pointer Add Byte Register (LAPAB)

The user can increase or decrease the contents of LAP2:LAP0 by writing a 2s compliment value to LAPAB. The value written is added to the current contents of LAP2:LAP0.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	D7	D6	D5	D4	D3	D2	D1	D0
Reset:	0	0	0	0	0	0	0	0

Figure 4-8. Linear Address Pointer Add Byte Register (LAPAB)

Table 4-12. Linear Address Pointer Add Byte Register Field Descriptions

Field	Description
7:0 D7:D0	The 2s compliment value written to LAPAB will be added to contents of the linear address pointer register, LAP2:LAP0. Writing a value of 0x7f to LAPAB will increase LAP by 127, a value of 0xff will decrease LAP by 1, and a value of 0x80 will decrease LAP by 128.

## 4.5.3 Functional Description

### 4.5.3.1 Memory Expansion

The HCS08 core architecture limits the CPU addressable space available to 64 KB. The program page (PPAGE) allows for integrating up to 128 KB of flash into the system by selecting one of the 16 KB blocks to be accessed through the paging window located at 0x8000-0xBFFF. The MMU module also provides a linear address pointer that allows extension of data access up to 128 KB.

#### 4.5.3.1.1 Program Space

The PPAGE register holds the page select value for the paging window. The value in PPAGE can be manipulated by using normal read and write instructions as well as the CALL and RTC instructions. The user must not change PPAGE directly when running from paged memory, only CALL and RTC can be used.

When the MMU detects that the CPU is addressing the paging window, the value currently in PPAGE will be used to create an extended address for MCU's decode logic to select the desired flash location.

As seen in [Figure 4-1](#) and [Figure 4-2](#), the flash blocks in the CPU addressable memory can be accessed directly or using the paging window and PPAGE register. For example, the flash from location 0x4000-0x7FFF can be accessed directly or using the paging window, PPAGE = 1, address 0x8000-0xBFFF.

#### 4.5.3.1.2 CALL and RTC (Return from Call) Instructions

CALL and RTC are instructions that perform automated page switching when executed in the user program. CALL is similar to a JSR instruction, but the subroutine that it called can be located anywhere in the normal 64 KB address space or on any page of program memory.

During the execution of a CALL instruction, the CPU:

- Stacks the return address.
- Pushes the current PPAGE value onto the stack.

- Writes the new instruction-supplied PPAGE value into the PPAGE register.
- Transfers control to the subroutine of the new instruction-supplied address.

This sequence is not interruptible; there is no need to inhibit interrupts during CALL execution. A CALL can be executed from any address in memory to any other address.

The new PPAGE value is provided by an immediate operand in the instruction along with the address within the paging window, 0x8000-0xBFFF.

RTC is similar to an RTS instruction.

The RTC instruction terminates subroutines invoked by a CALL instruction.

During the execution of an RTC instruction, the CPU:

- Pulls the old PPAGE value from the stack and loads it into the PPAGE register
- Pulls the 16-bit return address from the stack and loads it into the PC
- Resumes execution at the return address

This sequence is not interruptible; there is no need to inhibit interrupts during RTC execution. An RTC can be executed from any address in memory.

#### 4.5.3.1.3 Data Space

The linear address pointer registers, LAP2:LAP0 along with the linear data register allow the CPU to read or write any address in the extended flash memory space. This linear address pointer may be used to access data from any memory location while executing code from any location in extended memory, including accessing data from a different PPAGE than the currently executing program.

To access data using the linear address pointer, the user would first setup the extended address in the 17-bit address pointer, LAP2:LAP0. Accessing one of the three linear data registers LB, LBP and LWP will access the extended memory location specified by LAP2:LAP0. The three linear data registers access the memory locations in the same way, however the LBP and LWP will also increment LAP2:LAP0.

Accessing either the LBP or LWP registers allows a user program to read successive memory locations without re-writing the linear address pointer. Accessing LBP or LWP does the exact same function. However, because of the address mapping of the registers with LBP following LWP, a user can do word accesses in the extended address space using the LDHX or STHX instructions to access location LWP.

The MMU supports the addition of a 2s complement value to the linear address pointer without using any math instructions or memory resources. Writing to LAPAB with a 2s complement value will cause the MMU to add that value to the existing value in LAP2:LAP0.

#### 4.5.3.1.4 PPAGE and Linear Address Pointer to Extended Address

See [Figure 4-1](#) or [Figure 4-2](#), on how the program PPAGE memory pages and the linear address pointer are mapped to extended address space.

## 4.6 RAM

The MC9S08GW64 series include 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.

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 MC9S08GW64 series, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct-page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor-provided equate file).

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

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See “[Section 4.7.5](#)”, for a detailed description of the security feature.

## 4.7 Flash

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

### 4.7.1 Features

Features of the flash memory include:

- Flash size
  - MC9S08GW64: 65,326 bytes
  - MC9S08GW32: 32,768 bytes
- Single power supply program and erase
- Automated program and erase algorithm
- Fast program and erase operation
- Burst program command for faster flash array program times
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible protection scheme to prevent accidental program or erase
- Security feature to prevent unauthorized access to the flash and RAM

- Auto power-down for low-frequency read accesses

### 4.7.2 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 4.9.1, “Flash Clock Divider Register \(FCDIV\)”](#)). This register can be written only once, so normally this write is done during reset initialization. It is not recommended to write FCDIV when 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 4-13](#) 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 4-13. 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

### 4.7.3 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 4-9](#) 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.

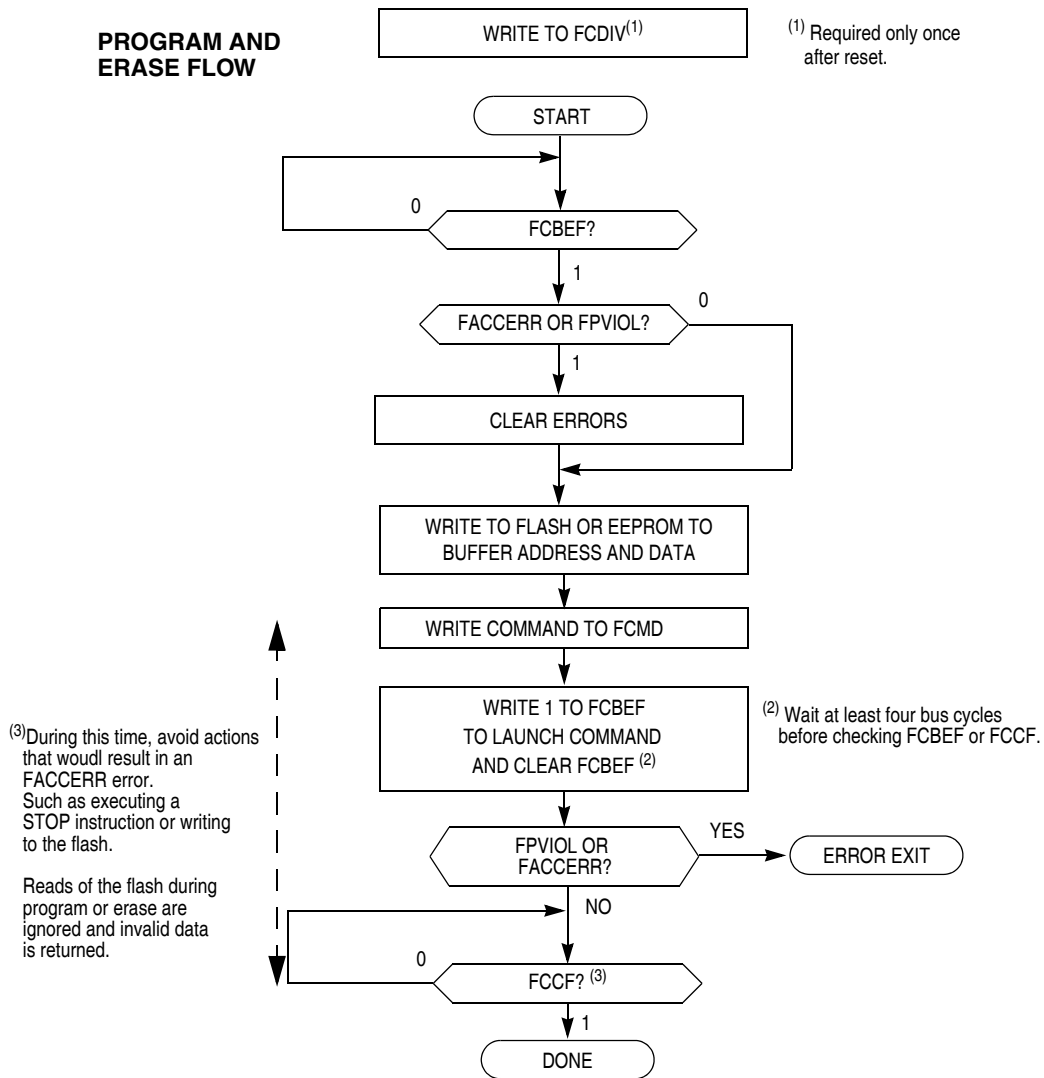


Figure 4-9. Flash Program and Erase Flowchart

#### 4.7.4 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 is 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.

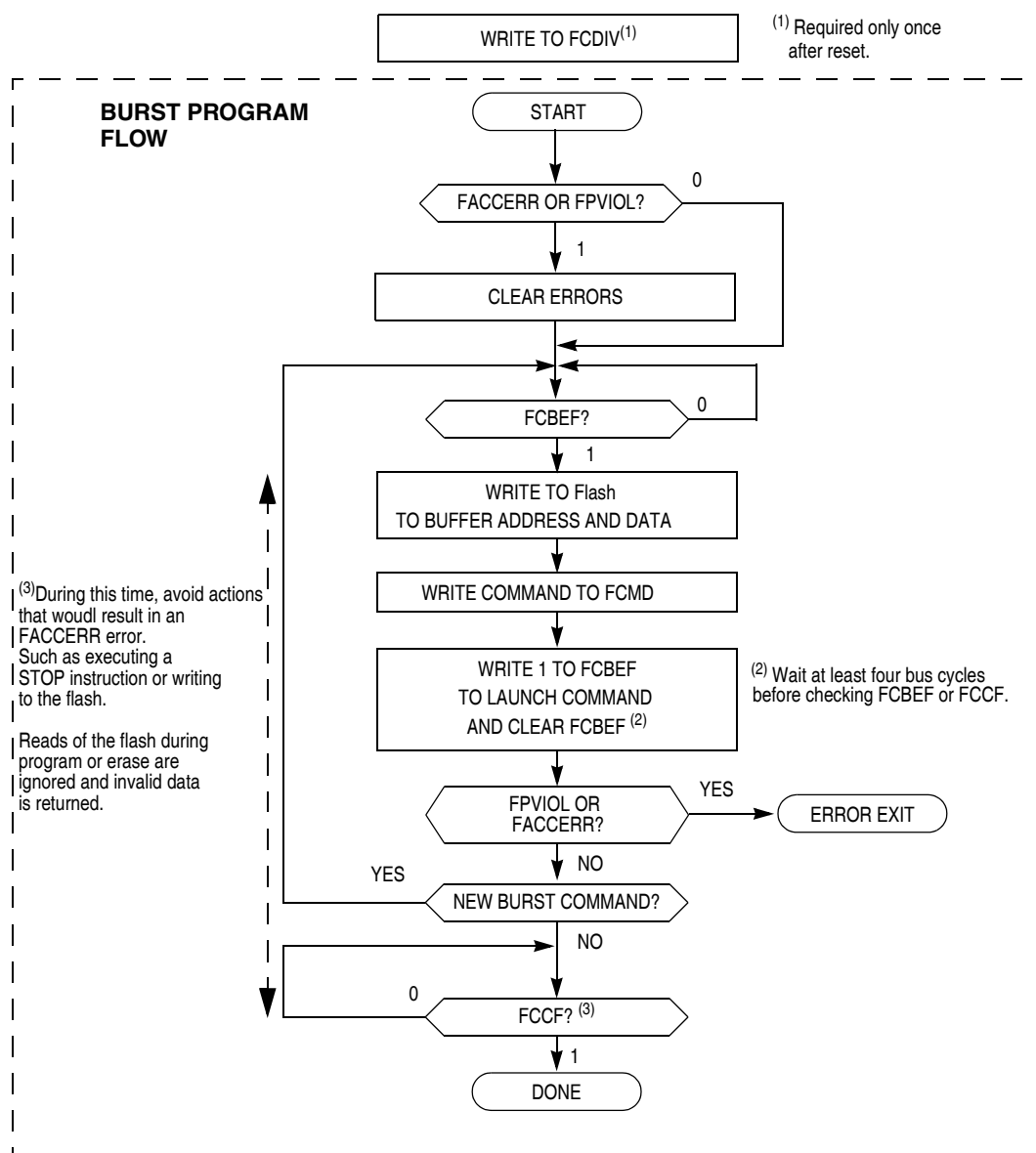


Figure 4-10. Flash Burst Program Flowchart

### 4.7.5 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

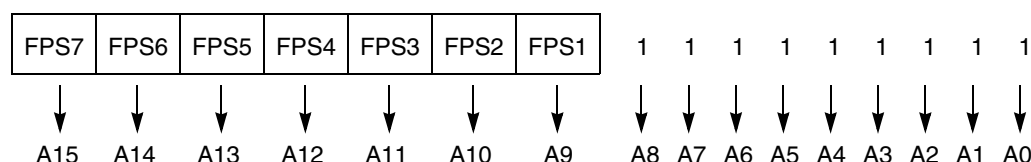
### 4.7.6 Flash Block Protection

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 4.9.4, “Flash Protection Register \(FPROT and NVPROT\)”](#)).

After exit from reset, FPROT is loaded with the contents of the NVPROT location, which is in the nonvolatile register block of the flash memory. With FPDIS set all FPROT bits are writable, but with FPDIS clear the FPS bits are writable as long as the size of the protected region is being increased. Any FPROT write that attempts to decrease the size of the protected region will be ignored and the FPVIOL flag in the FSTAT register will not be set. Because NVPROT is within the last 512 bytes of flash, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands, which allows a protected flash memory to be erased and reprogrammed.

The block protection mechanism is illustrated in [Figure 4-11](#). 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 4-11. 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.

### 4.7.7 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 unprotected bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the flash memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFD) are redirected, though the reset vector (0xFFFE:FFFF) is not.

For example, if 512 bytes of flash are protected, the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFD) are redirected to the locations 0xFDC0–0xFDFD. For instance, if an SPI interrupt is taken, the values in the locations 0xFDE0:FDE1 are used for the vector instead of the values in the locations 0xFFE0:FFE1. This allows the user to reprogram the unprotected portion of the flash with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

## 4.8 Security

The MC9S08GW64 series include 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 must 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 must not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was just written matches the key stored in the flash locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

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

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

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

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase flash if necessary.
3. Blank check flash. Provided flash is completely erased, security is disengaged until the next reset. To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

## 4.9 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 4-5](#) and [Table 4-6](#) 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.

### 4.9.1 Flash Clock Divider Register (FCDIV)

Bit 7 of this register is a read-only flag. DIV Bits 6:0 may be read at any time but can be written only once. 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. [Table 4-15](#) shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

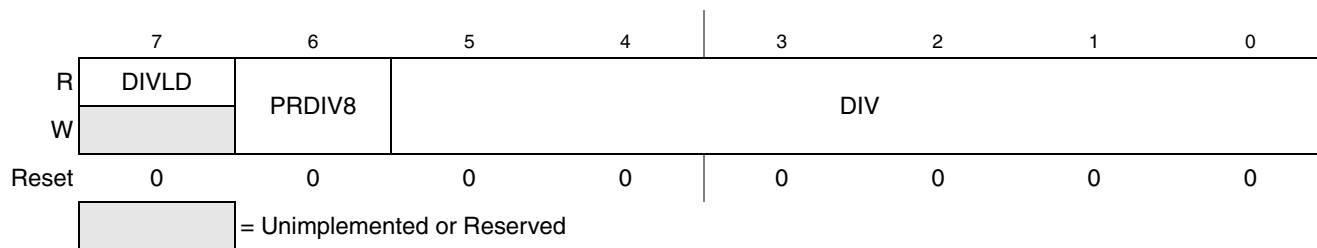


Figure 4-12. Flash Clock Divider Register (FCDIV)

Table 4-14. 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.
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	<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 μs to 6.7 μs. 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> .

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

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

[Table 4-15](#) shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

Table 4-15. Flash Clock Divider Settings

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

## 4.9.2 Flash Options Register (FOPT and NVOPT)

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

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
W								

Reset This register is loaded from nonvolatile location NVOPT during reset.


 = Unimplemented or Reserved

Figure 4-13. Flash Options Register (FOPT)

Table 4-16. 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 4.8, “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 4-17</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 4.8, “Security.”</a>

Table 4-17. 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.

### 4.9.3 Flash Configuration Register (FCNFG)

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


 = Unimplemented or Reserved

Figure 4-14. Flash Configuration Register (FCNFG)

Table 4-18. 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 4.8, “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.

### 4.9.4 Flash Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT is copied from flash into FPROT. FPROT can be read at any time. With FPDIS set, all bits are writable, but with FPDIS clear the FPS bits are writable as long as the size of the protected region is being increased. Any FPROT write that attempts to decrease the size of the protected region will be ignored.

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

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

Figure 4-15. Flash Protection Register (FPROT)

Table 4-19. 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 FPS7:FPS1 is block protected (program and erase not allowed). 1 No flash block is protected.

## 4.9.5 Flash Status Register (FSTAT)

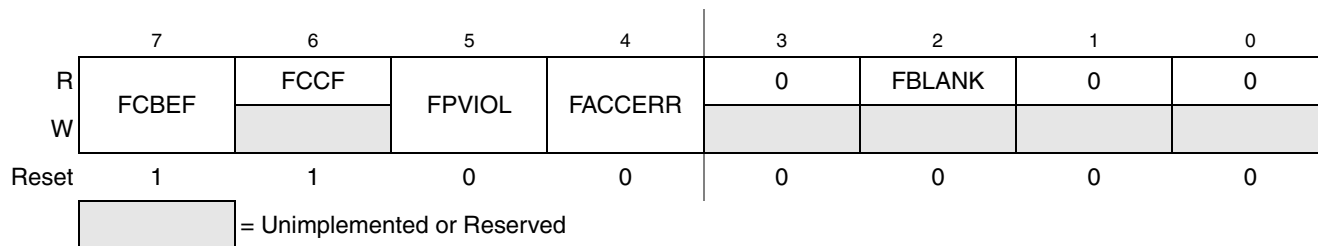


Figure 4-16. Flash Status Register (FSTAT)

Table 4-20. 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.



Table 4-20. FSTAT Register Field Descriptions (continued)

Field	Description
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 4.7.5, “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 FCBF 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).

### 4.9.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in [Table 4-21](#). Refer to [Section 4.7.3, “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 4-17. Flash Command Register (FCMD)

Table 4-21. 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. 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 MC9S08GW64 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.

#### NOTE

When MC9S08GW64 series operation voltage is lower than LVD threshold, LVD should be disabled.

### 5.2 Features

Reset and interrupt features include:

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

### 5.3 MCU Reset

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

The MC9S08GW64 series have the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Computer operating properly (COP) timer
- Illegal opcode detect (IOP)
- Illegal address detect (ILAD)
- Low-voltage detect (LVD)
- 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 COP watchdog becomes enabled (see [Section 5.8.4, “System Options Register 1 \(SOPT1\),”](#) for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPT in SOPT1 register. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The COPCLKS bit in SOPT1 (see [Section 5.8.4, “System Options Register 1 \(SOPT1\),”](#) 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](#) summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz LPO clock source and the longest time-out ( $2^{10}$  cycles)

When the bus clock source is selected, windowed COP operation is available by setting COPW in the SOPT1 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 LPO clock source is selected, windowed COP operation is not available.

The COP counter is initialized by the first writes to the SOPT1 register and after any system reset. Subsequent writes to SOPT1 have no effect on COP operation. Even if the application will use the reset default settings of COPT, COPCLKS, and COPW bits, the user must write to the write-once SOPT1 register 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 must not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

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 LPO 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.

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 LPO clock	N/A	2 <sup>5</sup> cycles (32 ms <sup>2</sup> )
0	1:0	1 kHz LPO clock	N/A	2 <sup>8</sup> cycles (256 ms <sup>2</sup> )
0	1:1	1 kHz LPO clock	N/A	2 <sup>10</sup> cycles (1.024 s <sup>2</sup> )
1	0:1	BUSCLK	6144 cycles	2 <sup>13</sup> cycles
1	1:0	BUSCLK	49,152 cycles	2 <sup>16</sup> cycles
1	1:1	BUSCLK	196,608 cycles	2 <sup>18</sup> 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.

## 5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing 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 the IRQ pin or a timer-overflow event.

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 (enabled) 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.

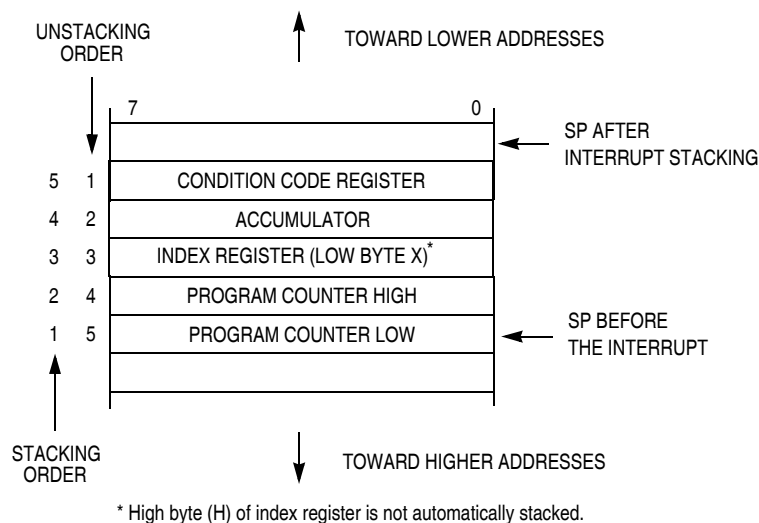
## NOTE

For compatibility with M68HC08 devices, the H register is not automatically saved and restored. 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 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQ status and control register (IRQSC). To enable the IRQ function, the A5 in PTAPF3 register has to be set to 0100b. When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events. When the MCU is in stop mode and system clocks are shut down, a separate asynchronous path is used so the IRQ pin (if enabled) can wake the MCU.

There is a delay between PUE assertion and internal pull up being enabled on IRQ pad, which may cause spurious interrupt in IRQ. To avoid this, the following sequence is recommended to initialize IRQ.

1. Mask IRQ interrupt by clearing IRQIE in IRQSC.
2. Enable the IRQ polarity by setting IRQEDG in IRQSC register.
3. If using internal pullup/pulldown device, configure the associated pullup enable bits in IRQSC
4. Enable IRQMOD bit in IRQSC register if required.
5. Write to IRQACK in IRQSC to clear any false interrupts.
6. Set IRQIE in IRQSC to enable interrupt.

### 5.5.2.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in IRQSC must be 1 in order for the IRQ pin to act as the interrupt request (IRQ) input. As an IRQ input, the user can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD) and whether an event causes an interrupt or only sets the IRQF flag which can be polled by software (IRQIE).

The IRQ pin, when enabled, defaults to use an internal pull device (IRQPDD = 0), configured as a pullup or pulldown depending on the polarity chosen. If the user desires to use an external pullup or pulldown, the IRQPDD can be written to a 1 to turn off the internal device.

BIH and BIL instructions may be used to detect the level on the IRQ pin when the pin is configured to act as the IRQ input.

### 5.5.2.2 Edge and Level Sensitivity

The IRQMOD control bit reconfigures the detection logic so it detects edge events and pin levels. In the edge and level detection mode, the IRQF status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

### 5.5.3 Interrupt Vectors, Sources, and Local Masks

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

When an interrupt condition occurs, an associated flag bit becomes set. If the associated local interrupt enable is 1, an interrupt request is sent to the CPU. Within the CPU, if the global interrupt mask (I bit in the CCR) is 0, the CPU will finish the current instruction; stack the PCL, PCH, X, A, and CCR CPU registers; set the I bit; and then fetch the interrupt vector for the highest priority pending interrupt. Processing then continues in the interrupt service routine.



Table 5-2. Interrupt Vectors

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lowest  Highest	47 through 37	0xFF80:FF81 through 0xFF94:FF95	Unused Vector Space (available for user program)				
	36	0xFF96/0xFF97	Vpdb	PDB	IF	IE	PDB interrupt
	35	0xFF98/0xFF99	Vpdber	PDB	ERRA,ERRB	N/A	PDB sequence error
	34	0xFF9A/0xFF9B	Vlcd	LCD	LCDF	LCDIE	LCD Frame Interrupt
	33	0xFF9C/0xFF9D	Virtc	IRTC	IRTC_ISR	IRTC_IER	IRTC interrupt
	32	0xFF9E/0xFF9F	Vkeyboard	KBI	KBF	KBIE	Keyboard pins
	31	0xFFC0/0xFFC1	Vpcnt	PCNT	SINV,RCOVF,FCOVF	SINVIE,RCOVIE,FCOVIE	PCNT interrupt
	30	0xFFC2/0xFFC3	Vpracmp2	PRACMP2	ACMPF	ACIEN	PRACMP2 interrupt
	29	0xFFC4/0xFFC5	Vpracmp1	PRACMP1	ACMPF	ACIEN	PRACMP1 interrupt
	28	0xFFC6/0xFFC7	Vpracmp0	PRACMP0	ACMPF	ACIEN	PRACMP0 interrupt
	27	0xFFC8/0xFFC9	Vadc1	ADC1	COCO	AIEN	ADC1
	26	0xFFCA/0xFFCB	Vadc0	ADC0	COCO	AIEN	ADC0
	25	0xFFCC/0xFFCD	Vmtim2	MTIM2	TOF	TOIE	MTIM2 overflow
	24	0xFFCE/0xFFCF	Vmtim1	MTIM1	TOF	TOIE	MTIM1 overflow
	23	0xFFD0/0xFFD1	Vmtim0	MTIM0	TOF	TOIE	MTIM0 overflow
	22	0xFFD2/0xFFD3	Vftmovf	FTM	TOF	TOIE	FTM overflow
	21	0xFFD4/0xFFD5	Vftmch0	FTM	CH0F	CH0IE	FTM channel 0
	20	0xFFD6/0xFFD7	Vftmch1	FTM	CH1F	CH1IE	FTM channel 1
	19	0xFFD8/0xFFD9	Viic	IIC	IICIS	IICIE	IIC control
	18	0xFFDA/0xFFDB	Vspi2	SPI2	SPIF, MODF,SPTEF	SPIE, SPIE, SPTIE	SPI2
	17	0xFFDC/0xFFDD	Vspi1	SPI1	SPIF, MODF,SPTEF	SPIE, SPIE, SPTIE	SPI1
	16	0xFFDE/0xFFDF	Vspi0	SPI0	SPIF, MODF,SPTEF	SPIE, SPIE, SPTIE	SPI0
	15	0xFFE0/0xFFE1	Vsci3tx	SCI3	TDRE, TC	TIE, TCIE	SCI3 transmit
	14	0xFFE2/0xFFE3	Vsci3rx	SCI3	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI3 receive
	13	0xFFE4/0xFFE5	Vsci3err	SCI3	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI3 error
	12	0xFFE6/0xFFE7	Vsci2tx	SCI2	TDRE, TC	TIE, TCIE	SCI2 transmit
	11	0xFFE8/0xFFE9	Vsci2rx	SCI2	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI2 receive
	10	0xFFEA/0xFFEB	Vsci2err	SCI2	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI2 error
	9	0xFFEC/0xFFED	Vsci1tx	SCI1	TDRE, TC	TIE, TCIE	SCI1 transmit
	8	0xFFEE/0xFFEF	Vsci1rx	SCI1	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI1 receive
	7	0xFFFF0/0xFFFF1	Vsci1err	SCI1	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI1 error
	6	0xFFFF2/0xFFFF3	Vsci0tx	SCI0	TDRE, TC	TIE, TCIE	SCI0 transmit
	5	0xFFFF4/0xFFFF5	Vsci0rx	SCI0	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI0 receive
	4	0xFFFF6/0xFFFF7	Vsci0err	SCI0	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI0 error
	3	0xFFFF8/0xFFFF9	Vlvd	System control	LVDF, LVWF	LVDIE, LVWIE	Low-voltage detect, Low-voltage warning
	2	0xFFFFA/0xFFFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	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 — — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address

## 5.6 Low-Voltage Detect (LVD) System

The MC9S08GW64 series include a system to protect against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system is composed of a power-on reset (POR) circuit and an LVD circuit with a user selectable trip voltage, either high ( $V_{LVDH}$ ) or low ( $V_{LVDL}$ ). The LVD circuit is enabled when LVDE in SPMSC1 is high and the trip voltage is selected by LVDV in SPMSC3. The LVD is disabled upon entering either of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU will enter stop3 instead of stop2, and the current consumption in stop3 with the LVD enabled will be greater.

### 5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the 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 Detection (LVD) Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF in SPMSC1 will be set and an LVD interrupt request will occur. The LVDF bit is cleared by writing a 1 to the LVDACK bit in SPMSC1.

### 5.6.4 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 is above, the LVD voltage. The LVW also has an interrupt associated with it, enabled by setting the LVWIE bit in the SPMSC3 register. If enabled, an LVW interrupt request will occur when the LVWF is set. LVWF is cleared by writing a 1 to the LVWACK bit in SPMSC3. There are two user selectable trip voltages for the LVW, one high ( $V_{LVWH}$ ) and one low ( $V_{LVWL}$ ). The trip voltage is selected by LVWV in SPMSC3.

## 5.7 Peripheral Clock Gating

The MC9S08GW64 series include 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, most of the peripheral clocks will be disabled in order to save the silicon power. The user have to enable the clocks to the peripherals before running the peripherals. To get lowest power consumption the user have to disable the peripherals clocks as many as possible. For lowest possible run or wait currents, user software must 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 SCGCx. 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 must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.

In stop modes, the bus clock is disabled for all gated peripherals, regardless of the settings in SCGCx registers.

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

One 8-bit register in the direct-page register space and couples of 8-bit registers in the high-page register space are related to reset and system control.


Refer to [Table 4-3](#) and [Table 4-5](#) in [Chapter 4](#), “Memory,” of this manual 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 Interrupt Pin Request Status and Control Register (IRQSC)

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

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

 = Unimplemented or Reserved

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

Table 5-3. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	<b>Interrupt Request (IRQ) Pull Device Disable</b> — This read/write control bit is used to disable the internal pullup/pulldown device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	<b>Interrupt Request (IRQ) Edge Select</b> — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When IRQEDG = 1 and the internal pull device is enabled, the pullup device is reconfigured as an optional pulldown device. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	<b>IRQ Pin Enable</b> — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	<b>IRQ Flag</b> — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 IRQACK	<b>IRQ Acknowledge</b> — This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.
1 IRQIE	<b>IRQ Interrupt Enable</b> — This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever IRQF = 1.
0 IRQMOD	<b>IRQ Detection Mode</b> — This read/write control bit selects either edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels that are detected as interrupt request events. See <a href="#">Section 5.5.2.2, “Edge and Level Sensitivity”</a> for more details. 0 IRQ event on falling edges or rising edges only. 1 IRQ event on falling edges and low levels or on rising edges and high levels.

## 5.8.2 System Reset Status Register (SRS)

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

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u <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 5-3. System Reset Status (SRS)**

**Table 5-4. 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 (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	<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.

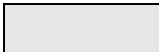
Table 5-4. SRS Register Field Descriptions (continued)

Field	Description
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.

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

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

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

 = Unimplemented or Reserved

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

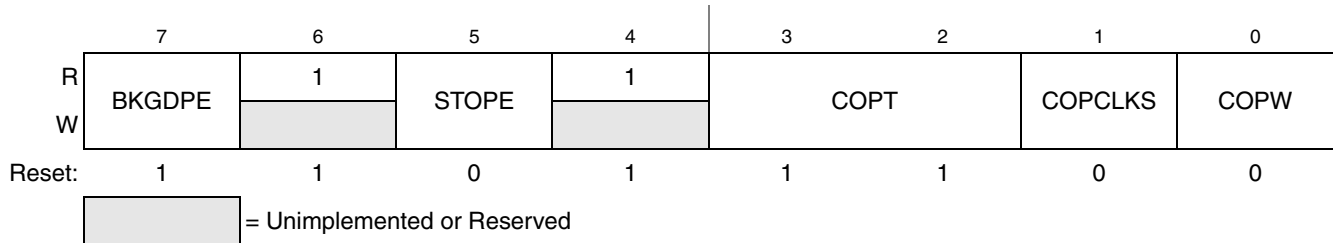
Figure 5-4. System Background Debug Force Reset Register (SBDFR)

Table 5-5. SBDFR Register Field Descriptions

Field	Description
0 BDFR	<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. To enter user mode, PTC6/ACMPO/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTC6/ACMPO/BKGD/MS must be low immediately after issuing WRITE_BYTE command. See the data sheet for more information.

### 5.8.4 System Options Register 1 (SOPT1)

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



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

**Table 5-6. SOPT1 Register Field Descriptions**

Field	Description
7 BKGDPE	<b>Background Pin Enable</b> — When BKGD/MS is shared with general-purpose I/O through chip-level hookup, The BKGDPE bit enables the BKGD/MS pin to function as BKGD. When the BKGDPE bit is clear, the BKGD function is disabled. 0: BKGD pin disabled. 1: BKGD pin enabled.
5 STOPE	<b>Stop Mode Enable</b> — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
3:2 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> .
1 COPCLKS	<b>COP Watchdog Clock Select</b> — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz LPO clock is source to COP. 1 Bus clock is source to COP.
0 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.

### 5.8.5 Internal Peripheral Select Register 1 (SIMIPS1)

The fields in this register control sources used for FTM, MTIM3 and MTIM2 to select their external clock. And one bit in this register used for PCNT to select its sensor type. See [Figure 2-6](#) for an illustration of one control bit for PCNT to select its sensor in action.

The unimplemented bits change its value to 1 when they are written.

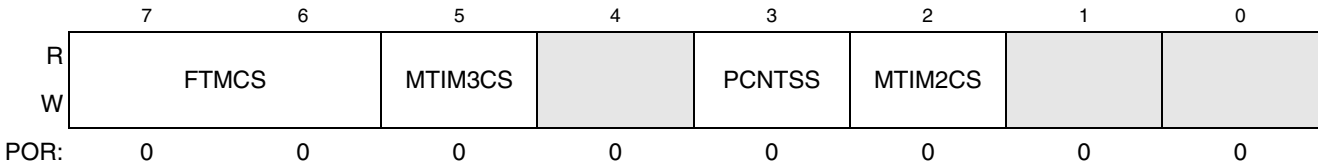


Figure 5-6. Internal Peripheral Select Register 1 (SIMIPS1)

Table 5-7. SIMIPS1 Register Field Descriptions

Field	Description
7:6 FTMCS	<b>FTM External Clock Select</b> — 00 = FTMCLK 01 = MTIMCLK 10 = Reserved 11 = TOF of MTIM1
5 MTIM3CS	<b>MTIM3 External Clock Select</b> — 0 = MTIMCLK 1 = TOF of MTIM1
3 PCNTSS	<b>PCNT Sensor Select</b> — 0 = To select the pin input coming as digital 1 = To select the pin input coming to go through PRACMPx to be converted into Digital
2 MTIM2CS	<b>MTIM2 External Clock Select</b> — 0 = MTIMCLK 1 = TOF of MTIM1



## 5.8.6 Internal Peripheral Select Register 2 (SIMIPS2)

The fields in this register control sources used for two of the SCI Rx1 pins, as well as modulation choices for the SCI Tx1 pins. The various clock sources used for modulation purposes must be enabled/disabled via the appropriate controls elsewhere in the device. See [Figure 2-4](#) for an illustration of these controls in action.

The unimplemented bits change its value to 1 when they are written.

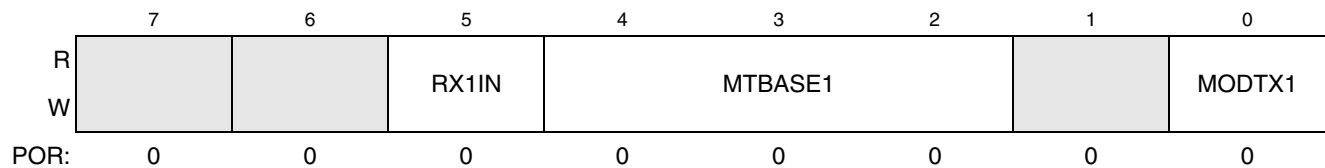


Figure 5-7. Internal Peripheral Select Register 2 (SIMIPS2)

Table 5-8. SIMIPS2 Register Field Descriptions

Field	Description
5 RX1IN	<b>SCI1 Rx Input Pin Select</b> — 0 = Rx1 is fed from the digital input pin (assuming the Rx1 is enabled on that pin via the MC registers) 1 = Rx1 is fed from the output of comparator 0
4-2 MTBASE1	<b>SCI1 Tx Modulation Time Base Select</b> — 000 = FTMCH0 001 = FTMCH1 010 = MTIM1 output 011 = MTIM2 output 100 = MTIM3 output 101 = PCNTCH0 110 = PCNTCH1 111 = Reserved
0 MODTX1	<b>Modulate TxD1</b> — 0 = Do not modulate the output of SCI1 1 = Modulate the output of SCI1 with the timebase selected via the MTBASE1 field

### 5.8.7 Internal Peripheral Select Register 3 (SIMIPS3)

The fields in this register control sources used for two of the SCI Rx2 pins, as well as modulation choices for the SCI Tx2 pins. The various clock sources used for modulation purposes must be enabled/disabled via the appropriate controls elsewhere in the device. See [Figure 2-4](#) for an illustration of these controls in action.

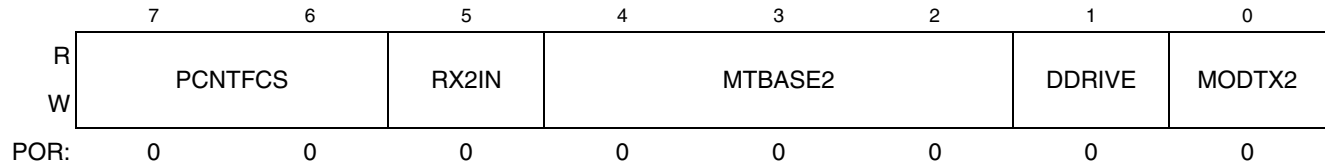


Figure 5-8. Internal Peripheral Select Register 3 (SIMIPS3)

Table 5-9. SIMIPS3 Register Bit Fields

Field	Description
7:6 PCNTFCS	<b>PCNT Filter Clock Select</b> — 000 = OSC1 001 = BUSCLK 010 = IRTC 512Hz clock
5 RX2IN	<b>SCI2 Rx Input Pin Select</b> — 0 = Rx2 is fed from the digital input pin (assuming the Rx2 is enabled on that pin via the MC registers) 1 = Rx2 is fed from the output of comparator 1
4:2 MTBASE2	<b>SCI2 Tx Modulation Time Base Select</b> — 000 = FTMCH0 001 = FTMCH1 010 = MTIM1 output 011 = MTIM2 output 100 = MTIM3 output 101 = PCNTCH0 110 = PCNTCH1 111 = Reserved
1 DDRIVE	<b>Double Current Drive on TxD2 Enable</b> — 0 = Disable double drive on SCI TxD2 pin 1 = Enable double drive on SCI TxD2 pin
0 MODTX2	<b>Modulate TxD2</b> — 0 = Do not modulate the output of SCI2 1 = Modulate the output of SCI2 with the timebase selected via the MTBASE2 field

## 5.8.8 SIM Clock Options Register (SIMCO)

This high-page register controls operation of the CLKOUT pin. The CS field in this register controls the output mux function for the CLKOUT pin. The various clock sources must be enabled/disabled via the appropriate controls elsewhere in the device.

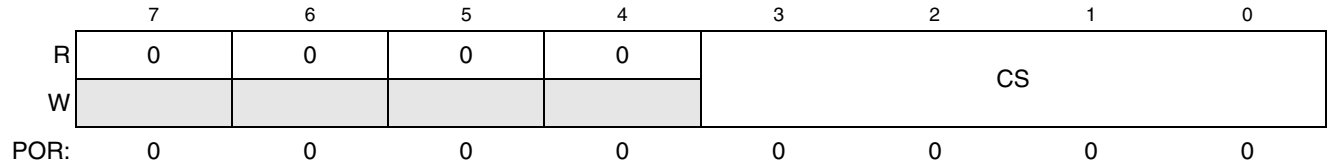


Figure 5-9. System Options Register 6 (SOPT6)

Table 5-10. SOPT6 Register Bit Fields

Field	Description
3-0 CS	<b>Clock Select: These bits select which clock is going to CLKOUT pin —</b> 0000 = OFF 0001 = OSCOUT1 0010 = OSCOUT2 0011 = Internal oscillator 0100 = BUSCLK 0101 = Core Clock 0110 = LPOCLK 1000 = ADC0 asynchronous clock 1001 = ADC1 asynchronous clock Rest options are reserved

5.8.9 Clock Check and Select Control Register (CCSCTRL)

This register is used to control the mux which selects the external clock for the ICS. This function is implemented external to the ICS module itself.

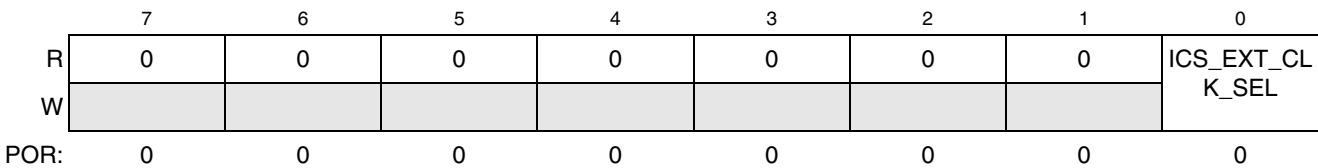


Figure 5-10. Clock Check and Select Control Register (CCSCTRL)

Table 5-11. CCSCTRL Register Bit Fields

Field	Description
0 ICS_EXT_CLK_SEL	<b>ICS External Clock Select: This bit selects the external clock input to the ICS —</b> 0: XOSC2 is selected as the external clock input to the ICS (default) 1: XOSC1 is selected as the external clock input to the ICS <b>Note:</b> This bit should only be changed when the ICS is NOT utilizing the external clock input.

### 5.8.10 CLKOUT Prescaler Register (CLK\_PRSC\_H, CLK\_PRSC\_L)

This register contains the bits of the prescaler counter match value. When this 11 bits value is non-zero, the clock selected by SIMCO is divided by the Prescaler and the final clock is output from CLKOUT pin. When the value is non-zero, the Prescaler is disabled and clock is output without any division.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	Bit 10	Bit 9	Bit 8
W								
POR:	0	0	0	0	0	0	0	0

Figure 5-11. CLKOUT Prescaler Register (CLK\_PRSC\_H)

Table 5-12. CLK\_PRSC\_H Register Bit Fields

Field	Description
2–0 Bit 10–Bit 8	<b>Bit 10–Bit 8:</b> CLKOUT Prescaler match value bit 10 to bit 8

	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:	0	0	0	0	0	0	0	0

Figure 5-12. CLKOUT Prescaler Register (CLK\_PRSC\_L)

Table 5-13. CLK\_PRSC\_L Register Bit Fields

Field	Description
7–0 Bit 7–Bit 0	<b>Bit 7–Bit 0:</b> CLKOUT Prescaler match value bit 7 to bit 0

### 5.8.11 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.

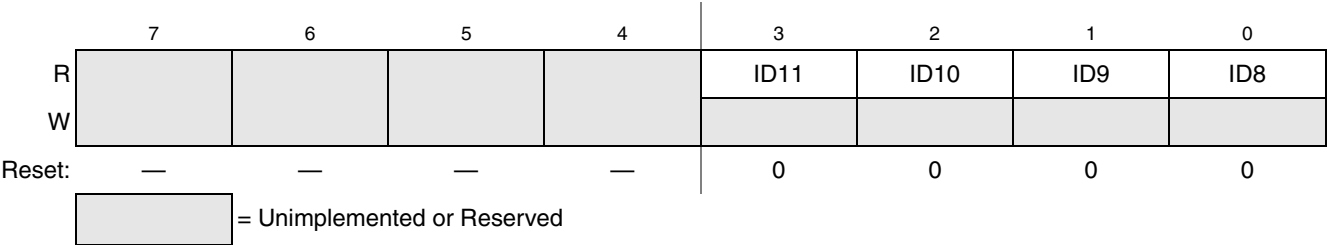


Figure 5-13. System Device Identification Register — High (SDIDH)

Table 5-14. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	<b>Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.</b>
3:0 ID[11:8]	<b>Part Identification Number</b> — Each derivative in the HCS08 family has a unique identification number. The MC9S08GW64 is hard coded to the value 0x030. See also ID bits in <a href="#">Table 5-15</a> .

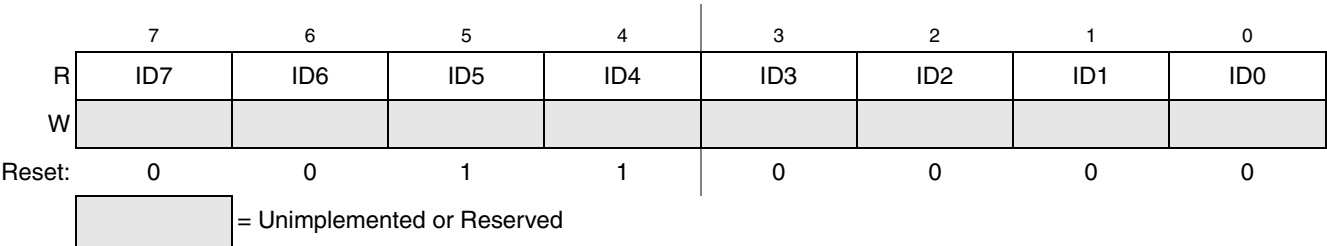


Figure 5-14. System Device Identification Register — Low (SDIDL)

Table 5-15. 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 MC9S08GW64 is hard coded to the value 0x030. See also ID bits in <a href="#">Table 5-14</a> .

## 5.8.12 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 or ACMP modules. The bandgap voltage reference is 1.2 V. To configure the low-voltage detect trip voltage, see [Table 5-18](#) for the LVDV bit description in SPMSC3.

	7	6	5	4	3	2	1	0
R	LVDF	0	LVDIE	LVDRE <sup>1</sup>	LVDSE	LVDE <sup>1</sup>	BGBDS	BGBE
W		LVDACK						
Reset:	0	0	0	1	1	1	0	0
Stop2 wakeup:	u	0	u	u	u	u	0	u

= Unimplemented or Reserved      u = Unaffected by reset

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

**Figure 5-15. System Power Management Status and Control 1 Register (SPMSC1)**

**Table 5-16. SPMSC1 Register Field Descriptions**

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

Table 5-16. SPMSC1 Register Field Descriptions (continued)

Field	Description
1 BGBDS	<b>Bandgap Buffer Drive Select</b> — This bit is used to select the high-drive mode of the bandgap buffer. 0 Bandgap Buffer enabled in low-drive mode if BGBE = 1. Bandgap buffer drive choose VREFO1. 1 Bandgap Buffer enabled in high-drive mode if BGBE = 1. Bandgap buffer drive chooses an internal bandgap signal.
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 is disabled. 1 Bandgap Buffer is enabled.

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

This high-page register contains status and control bits to configure the low power run and wait modes as well as configure the stop mode behavior of the MCU. See [Section 3.3.1, “Low-Power Run Mode \(LPRun\),”](#) [Section 3.5.1, “Low-Power Wait Mode \(LPWait\),”](#) and [Section 3.6, “Stop Modes,”](#) for more information.

	7	6	5	4	3	2	1	0
R	LPR	LPRS	LPWUI	0	PPDF	0	PPDE <sup>1</sup>	PPDC
W						PPDACK		
Reset:	0	0	0	0	0	0	1	0
Stop2 wakeUp:	0	0	u	0	1	0	1	1

= Unimplemented or Reserved
 u= Unaffected by reset

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

Figure 5-16. System Power Management Status and Control 2 Register (SPMSC2)

Table 5-17. SPMSC2 Register Field Descriptions

Field	Description
7 LPR	<b>Low Power Regulator Control</b> — The LPR bit controls entry into the low power run and wait modes in which the voltage regulator is put into standby. This bit cannot be set if PPDC = 1. If PPDC and LPR are set in a single write instruction, only PPDC will actually be set. Automatically cleared when LPWUI is set and an interrupt occurs. 0 Low power run and wait modes are disabled. 1 Low power run and wait modes are enabled.
6 LPRS	<b>Low Power Regulator Status</b> — This read-only status bit indicates that the voltage regulator has entered into standby for the low power run or wait mode. 0 The voltage regulator is not currently in standby. 1 The voltage regulator is currently in standby.
5 LPWUI	<b>Low Power Wake Up on Interrupt</b> — This bit controls whether or not the voltage regulator exits standby when any active MCU interrupt occurs. 0 The voltage regulator will remain in standby on an interrupt. 1 The voltage regulator will exit standby on an interrupt.



**Table 5-17. SPMSC2 Register Field Descriptions (continued)**

Field	Description
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.
1 PPDE	<b>Partial Power Down Enable</b> — The write-once PPDE bit can be used to “lockout” the partial power down mode. 0 Partial power down is not enabled. 1 Partial power down is enabled and controlled via the PPDC bit.
0 PPDC	<b>Partial Power Down Control</b> — The PPDC bit controls which power down mode is selected. This bit cannot be set if LPR = 1. If PPDC and LPR are set in a single write instruction, only PPDC will actually be set. 0 Stop3 low power mode enabled. 1 Stop2 partial power down mode enabled.

### 5.8.14 System Power Management Status and Control 3 Register (SPMSC3)

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

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	LVWIE	0	0	0
W		LVWACK						
POR:	0 <sup>1</sup>	0	0	0	0	0	0	0
LVR:	0 <sup>1</sup>	0	U	U	0	0	0	0
Any other reset:	0 <sup>1</sup>	0	U	U	0	0	0	0

= Unimplemented or Reserved
 U= Unaffected by reset

<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  $V_{LVW}$ .

**Figure 5-17. System Power Management Status and Control 3 Register (SPMSC3)**

**Table 5-18. SPMSC3 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 <b>not</b> 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 LVDV	<b>Low-Voltage Detect Voltage Select</b> — The LVDV bit indicates the LVD trip point voltage ( $V_{LVD}$ ). 0 Low trip point selected ( $V_{LVD} = V_{LVOL}$ ). 1 High trip point selected ( $V_{LVD} = V_{LVOH}$ ).

Table 5-18. SPMSC3 Register Field Descriptions (continued)

Field	Description
4 LVWV	<b>Low-Voltage Detect Voltage Select</b> — The LVWV bit indicates the LVW trip point voltage ( $V_{LVW}$ ). 0 Low trip point selected ( $V_{LVW} = V_{LVWL}$ ). 1 High trip point selected ( $V_{LVW} = V_{LVWH}$ ).
3 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.

Table 5-19. LVD and LVW Trip Point Typical Values<sup>1</sup>

LVDV:LVWV	LVW Trip Point	LVD Trip Point
0:0	$V_{LVWL} = 2.16 \text{ V}$	$V_{LVDL} = 1.82 \text{ V}$
0:1	$V_{LVWH} = 2.46 \text{ V}$	
1:0 <sup>2</sup>	$V_{LVWL} = 2.16 \text{ V}$	$V_{LVDH} = 2.16 \text{ V}$
1:1	$V_{LVWH} = 2.46 \text{ V}$	

<sup>1</sup> See *MC9S08GW64 Series MCU Data Sheet* for minimum and maximum values.

<sup>2</sup> This setting is not recommended.

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

This high-page register contains control bits to enable or disable the bus clock to the SCIx, IIC, KBI and ADCx 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 must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.

	7	6	5	4	3	2	1	0
R	ADC1	ADC0	KBI	IIC	SCI3	SCI2	SCI1	SCI0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 5-18. System Clock Gating Control 1 Register (SCGC1)

Table 5-20. SCGC1 Register Field Descriptions

Field	Description
7 ADC1	<b>ADC1 Clock Gate Control</b> — This bit controls the clock gate to the ADC1 module. 0 Bus clock to the ADC1 module is disabled. 1 Bus clock to the ADC1 module is enabled.
6 ADC0	<b>ADC0 Clock Gate Control</b> — This bit controls the clock gate to the ADC0 module. 0 Bus clock to the SPI2 module is disabled. 1 Bus clock to the SPI2 module is enabled.
5 KBI	<b>KBI Clock Gate Control</b> — This bit controls the clock gate to the KBI module. 0 Bus clock to the KBI module is disabled. 1 Bus clock to the KBI module is enabled.
4 IIC	<b>IIC Clock Gate Control</b> — This bit controls the clock gate to the IIC module. 0 Bus clock to the IIC module is disabled. 1 Bus clock to the IIC module is enabled.
3 SCI3	<b>SCI3 Clock Gate Control</b> — This bit controls the clock gate to the SCI3 module. 0 Bus clock to the SCI3 module is disabled. 1 Bus clock to the SCI3 module is enabled.
2 SCI2	<b>SCI2 Clock Gate Control</b> — This bit controls the clock gate to the SCI2 module. 0 Bus clock to the SCI2 module is disabled. 1 Bus clock to the SCI2 module is enabled.
1 SCI1	<b>SCI1 Clock Gate Control</b> — This bit controls the clock gate to the SCI1 module. 0 Bus clock to the SCI1 module is disabled. 1 Bus clock to the SCI1 module is enabled.
0 SCI0	<b>SCI0 Clock Gate Control</b> — This bit controls the clock gate to the SCI0 module. 0 Bus clock to the SCI0 module is disabled. 1 Bus clock to the SCI0 module is enabled.

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

This high-page register contains control bits to enable or disable the bus clock to the SPIx, LCD, IRQ, VREF and bus clock prescaler (CLKPRE, see [Section 5.8.10, “CLKOUT Prescaler Register \(CLK\\_PRSC\\_H, CLK\\_PRSC\\_L\),”](#) for details) 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 must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.

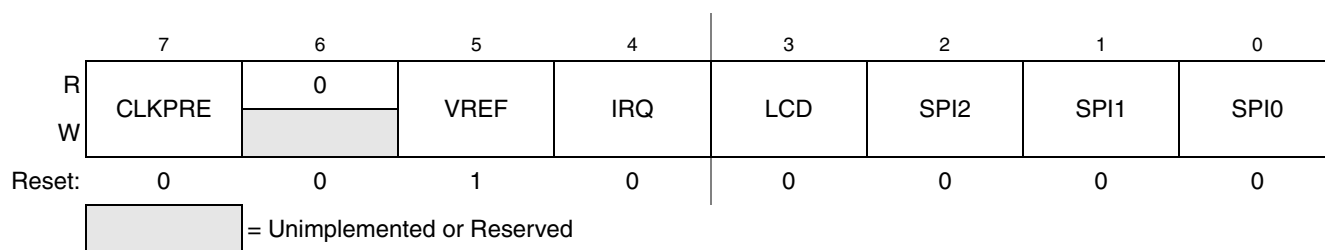


Figure 5-19. System Clock Gating Control 2 Register (SCGC2)

Table 5-21. SCGC2 Register Field Descriptions

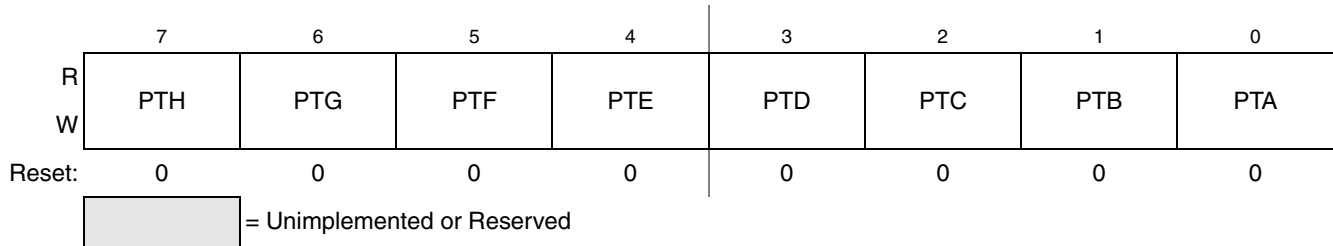
Field	Description
7 CLKPRE	<b>CLKPRE Clock Gate Control</b> — This bit controls the bus clock gate to the CLKPRE module. 0 Bus clock to the CLKPRE module is disabled. 1 Bus clock to the CLKPRE module is enabled.
5 VREF	<b>VREF Clock Gate Control</b> — This bit controls the clock gate to the VREF module. 0 Bus clock to the VREF module is disabled. 1 Bus clock to the VREF module is enabled.
4 IRQ	<b>IRQ Clock Gate Control</b> — This bit controls the clock gate to both of the IRQ modules. 0 Bus clock to the IRQ modules is disabled. 1 Bus clock to the IRQ modules is enabled.
3 LCD	<b>LCD Clock Gate Control</b> — This bit controls the bus clock gate to the LCD module. 0 Bus clock to the LCD module is disabled. 1 Bus clock to the LCD module is enabled.
2 SPI2	<b>SPI2 Clock Gate Control</b> — This bit controls the bus clock gate to the SPI2 module. 0 Bus clock to the SPI2 module is disabled. 1 Bus clock to the SPI2 module is enabled.
1 SPI1	<b>SPI1 Clock Gate Control</b> — This bit controls the clock gate to the SPI1 module. 0 Bus clock to the SPI1 module is disabled. 1 Bus clock to the SPI1 module is enabled.
0 SPI0	<b>SPI0 Clock Gate Control</b> — This bit controls the clock gate to the SPI0 module. 0 Bus clock to the SPI0 module is disabled. 1 Bus clock to the SPI0 module is enabled.

### 5.8.17 System Clock Gating Control 3 Register (SCGC3)

This high-page register contains control bits to enable or disable the bus clock to the PTx 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 must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.



**Figure 5-20. System Clock Gating Control 2 Register (SCGC3)**

**Table 5-22. SCGC3 Register Field Descriptions**

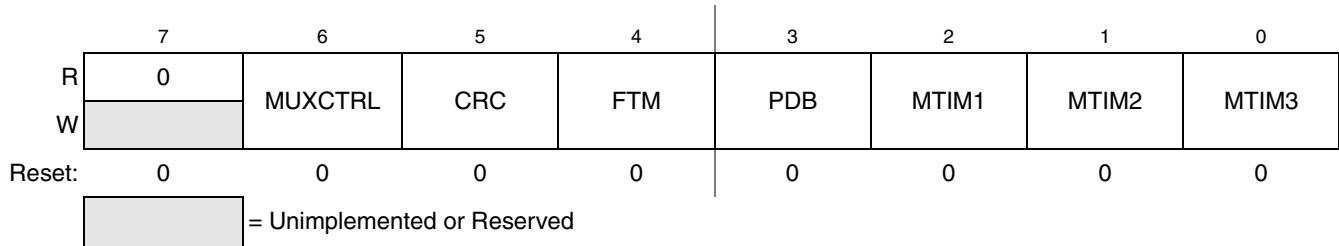
Field	Description
7 PTH	<b>PTH Clock Gate Control</b> — This bit controls the bus clock gate to the PTH module. 0 Bus clock to the PTH module is disabled. 1 Bus clock to the PTH module is enabled.
6 PTG	<b>PTG Clock Gate Control</b> — This bit controls the bus clock gate to the PTG module. 0 Bus clock to the PTG module is disabled. 1 Bus clock to the PTG module is enabled.
5 PTF	<b>PTF Clock Gate Control</b> — This bit controls the clock gate to the PTF module. 0 Bus clock to the PTF module is disabled. 1 Bus clock to the PTF module is enabled.
4 PTE	<b>PTE Clock Gate Control</b> — This bit controls the clock gate to both of the PTE modules. 0 Bus clock to the PTE modules is disabled. 1 Bus clock to the PTE modules is enabled.
3 PTD	<b>PTD Clock Gate Control</b> — This bit controls the bus clock gate to the PTD module. 0 Bus clock to the PTD module is disabled. 1 Bus clock to the PTD module is enabled.
2 PTC	<b>PTC Clock Gate Control</b> — This bit controls the bus clock gate to the PTC module. 0 Bus clock to the PTC module is disabled. 1 Bus clock to the PTC module is enabled.
1 PTB	<b>PTB Clock Gate Control</b> — This bit controls the clock gate to the PTB module. 0 Bus clock to the PTB module is disabled. 1 Bus clock to the PTB module is enabled.
0 PTA	<b>PTA Clock Gate Control</b> — This bit controls the clock gate to the PTA module. 0 Bus clock to the PTA module is disabled. 1 Bus clock to the PTA module is enabled.

### 5.8.18 System Clock Gating Control 4 Register (SCGC4)

This high-page register contains control bits to enable or disable the bus clock to the PIN MUXING, CRC, FTM, PDB and MTIMx 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. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.



**Figure 5-21. System Clock Gating Control 4 Register (SCGC4)**

**Table 5-23. SCGC4 Register Field Descriptions**

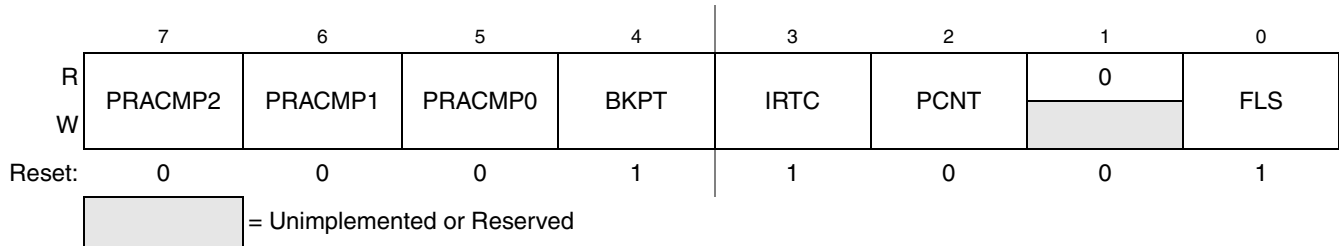
Field	Description
6 MUXCTRL	<b>MUX Control Logic Clock Gate Control</b> — This bit controls the bus clock gate to the MUX Control Logic module. 0 Bus clock to the MUX Control Logic module is disabled. 1 Bus clock to the MUX Control Logic module is enabled.
5 CRC	<b>CRC Clock Gate Control</b> — This bit controls the bus clock gate to the CRC module. 0 Bus clock to the CRC module is disabled. 1 Bus clock to the CRC module is enabled.
4 FTM	<b>FTM Clock Gate Control</b> — This bit controls the bus clock gate to the FTM module. 0 Bus clock to the FTM module is disabled. 1 Bus clock to the FTM module is enabled.
3 PDB	<b>PDB Clock Gate Control</b> — This bit controls the bus clock gate to the PDB module. 0 Bus clock to the PDB module is disabled. 1 Bus clock to the PDB module is enabled.
2 MTIM1	<b>MTIM1 Clock Gate Control</b> — This bit controls the bus clock gate to the MTIM1 module. 0 Bus clock to the MTIM1 module is disabled. 1 Bus clock to the MTIM1 module is enabled.
1 MTIM2	<b>MTIM2 Clock Gate Control</b> — This bit controls the bus clock gate to the MTIM2 module. 0 Bus clock to the MTIM2 module is disabled. 1 Bus clock to the MTIM2 module is enabled.
0 MTIM3	<b>MTIM3 Clock Gate Control</b> — This bit controls the bus clock gate to the MTIM3 module. 0 Bus clock to the MTIM3 module is disabled. 1 Bus clock to the MTIM3 module is enabled.

### 5.8.19 System Clock Gating Control 5 Register (SCGC5)

This high-page register contains control bits to enable or disable the bus clock to the PRACMPx, BKPT, IRTC, PCNT and FLS 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. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.



**Figure 5-22. System Clock Gating Control 5 Register (SCGC5)**

**Table 5-24. SCGC5 Register Field Descriptions**

Field	Description
7 PRACMP2	<b>PRACMP2 Clock Gate Control</b> — This bit controls the bus clock gate to the PRACMP2 module. 0 Bus clock to the PRACMP2 module is disabled. 1 Bus clock to the PRACMP2 module is enabled.
6 PRACMP1	<b>PRACMP1 Clock Gate Control</b> — This bit controls the bus clock gate to the PRACMP1 module. 0 Bus clock to the PRACMP1 module is disabled. 1 Bus clock to the PRACMP1 module is enabled.
5 PRACMP0	<b>PRACMP0 Clock Gate Control</b> — This bit controls the bus clock gate to the PRACMP0 module. 0 Bus clock to the PRACMP0 module is disabled. 1 Bus clock to the PRACMP0 module is enabled.
4 BKPT	<b>BKPT Clock Gate Control</b> — This bit controls the bus clock gate to the BKPT module. 0 Bus clock to the BKPT module is disabled. 1 Bus clock to the BKPT module is enabled.
3 IRTC	<b>IRTC Clock Gate Control</b> — This bit controls the bus clock gate to the IRTC module. 0 Bus clock to the IRTC module is disabled. 1 Bus clock to the IRTC module is enabled.
2 PCNT	<b>PCNT Clock Gate Control</b> — This bit controls the bus clock gate to the PCNT module. 0 Bus clock to the PCNT module is disabled. 1 Bus clock to the PCNT module is enabled.
0 FLS	<b>FLASH Clock Gate Control</b> — This bit controls the bus clock gate to the FLASH module. 0 Bus clock to the FLASH module is disabled. 1 Bus clock to the FLASH module is enabled.





## Chapter 6

# Parallel Input/Output Control

### 6.1 Introduction

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9S08GW64 has eight parallel I/O ports which include a total of 57 I/O pins, including one 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 may be disabled.

After reset, the shared peripheral functions are disabled and the pins are configured as inputs (PTxDDn = 0). The pin control functions for each pin are configured as follows: slew rate control enabled (PTxSEn = 1), low drive strength selected (PTxDSn = 0), and internal pullups disabled (PTxPEn = 0).

Pins that have shared function with the LCD have special behavior based on the state of the VSUPPLY bits in the LCDSUPPLY register. These pins (PTD, PTE and PTA[5:4]) can operate as full complementary drive or open drain drive depending on the VSUPPLY bits. When  $V_{LL3}$  is connected to  $V_{DD}$  externally, VSUPPLY = 11, FCDEN = 1, and RVEN = 0, the pins operate as full complementary drive. For all other VSUPPLY modes, the LCD/GPIO will operate as open drain.

#### 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 pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

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

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

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

Write to the port data register before changing the direction of a port pin so it becomes an output. This ensures that the pin are not driven momentarily with an old data value that happened to be in the port data register.

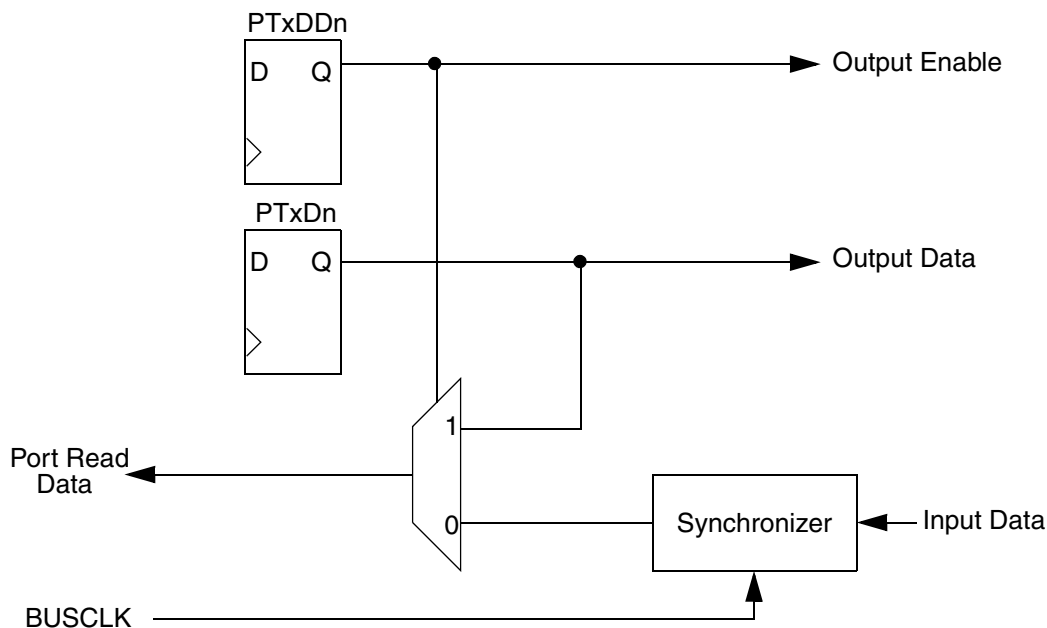


Figure 6-1. Parallel I/O Block Diagram

## 6.3 Pullup, Slew Rate, and Drive Strength

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

### 6.3.1 Port Internal Pullup Enable

For all GPIO, an internal pullup resistor can be enabled for each port pin by setting the corresponding bit in the pullup enable register ( $PTxPEN$ ). Typically, GPIO internal pullups are disabled when in output mode. However, for GPIO that are muxed with LCD pins, the internal pullup is not disabled when in open drain, output mode.

The pullup device is disabled if the pin is controlled by an analog function or any shared peripheral function regardless of the state of the corresponding pullup enable register bit.

### 6.3.2 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.

### 6.3.3 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.

## 6.4 Open Drain Operation

For most cases, port pins that share functions with the LCD operate as open drain outputs. As an open drain output, the output high of the pin is dependent upon the pullup resistor. The pullup resistor can be an internal resistor enabled by the PTxPEx bit or an external resistor.

- The value of the internal resistor can be in the range of 17.5 to 52.5 k $\Omega$
- The value of an external resistor must be carefully selected to ensure it support the output loads that are being driven.

Pins that have shared function with the LCD have special behavior based on the state of the VSUPPLY bits in the LCDSUPPLY register. These pins can operate as full complementary drive or open drain drive depending on the VSUPPLY bits. When VLL3 is connected to V<sub>DD</sub> externally, VSUPPLY = 11, FCDEN = 1 and RVEN = 0, the pins operate as full complementary drive. For all other VSUPPLY modes the LCD/GPIO operates as open drain.

## 6.5 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 pre-STOP instruction state. CPU register status and the state of I/O registers is 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 must 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 are 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.
- If the LCD module is configured to operate in stop modes, the drive mode of the GPIO shared with LCD will be retained upon stop recovery.

- 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.

## 6.6 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 direct-page of the memory map. The pullup, slew rate, drive strength, input filter and interrupt control registers are located in the high-page section of the memory map.

Refer to tables 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 to translate these names into the appropriate absolute addresses.

An internal pullup device can be enabled for each port pin by setting the corresponding bit in the pullup enable register (PTxPE[n]). The pullup device is disabled if the pin is either:

- Configured as an output by the parallel I/O control logic
- Configured as a shared peripheral function
- Controlled by an analog function.
- At reset, except for  $\overline{\text{RESET}}$  and BKGD/MS.

Slew rate control can be enabled for each port pin by setting the corresponding bit in the slew rate control register (PTxSE[n]). 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.

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PTxDS[n]). 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, 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.

The pad cells on this device incorporate optional low pass filters on the digital input functions. These are enabled by setting the appropriate bit in the input filter enable register (PTxIFE[n]). When set, a low pass filter (10 MHz to 30 MHz bandwidth) is enabled in the logic input path. When cleared, the filter is bypassed.

[Table 6-1](#) shows the fixed control signals for some dedicated pins.

**Table 6-1. Fixed Control Signal for Some Pins**

	PE	SE	DE	IFE
RESET	1	0	1	1
TAMPER1/2	NA	NA	NA	NA
BKGD/MS	1	0	1	1

Table 6-1. Fixed Control Signal for Some Pins

	PE	SE	DE	IFE
IRQ	From IRQ module	NA	NA	1

## 6.6.1 Port A Registers

Port A is controlled by the registers listed below.

The pin PTA6 is unique. PTA6 is an output only, so the control bits for the input functions will not have any effect on this pin.

### 6.6.1.1 Port A Data Register (PTAD)

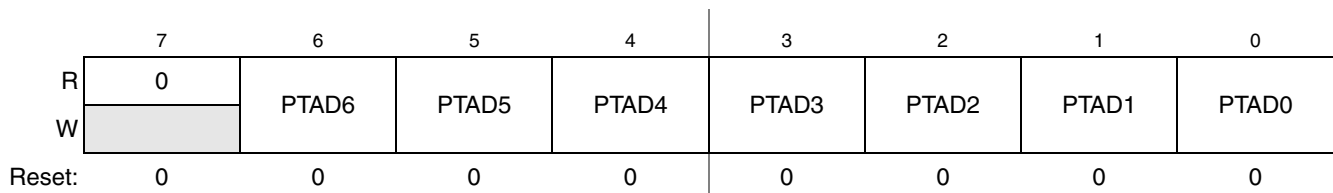
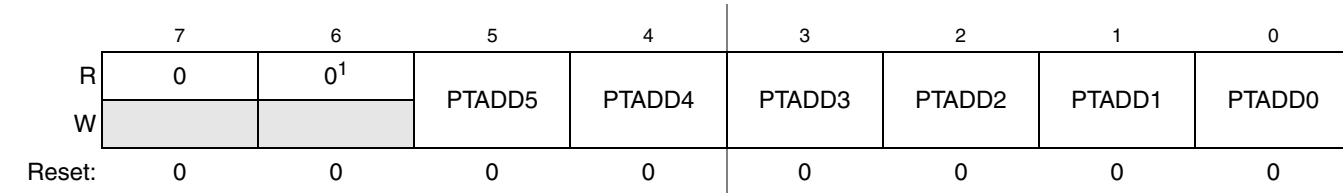


Figure 6-2. Port A Data Register (PTAD)

Table 6-2. PTAD Register Field Descriptions

Field	Description
6:0 PTAD[6:0]	<p><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.</p> <p>Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups/pulldowns disabled.</p>

6.6.1.2 Port A Data Direction Register (PTADD)



<sup>1</sup> Writing to this bit has no affect on the output-only PTA6 pin.

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

Table 6-3. PTADD Register Field Descriptions

Field	Description
5:0 PTADD[5:0]	<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.

### 6.6.1.3 Port A Pull Enable Register (PTAPE)

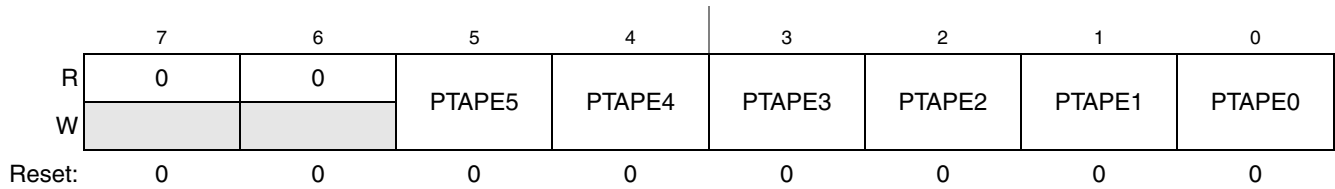


Figure 6-4. Internal Pull Enable for Port A Register (PTAPE)

Table 6-4. PTAPE Register Field Descriptions

Field	Description
5:0 PTAPE[5:0]	<b>Internal Pull Enable for Port A Bits</b> — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTA pin. For port A pins that are configured as outputs (except for PTA4 and PTA5), these bits have no effect and the internal pull devices are disabled. 0 Internal pullup/pulldown device disabled for port A bit n. 1 Internal pullup/pulldown device enabled for port A bit n.

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

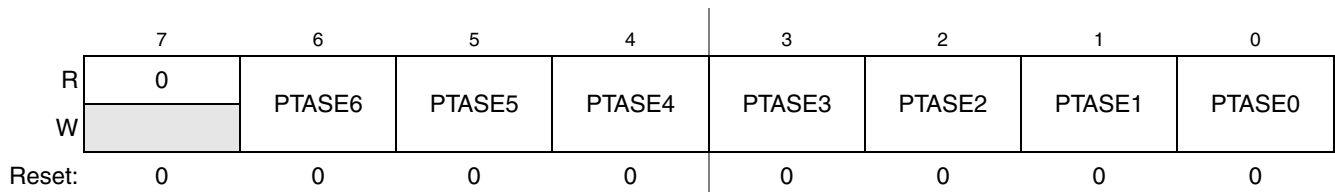


Figure 6-5. Slew Rate Enable for Port A Register (PTASE)

Table 6-5. PTASE Register Field Descriptions

Field	Description
6:0 PTASE[6:0]	<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.

6.6.1.5 Port A Drive Strength Selection Register (PTADS)

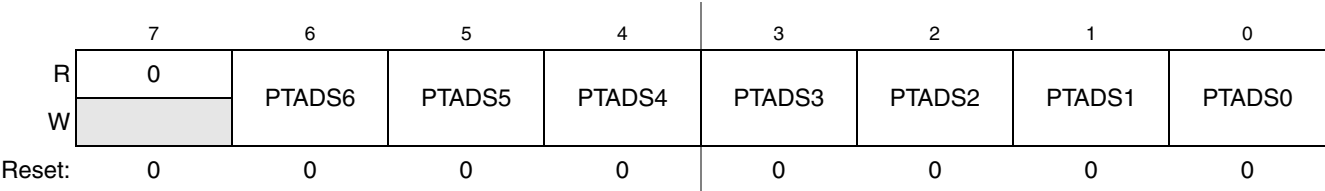


Figure 6-6. Drive Strength Selection for Port A Register (PTADS)

Table 6-6. PTADS Register Field Descriptions

Field	Description
6:0 PTADS[6:0]	<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.

6.6.1.6 Port A Input Filter Enable Register (PTAIFE)

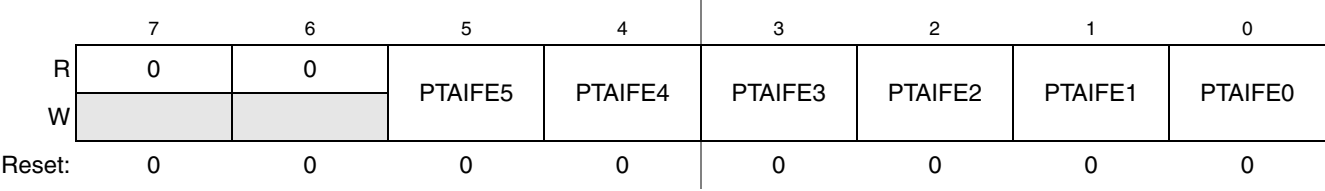


Figure 6-7. Port A Input Filter Enable Register (PTAIFE)

Table 6-7. PTAIFE Field Descriptions

Field	Description
5:0 PTAIFE[5:0]	<b>Port A Input Filter Enable</b> — Input low-pass filter enable control bits for PTA pins. 0 Input filter disabled 1 Input filter enabled



## 6.6.2 Port B Registers

Port B is controlled by the registers listed below.

### 6.6.2.1 Port B Data Register (PTBD)

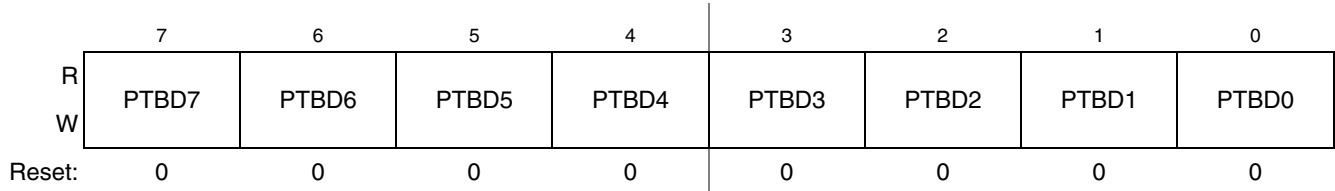


Figure 6-8. Port B Data Register (PTBD)

Table 6-8. 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 pullups/pulldowns disabled.

### 6.6.2.2 Port B Data Direction Register (PTBDD)

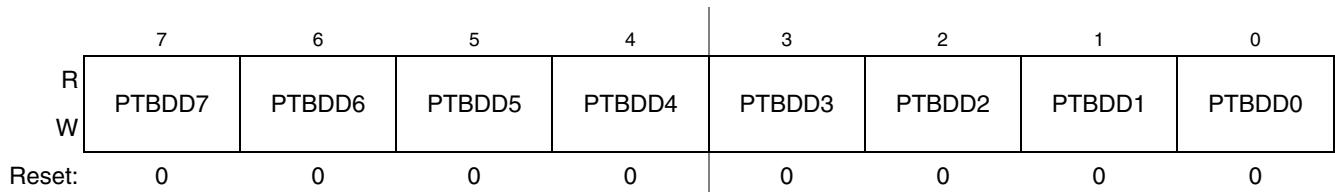


Figure 6-9. Port B Data Direction Register (PTBDD)

Table 6-9. 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.

### 6.6.2.3 Port B Pull Enable Register (PTBPE)

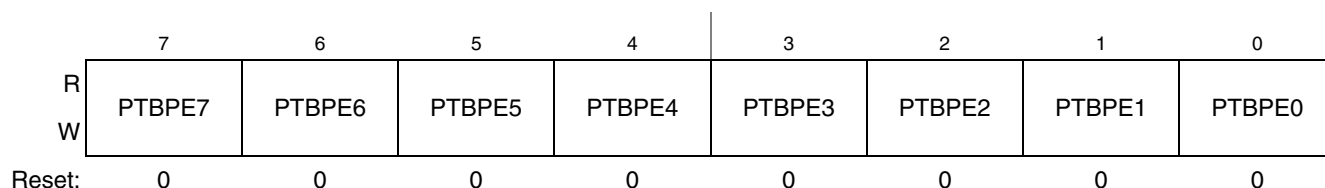


Figure 6-10. Internal Pull Enable for Port B Register (PTBPE)

Table 6-10. 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 pullup or pulldown 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 pullup/pulldown device disabled for port B bit n. 1 Internal pullup/pulldown device enabled for port B bit n.

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

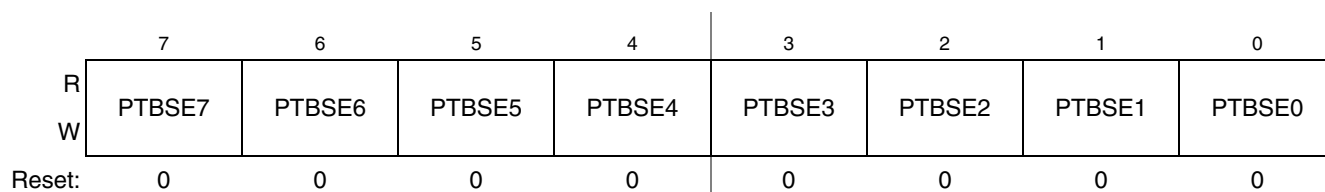


Figure 6-11. Slew Rate Enable for Port B Register (PTBSE)

Table 6-11. 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.

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

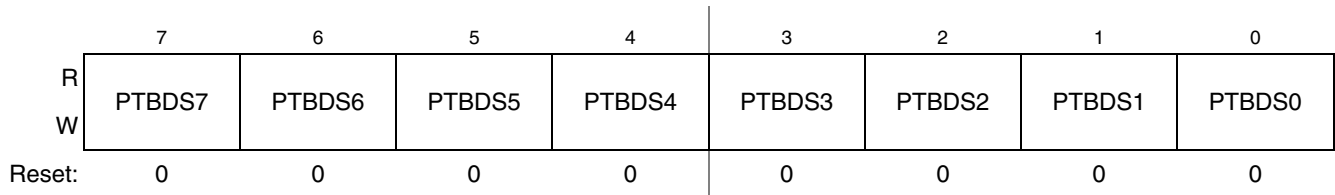


Figure 6-12. Drive Strength Selection for Port B Register (PTBDS)

Table 6-12. 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.

### 6.6.2.6 Port B Input Filter Enable Register (PTBIFE)

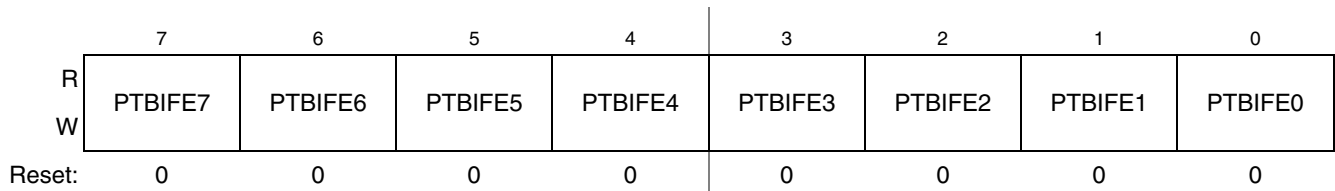


Figure 6-13. Port B Input Filter Enable Register (PTBIFE)

Table 6-13. PTBIFE Field Descriptions

Field	Description
7:0 PTBIFE[7:0]	<b>Port B Input Filter Enable</b> — Input low-pass filter enable control bits for PTB pins. 0 Input filter disabled 1 Input filter enabled

### 6.6.3 Port C Registers

Port C is controlled by the registers listed below.

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

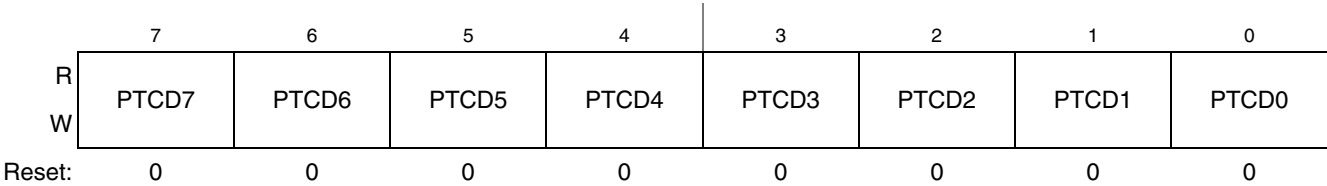


Figure 6-14. Port C Data Register (PTCD)

Table 6-14. 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 of the corresponding MCU pin. Reset forces PTCD to all 0s, but these 0s are not driven out of the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

#### 6.6.3.2 Port C Data Direction Register (PTCDD)

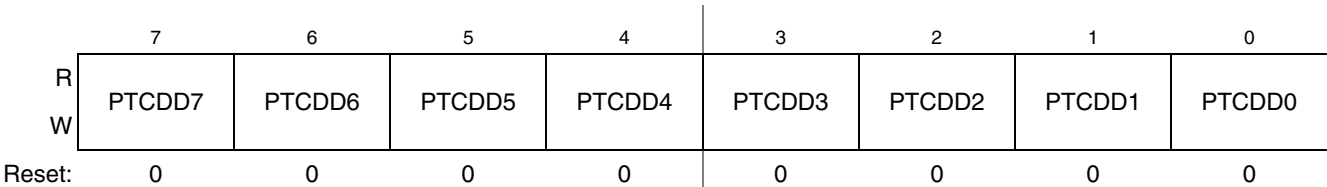


Figure 6-15. Port C Data Direction Register (PTCDD)

Table 6-15. 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.

### 6.6.3.3 Port C Pull Enable Register (PTCPE)

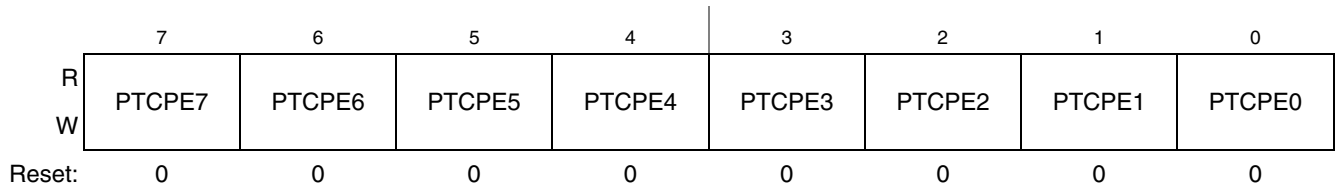


Figure 6-16. Internal Pull Enable for Port C Register (PTCPE)

Table 6-16. 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 pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pullup device disabled for port C bit n. 1 Internal pullup device enabled for port C bit n.

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

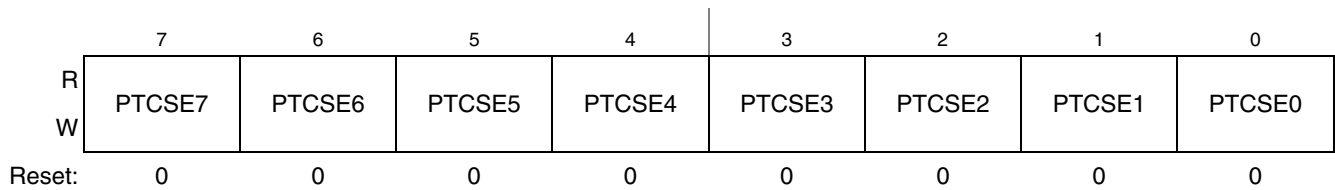


Figure 6-17. Slew Rate Enable for Port C Register (PTCSE)

Table 6-17. 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.

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

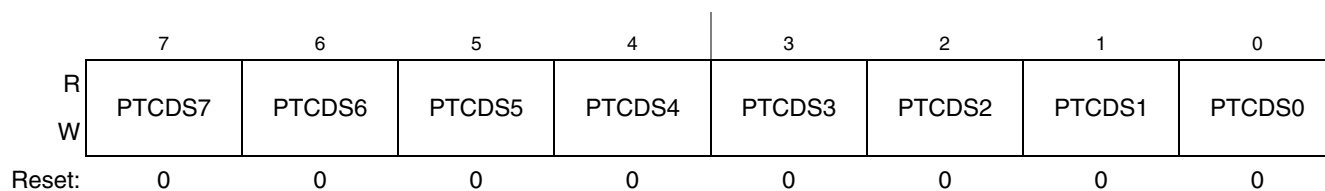


Figure 6-18. Drive Strength Selection for Port C Register (PTCDS)

Table 6-18. 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.

### 6.6.3.6 Port C Input Filter Enable Register (PTCIFE)

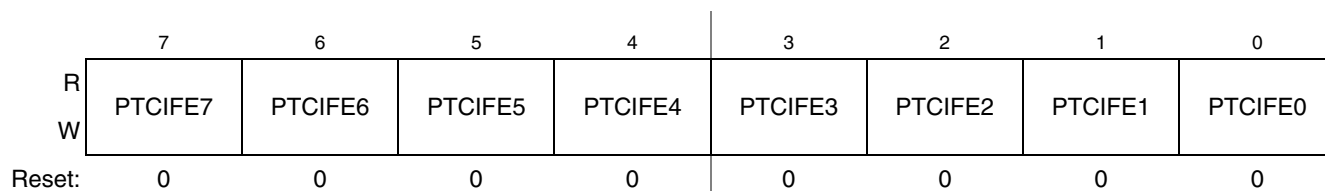


Figure 6-19. Port C Input Filter Enable Register (PTCIFE)

Table 6-19. PTCIFE Field Descriptions

Field	Description
7:0 PTCIFE[7:0]	<b>Port C Input Filter Enable</b> — Input low-pass filter enable control bits for PTC pins. 0 Input filter disabled 1 Input filter enabled

## 6.6.4 Port D Registers

Port D is controlled by the registers listed below.

### 6.6.4.1 Port D Data Register (PTDD)

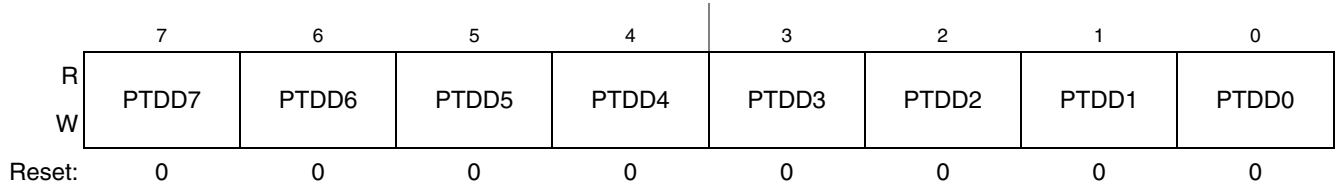


Figure 6-20. Port D Data Register (PTDD)

Table 6-20. 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 pullups/pulldowns disabled.

### 6.6.4.2 Port D Data Direction Register (PTDDD)



Figure 6-21. Port D Data Direction Register (PTDDD)

Table 6-21. 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.

### 6.6.4.3 Port D Pull Enable Register (PTDPE)

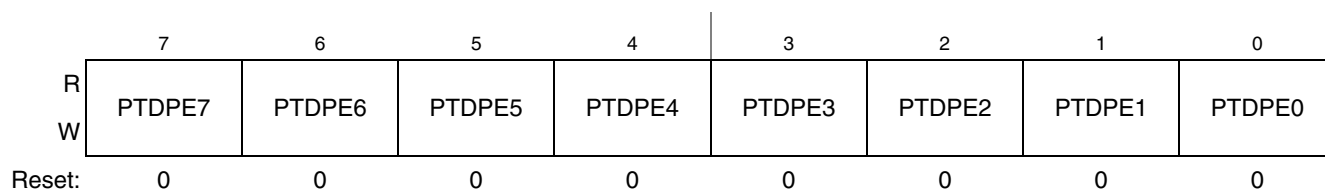


Figure 6-22. Internal Pull Enable for Port D Register (PTDPE)

Table 6-22. 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 pullup or pulldown device is enabled for the associated PTD pin. 0 Internal pullup/pulldown device disabled for port D bit n. 1 Internal pullup/pulldown device enabled for port D bit n.

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

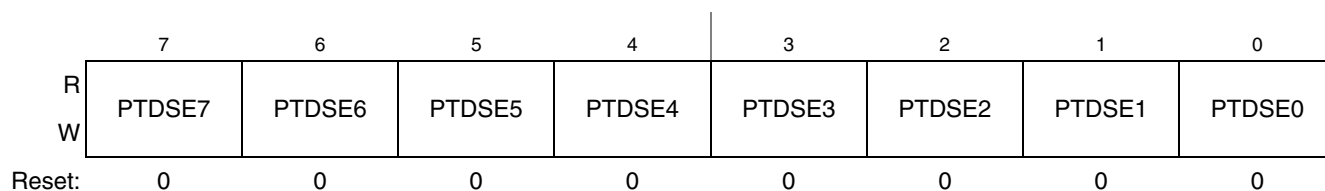


Figure 6-23. Slew Rate Enable for Port D Register (PTDSE)

Table 6-23. 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. 0 Output slew rate control disabled for port D bit n. 1 Output slew rate control enabled for port D bit n.



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

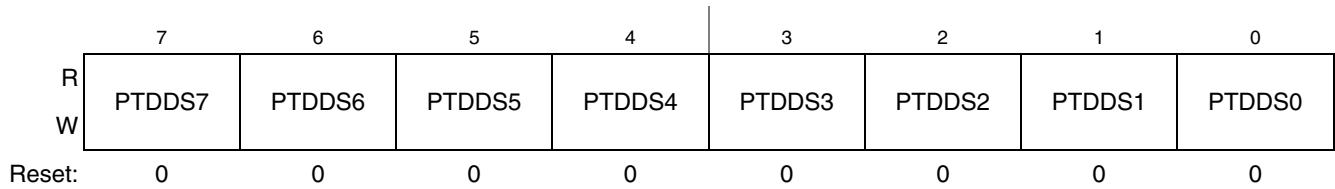


Figure 6-24. Drive Strength Selection for Port D Register (PTDDS)

Table 6-24. 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.

### 6.6.4.6 Port D Input Filter Enable Register (PTDIFE)

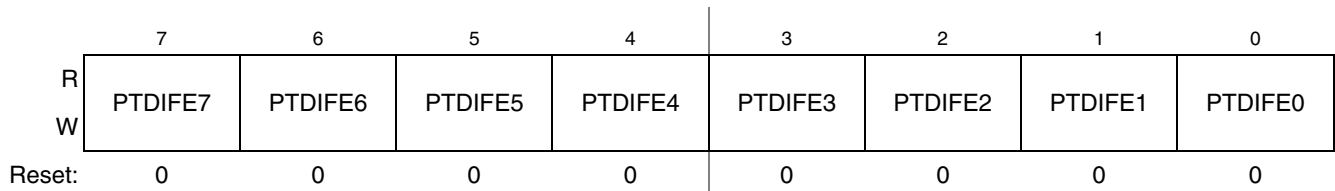


Figure 6-25. Port D Input Filter Enable Register (PTDIFE)

Table 6-25. PTDIFE Field Descriptions

Field	Description
7:0 PTDIFE[7:0]	<b>Port D Input Filter Enable</b> — Input low-pass filter enable control bits for PTD pins. 0 Input filter disabled 1 Input filter enabled

### 6.6.5 Port E Registers

Port E is controlled by the registers listed below.

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

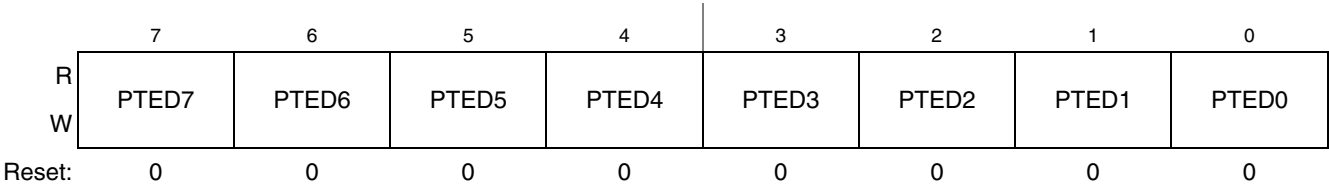


Figure 6-26. Port E Data Register (PTED)

Table 6-26. PTED Register Field Descriptions

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

#### 6.6.5.2 Port E Data Direction Register (PTEDD)

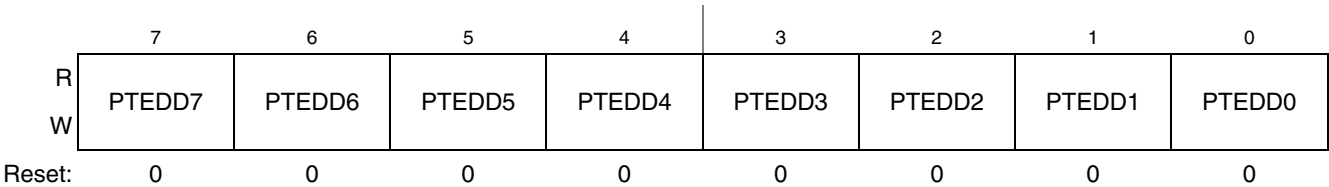


Figure 6-27. Port E Data Direction Register (PTEDD)

Table 6-27. PTEDD Register Field Descriptions

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

### 6.6.5.3 Port E Pull Enable Register (PTEPE)

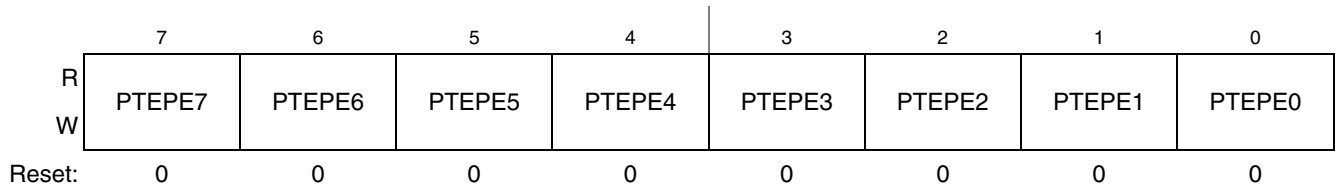


Figure 6-28. Internal Pull Enable for Port E Register (PTEPE)

Table 6-28. PTEPE Register Field Descriptions

Field	Description
7:0 PTEPE[7:0]	<b>Internal Pull Enable for Port E Bits</b> — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTE pin. 0 Internal pullup/pulldown device disabled for port E bit n. 1 Internal pullup/pulldown device enabled for port E bit n.

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

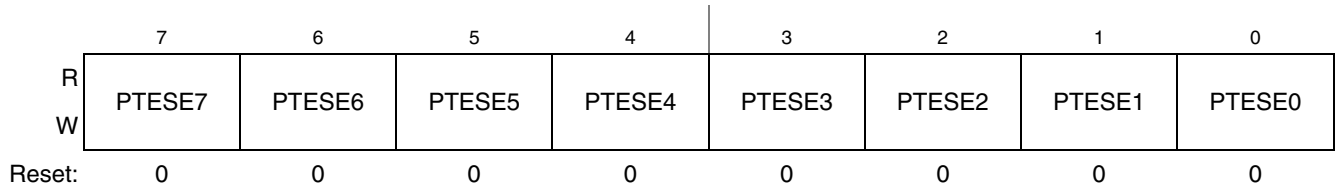


Figure 6-29. Slew Rate Enable for Port E Register (PTESE)

Table 6-29. PTESE Register Field Descriptions

Field	Description
7:0 PTESE[7:0]	<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. 0 Output slew rate control disabled for port E bit n. 1 Output slew rate control enabled for port E bit n.

### 6.6.5.5 Port E Input Filter Enable Register (PTEIFE)

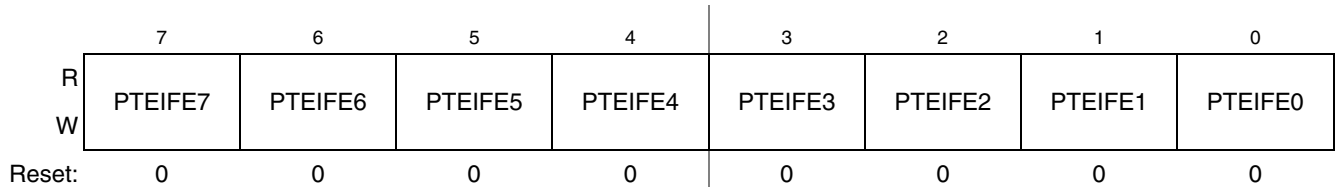


Figure 6-30. Port E Input Filter Enable Register (PTEIFE)

Table 6-30. PTEIFE Field Descriptions

Field	Description
7:0 PTEIFE[7:0]	<b>Port E Input Filter Enable</b> — Input low-pass filter enable control bits for PTE pins. 0 Input filter disabled 1 Input filter enabled

## 6.6.6 Port F Registers

Port F is controlled by the registers listed below.

### 6.6.6.1 Port F Data Register (PTFD)

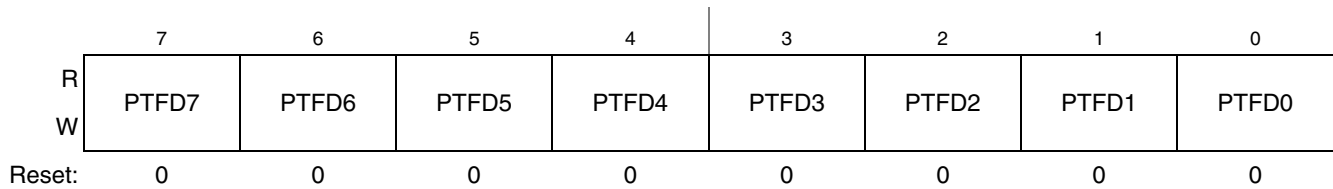


Figure 6-31. Port F Data Register (PTFD)

Table 6-31. PTFD Register Field Descriptions

Field	Description
7:0 PTFD[7:0]	<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 pullups/pulldowns disabled.

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

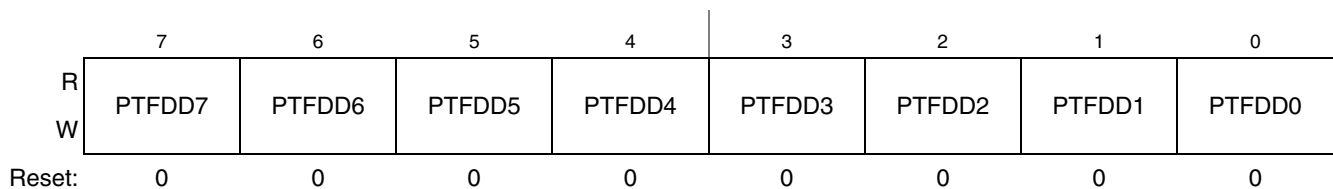


Figure 6-32. Port F Data Direction Register (PTFDD)

Table 6-32. PTFDD Register Field Descriptions

Field	Description
7:0 PTFDD[7:0]	<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.

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

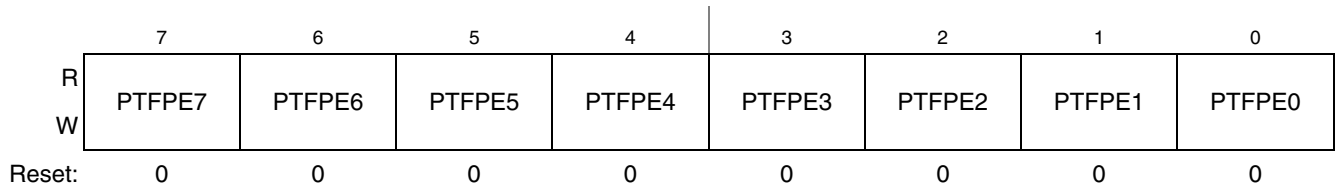


Figure 6-33. Internal Pull Enable for Port F Register (PTFPE)

Table 6-33. PTFPE Register Field Descriptions

Field	Description
7:0 PTFPE[7:0]	<b>Internal Pull Enable for Port F Bits</b> — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTF pin. 0 Internal pullup/pulldown device disabled for port F bit n. 1 Internal pullup/pulldown device enabled for port F bit n.

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

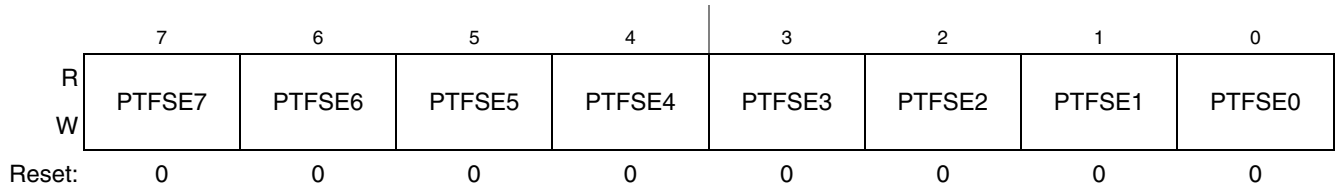


Figure 6-34. Slew Rate Enable for Port F Register (PTFSE)

Table 6-34. PTFSE Register Field Descriptions

Field	Description
7:0 PTFSE[7:0]	<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. 0 Output slew rate control disabled for port F bit n. 1 Output slew rate control enabled for port F bit n.

### 6.6.6.5 Port F Input Filter Enable Register (PTFIFE)

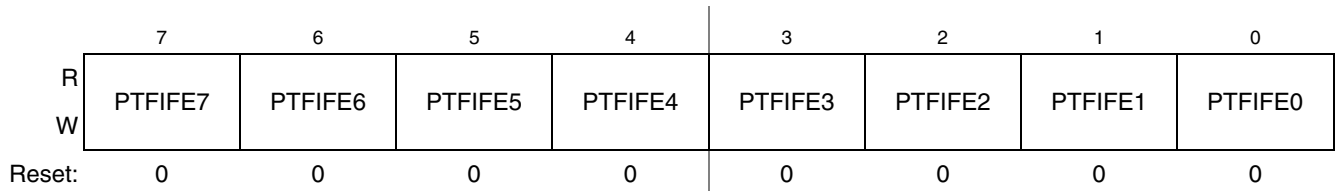


Figure 6-35. Port F Input Filter Enable Register (PTFIFE)

Table 6-35. PTFIFE Field Descriptions

Field	Description
7:0 PTFIFE[7:0]	<b>Port F Input Filter Enable</b> — Input low-pass filter enable control bits for PTF pins. 0 Input filter disabled 1 Input filter enabled

## 6.6.7 Port G Registers

Port G is controlled by the registers listed below.

### 6.6.7.1 Port G Data Register (PTGD)

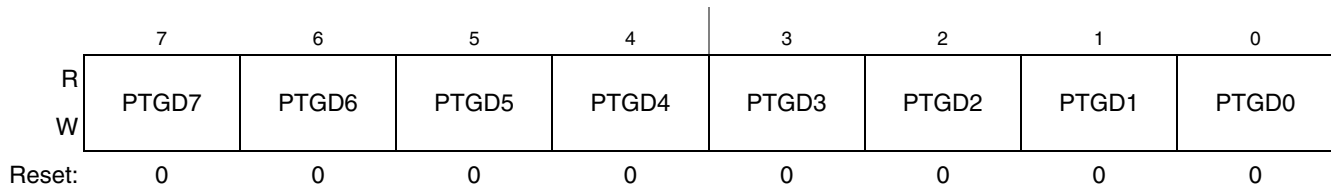


Figure 6-36. Port G Data Register (PTGD)

Table 6-36. PTGD Register Field Descriptions

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

### 6.6.7.2 Port G Data Direction Register (PTGDD)

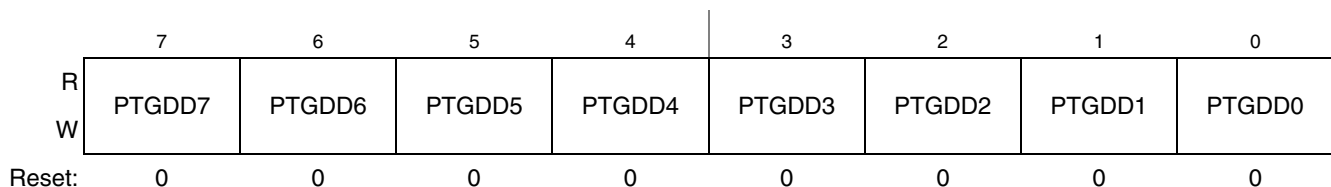


Figure 6-37. Port G Data Direction Register (PTGDD)

Table 6-37. PTGDD Register Field Descriptions

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

### 6.6.7.3 Port G Pull Enable Register (PTGPE)

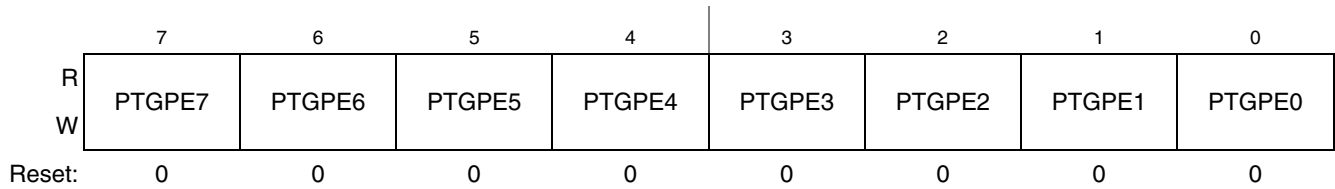


Figure 6-38. Internal Pull Enable for Port G Register (PTGPE)

Table 6-38. PTGPE Register Field Descriptions

Field	Description
7:0 PTGPE[7:0]	<b>Internal Pull Enable for Port G Bits</b> — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTG pin. 0 Internal pullup/pulldown device disabled for port G bit n. 1 Internal pullup/pulldown device enabled for port G bit n.

### 6.6.7.4 Port G Slew Rate Enable Register (PTGSE)

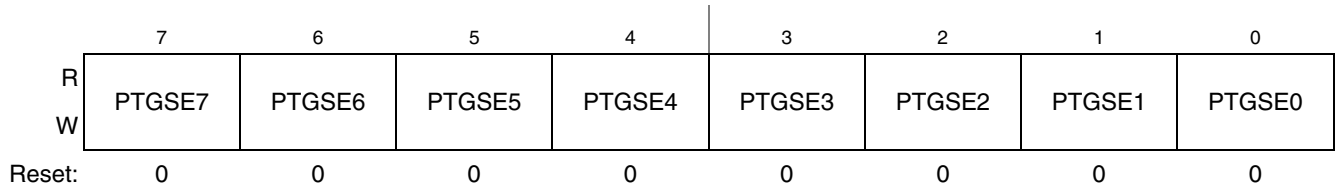


Figure 6-39. Slew Rate Enable for Port G Register (PTGSE)

Table 6-39. PTGSE Register Field Descriptions

Field	Description
7:0 PTGSE[7:0]	<b>Output Slew Rate Enable for Port G Bits</b> — Each of these control bits determines if the output slew rate control is enabled for the associated PTG pin. 0 Output slew rate control disabled for port G bit n. 1 Output slew rate control enabled for port G bit n.

### 6.6.7.5 Port G Input Filter Enable Register (PTGIFE)

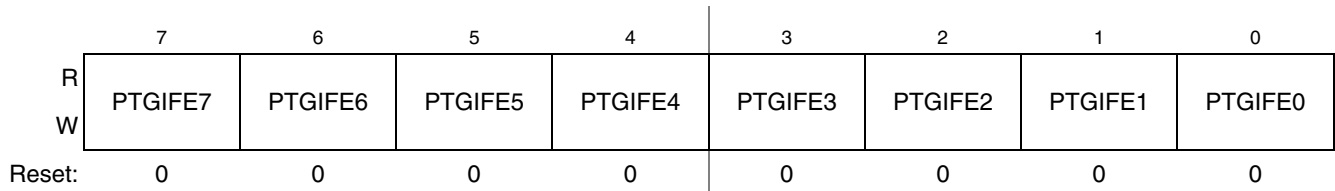


Figure 6-40. Port G Input Filter Enable Register (PTGIFE)

Table 6-40. PTGIFE Field Descriptions

Field	Description
7:0 PTGIFE[7:0]	<b>Port G Input Filter Enable</b> — Input low-pass filter enable control bits for PTG pins. 0 Input filter disabled 1 Input filter enabled

## 6.6.8 Port H Registers

Port H is controlled by the registers listed below.

### 6.6.8.1 Port H Data Register (PTHD)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTHD1	PTHD0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-41. Port H Data Register (PTHD)

Table 6-41. PTHD Register Field Descriptions

Field	Description
1:0 PTHD[1:0]	<b>Port H Data Register Bits</b> — For Port H pins that are inputs, reads return the logic level on the pin. For Port H 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 H pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTHD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups/pulldowns disabled.

### 6.6.8.2 Port H Data Direction Register (PTHDD)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTHDD1	PTHDD0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-42. Port H Data Direction Register (PTHDD)

Table 6-42. PTHDD Register Field Descriptions

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



### 6.6.8.3 Port H Pull Enable Register (PTHPE)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTHPE1	PTHPE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-43. Internal Pull Enable for Port H Register (PTHPE)

Table 6-43. PTHPE Register Field Descriptions

Field	Description
1:0 PTHPE[1:0]	<b>Internal Pull Enable for Port E Bits</b> — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTH pin. 0 Internal pullup/pulldown device disabled for port H bit n. 1 Internal pullup/pulldown device enabled for port H bit n.

### 6.6.8.4 Port H Slew Rate Enable Register (PTHSE)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTHSE1	PTHSE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-44. Slew Rate Enable for Port H Register (PTHSE)

Table 6-44. PTHSE Register Field Descriptions

Field	Description
1:0 PTHSE[1:0]	<b>Output Slew Rate Enable for Port H Bits</b> — Each of these control bits determines if the output slew rate control is enabled for the associated PTH pin. 0 Output slew rate control disabled for port H bit n. 1 Output slew rate control enabled for port H bit n.

### 6.6.8.5 Port H Input Filter Enable Register (PTHIFE)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTHIFE1	PTHIFE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-45. Port H Input Filter Enable Register (PTHIFE)

Table 6-45. PTHIFE Field Descriptions

Field	Description
1:0 PTHIFE[1:0]	<b>Port H Input Filter Enable</b> — Input low-pass filter enable control bits for PTH pins. 0 Input filter disabled 1 Input filter enabled

## 6.7 Pin Mux Controls

Package pins on the MC9S08GW64 series can be programmed for up to five different functions using the Pin Mux Control Registers. Controls are organized by GPIO Port. Each GPIO port has three mux control registers, comprised 4 bits per package pin. All pin mux registers default to value 0x00 at reset excluding PTA6. The default value of PTA6 is 0x11 at reset and its corresponding function is BKGD/MS. Alternate values are defined in the remainder of this section. Please note that the encoding for each function matches the column number in which that function occurs in [Table 2-1](#). That is, default functions are assigned value 0x00, ALT1 functions 0x01 and so forth.

Pin mux controls will leave GPIO input buffers enabled for all digital functions, regardless of mux control selection. This gives the user the ability to poll a port to decode a key pressed after a keyboard interrupt is asserted. The same function can be used for IRQ pin level detection and debounce software.

Table 6-46. Pin Mux Control Registers

Address	Register	Bit 7	6	5	4	3	2	1	Bit 0
0x1940	PTAPF1	0	A1			0	A0		
0x1941	PTAPF2	0	A3			0	A2		
0x1942	PTAPF3	0	A5			0	A4		
0x1943	PTAPF4	0	0	0	0	0	A6		
0x1944	PTBPF1	0	B1			0	B0		
0x1945	PTBPF2	0	B3			0	B2		
0x1946	PTBPF3	0	B5			0	B4		
0x1947	PTBPF4	0	B7			0	B6		
0x1948	PTCPF1	0	C1			0	C0		
0x1949	PTCPF2	0	C3			0	C2		
0x194A	PTCPF3	0	C5			0	C4		
0x194B	PTCPF4	0	C7			0	C6		
0x194C	PTDPF1	0	D1			0	D0		
0x194D	PTDPF2	0	D3			0	D2		
0x194E	PTDPF3	0	D5			0	D4		
0x194F	PTDPF4	0	D7			0	D6		
0x1950	PTEPF1	0	E1			0	E0		

Table 6-46. Pin Mux Control Registers

0x1951	PTEPF2	0	E3	0	E2
0x1952	PTEPF3	0	E5	0	E4
0x1953	PTEPF4	0	E7	0	E6
0x1954	PTFPF1	0	F1	0	F0
0x1955	PTFPF2	0	F3	0	F2
0x1956	PTFPF3	0	F5	0	F4
0x1957	PTFPF4	0	F7	0	F6
0x1958	PTGPF1	0	G1	0	G0
0x1959	PTGPF2	0	G3	0	G2
0x195A	PTGPF3	0	G5	0	G4
0x195B	PTGPF4	0	G7	0	G6
0x195C	PTHPF1	0	H1	0	H0

**NOTE**

Configuring the mux control registers to the ADC pins disables the digital function of the pin to ensure proper ADC conversion.

The port must not be configured as reserved functions, it may cause pin damage.

6.7.1 Port A Pin Function Register 1 (PTAPF1)

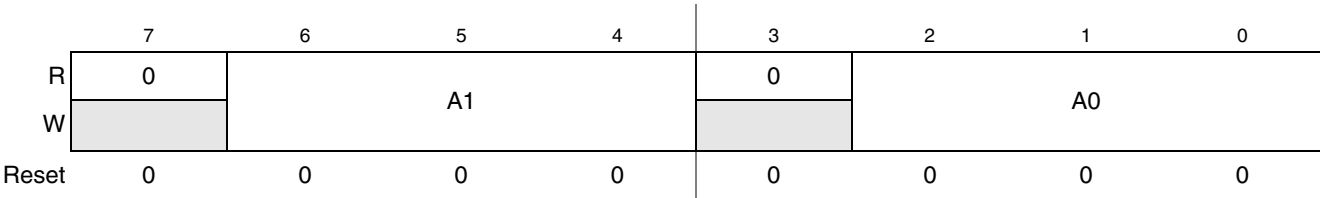


Figure 6-46. Port A Pin Function Register 1 (PTAPF1)

Table 6-47. PTAPF1 Field Descriptions

Field	Description
6–4 A1	<b>Port A1 Pin Mux Controls.</b> 000 PTA1 001 MISO2 010 PCNT1 011 SDA 100 AD3 101–111Reserved
2–0 A0	<b>Port A0 Pin Mux Controls.</b> 000 PTA0 001 MOSI2 010 PCNTCH0 011 SCL 100 AD2 101–111Reserved

## 6.7.2 Port A Pin Function Register 2 (PTAPF2)

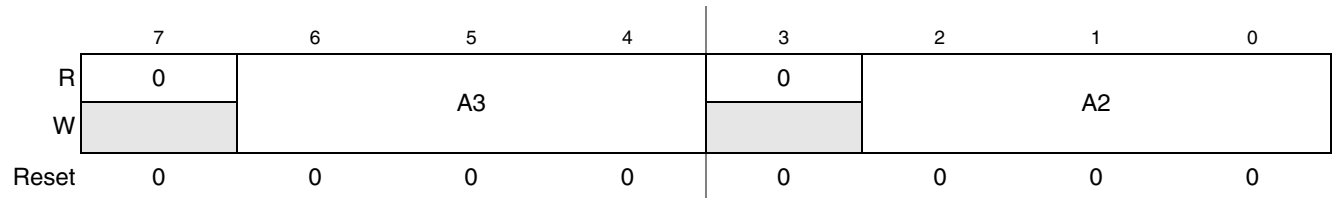


Figure 6-47. Port A Pin Function Register 2 (PTAPF2)

Table 6-48. PTAPF2 Field Descriptions

Field	Description
6–4 A3	<b>Port A3 Pin Mux Controls.</b> 000 PTA3 001 $\overline{SS2}$ 010 FTMCH1 011 PCNT1 100 CMMP1 101 ~ 111 Reserved
2–0 A2	<b>Port A2 Pin Mux Controls.</b> 000 PTA2 001 SCLK2 010 FTMCH0 011 PCNT0 100 CMPP0 101–111 Reserved

### 6.7.3 Port A Pin Function Register 3 (PTAPF3)



Figure 6-48. Port A Pin Function Register 3 (PTAPF3)

Table 6-49. PTAPF3 Field Descriptions

Field	Description
6–4 A5	<b>Port A5 Pin Mux Controls.</b> 000 PTA5 001 FTMCLK 010 TxD2 011 EXTRIG 100 IRQ 101–111 Reserved
2–0 A4	<b>Port A4 Pin Mux Controls.</b> 000 PTA4 001 MTIMCLK 010 RxD2 011 PCNT2 100 CMPP2 101–111 Reserved

### 6.7.4 Port A Pin Function Register 4 (PTAPF4)

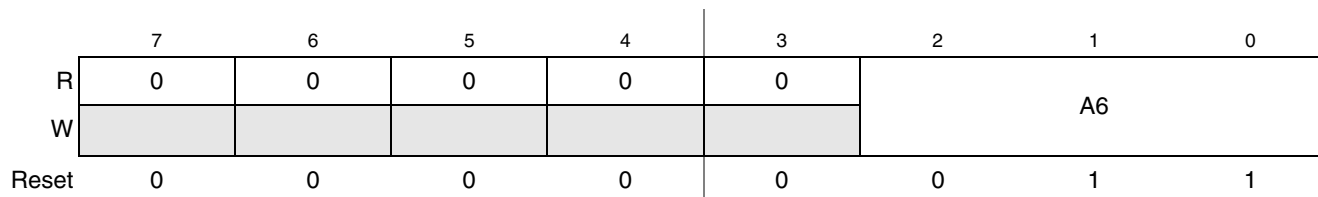


Figure 6-49. Port A Pin Function Register 4 (PTAPF4)

Table 6-50. PTAPF4 Field Descriptions

Field	Description
2–0 A6	<b>Port A6 Pin Mux Controls.</b> 000 PTA6 001 CMPOUT0 010 CLKOUT 011 BKGD/MS 100–111 Reserved

## 6.7.5 Port B Pin Function Register 1 (PTBPF1)

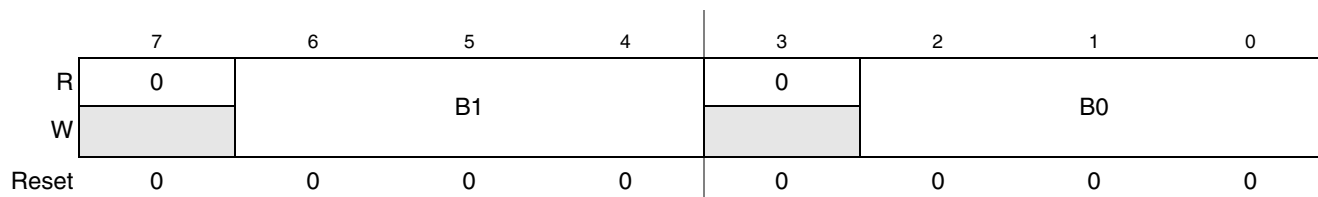


Figure 6-50. Port B Pin Function Register 1 (PTBPF1)

Table 6-51. PTBPF1 Field Descriptions

Field	Description
6–4 B1	<b>Port B1 Pin Mux Controls.</b> 000 PTB1 001 KBIP1 010 RxD1 011 CMPP6 100 XTAL2 101 –111Reserved
2–0 B0	<b>Port B0 Pin Mux Controls.</b> 000 PTB0 001 KBIP0 010 TxD1 011 EXTAL2 100–111Reserved

6.7.6 Port B Pin Function Register 2 (PTBPF2)

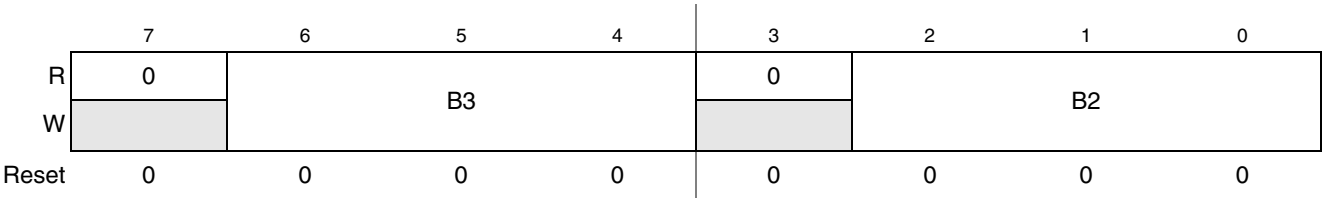


Figure 6-51. Port B Pin Function Register 2 (PTBPF2)

Table 6-52. PTBPF2 Field Descriptions

Field	Description
6–4 B3	<b>Port B3 Pin Mux Controls.</b> 000 PTB3 001 KBIP3 010 MISO0 011 MOSI0 100 TxD0 101–111Reserved
2–0 B2	<b>Port B2 Pin Mux Controls.</b> 000 PTB2 001 KBIP2 010 MOSI0 011 MISO0 100 RxD0 101–111Reserved



### 6.7.7 Port B Pin Function Register 3 (PTBPF3)



Figure 6-52. Port B Pin Function Register 3 (PTBPF3)

Table 6-53. PTBPF3 Field Descriptions

Field	Description
6–4 B5	<b>Port B5 Pin Mux Controls.</b> 000 PTB5 001 KBIP5 010 $\overline{SS0}$ 011 SDA 100–111 Reserved
2–0 B4	<b>Port B4 Pin Mux Controls.</b> 000 PTB4 001 KBIP4 010 SCLK0 011 SCL 100–111 Reserved

### 6.7.8 Port B Pin Function Register 4(PTBPF4)

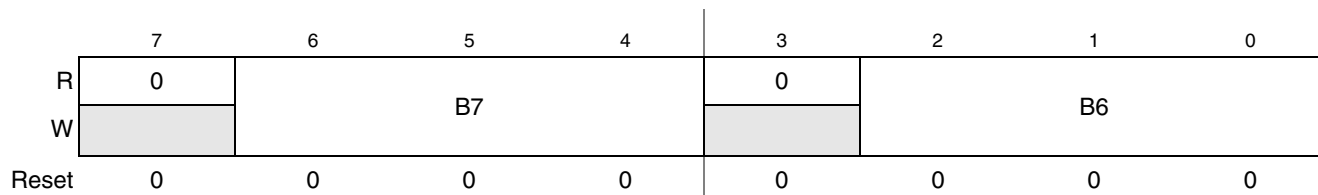


Figure 6-53. Port B Pin Function Register 4 (PTBPF4)

Table 6-54. PTBPF4 Field Descriptions

Field	Description
6–4 B7	<b>Port B7 Pin Mux Controls.</b> 000 PTB7 001 KBIP7 010 TxD2 011 LCD1 100–111 Reserved
2–0 B6	<b>Port B6 Pin Mux Controls.</b> 000 PTB6 001 KBIP6 010 RxD2 011 LCD0 100–111 Reserved

### 6.7.9 Port C Pin Function Register 1 (PTCPF1)



Figure 6-54. Port C Pin Function Register 1 (PTCPF1)

Table 6-55. PTCPF1 Field Descriptions

Field	Description
6–4 C1	<b>Port C1 Pin Mux Controls.</b> 000 PTC1 001 MISO1 010 LCD3 011–111Reserved
2–0 C0	<b>Port C0 Pin Mux Controls.</b> 000 PTC0 001 MOSI1 010 LCD2 011–111Reserved

### 6.7.10 Port C Pin Function Register 2 (PTCPF2)

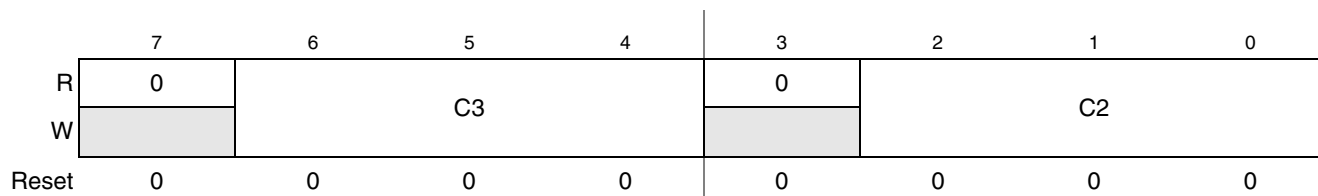


Figure 6-55. Port C Pin Function Register 2 (PTCPF2)

Table 6-56. PTCPF2 Field Descriptions

Field	Description
6–4 C3	<b>Port C3 Pin Mux Controls.</b> 000 PTC3 001 SS1 010 LCD5 011–111Reserved
2–0 C2	<b>Port C2 Pin Mux Controls.</b> 000 PTC2 001 SCLK1 010 LCD4 011–111Reserved

### 6.7.11 Port C Pin Function Register 3 (PTCPF3)



Figure 6-56. Port C Pin Function Register 3 (PTCPF3)

Table 6-57. PTCPF3 Field Descriptions

Field	Description
6–4 C5	<b>Port C5 Pin Mux Controls.</b> 000 PTC5 001 FTMCH1 010 TxD1 011 LCD7 100–111 Reserved
2–0 C4	<b>Port C4 Pin Mux Controls.</b> 000 PTC4 001 FTMCH0 010 RxD1 011 LCD6 100–111 Reserved

### 6.7.12 Port C Pin Function Register 4 (PTCPF4)

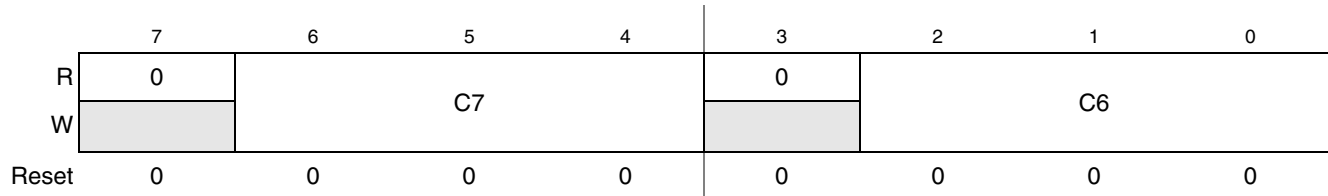


Figure 6-57. Port C Pin Function Register 4 (PTCPF4)

Table 6-58. PTCPF4 Field Descriptions

Field	Description
6–4 C7	<b>Port C7 Pin Mux Controls.</b> 000 PTC7 001 PCNTCH1 010 TxD3 011 LCD9 100–111 Reserved
2–0 C6	<b>Port C6 Pin Mux Controls.</b> 000 PTC6 001 PCNTCH0 010 RxD3 011 LCD8 100–111 Reserved

### 6.7.13 Port D Pin Function Register 1 (PTDPF1)

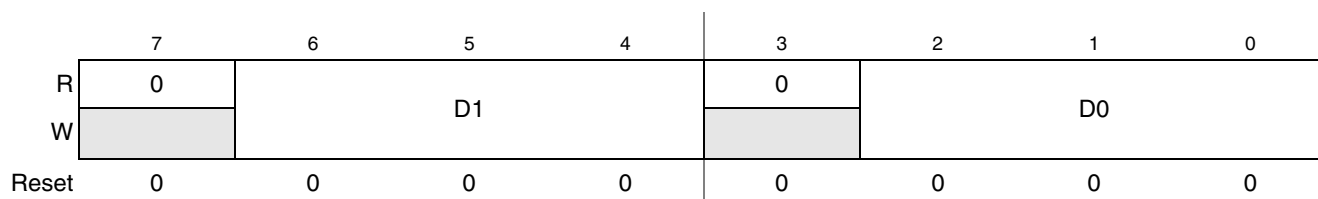


Figure 6-58. Port D Pin Function Register 1 (PTDPF1)

Table 6-59. PTDPF1 Field Descriptions

Field	Description
6–4 D1	<b>Port D1 Pin Mux Controls.</b> 000 PTD1 001 KBIP1 010 MISO2 011 LCD11 100–111Reserved
2–0 D0	<b>Port D0 Pin Mux Controls.</b> 000 PTD0 001 KBIP0 010 MOSI2 011 LCD10 100–111Reserved

### 6.7.14 Port D Pin Function Register 2 (PTDPF2)

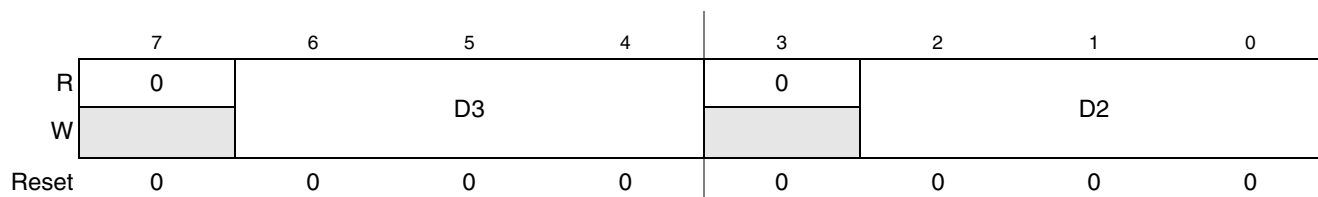


Figure 6-59. Port D Pin Function Register 2 (PTDPF2)

Table 6-60. PTDPF2 Field Descriptions

Field	Description
6–4 D3	<b>Port D3 Pin Mux Controls.</b> 000 PTD3 001 KBIP3 010 $\overline{SS2}$ 011 LCD13 100 –111Reserved
2–0 D2	<b>Port D2 Pin Mux Controls.</b> 000 PTD2 001 KBIP2 010 SCLK2 011 LCD12 100–111Reserved

### 6.7.15 Port D Pin Function Register 3 (PTDPF3)



Figure 6-60. Port D Pin Function Register 3 (PTDPF3)

Table 6-61. PTDPF3 Field Descriptions

Field	Description
6–4 D5	<b>Port D5 Pin Mux Controls.</b> 000 PTD5 001 KBIP5 010 CLKOUT 011 LCD15 100–111Reserved
2–0 D4	<b>Port D4 Pin Mux Controls.</b> 000 PTD4 001 KBIP4 010 LCD14 011–111Reserved

### 6.7.16 Port D Pin Function Register 4 (PTDPF4)



Figure 6-61. Port D Pin Function Register 4 (PTDPF4)

Table 6-62. PTDPF4 Field Descriptions

Field	Description
6–4 D7	<b>Port D7 Pin Mux Controls.</b> 000 PTD7 001 KBIP7 010 LCD17 011–111Reserved
2–0 D6	<b>Port D6 Pin Mux Controls.</b> 000 PTD6 001 KBIP6 010 LCD16 011–111Reserved

6.7.17 Port E Pin Function Register 1 (PTEPF1)

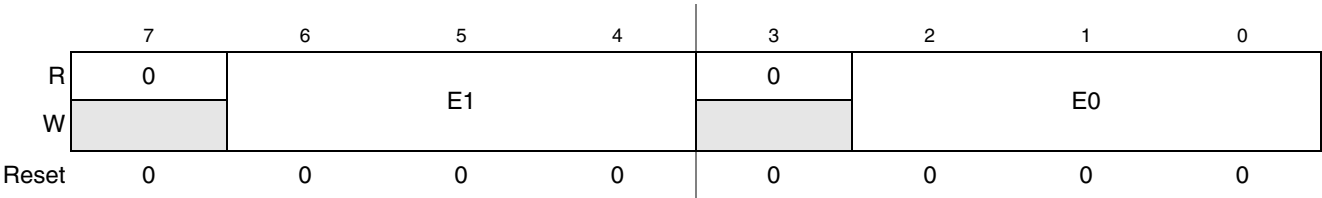


Figure 6-62. Port E Pin Function Register 1 (PTEPF1)

Table 6-63. PTEPF1 Field Descriptions

Field	Description
6–4 E1	<b>Port E1 Pin Mux Controls.</b> 000 PTE1 001 LCD19 010–111Reserved
2–0 E0	<b>Port E0 Pin Mux Controls.</b> 000 PTE0 001 LCD18 010–111Reserved

6.7.18 Port E Pin Function Register 2 (PTEPF2)

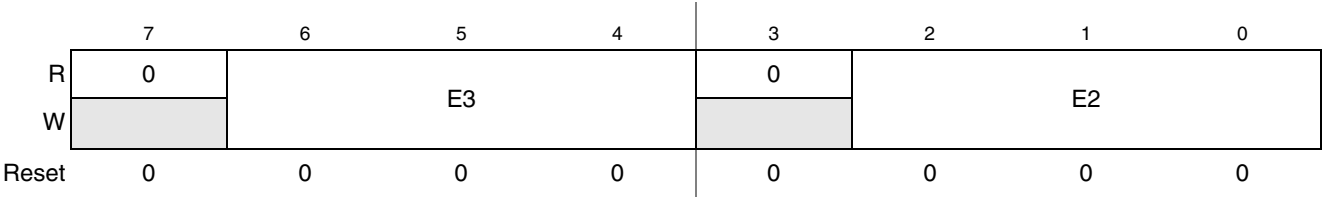


Figure 6-63. Port E Pin Function Register 2 (PTEPF2)

Table 6-64. PTEPF2 Field Descriptions

Field	Description
6–4 E3	<b>Port E3 Pin Mux Controls.</b> 000 PTE3 001 LCD21 010–111 Reserved
2–0 E2	<b>Port E2 Pin Mux Controls.</b> 000 PTE2 001 LCD20 010–111Reserved

### 6.7.19 Port E Pin Function Register 3 (PTEPF3)

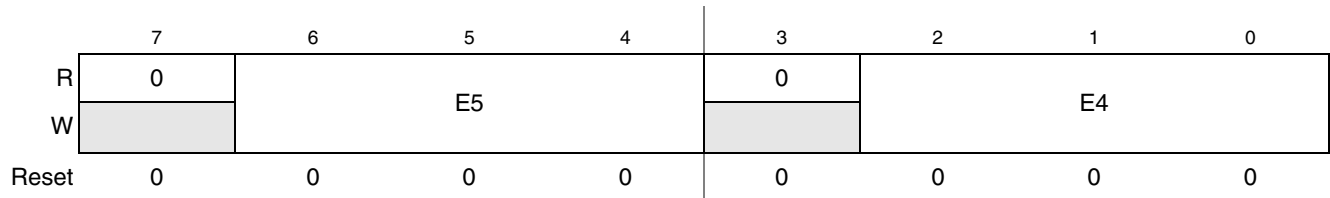


Figure 6-64. Port E Pin Function Register 3 (PTEPF3)

Table 6-65. PTEPF3 Field Descriptions

Field	Description
6–4 E5	<b>Port E5 Pin Mux Controls.</b> 000 PTE5 001 Reserved 010 LCD23 011–1111Reserved
2–0 E4	<b>Port E4 Pin Mux Controls.</b> 000 PTE4 001 Reserved 010 LCD22 011–111Reserved

### 6.7.20 Port E Pin Function Register 4 (PTEPF4)

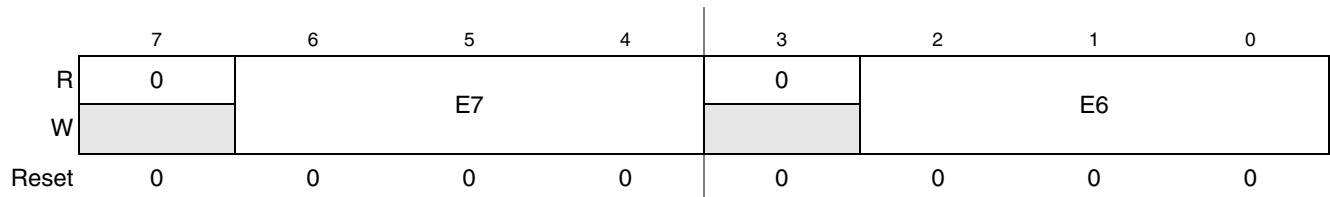


Figure 6-65. Port E Pin Function Register 4 (PTEPF4)

Table 6-66. PTEPF4 Field Descriptions

Field	Description
6–4 E7	<b>Port E7 Pin Mux Controls.</b> 000 PTE7 001 Reserved 010 LCD25 011–111Reserved
2–0 E6	<b>Port E6 Pin Mux Controls.</b> 000 PTE6 001 Reserved 010 LCD24 011–111Reserved

6.7.21 Port F Pin Function Register 1 (PTFPF1)

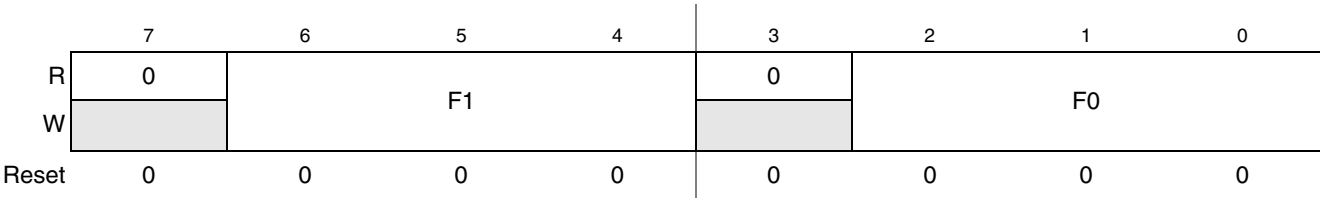


Figure 6-66. Port F Pin Function Register 1 (PTFPF1)

Table 6-67. PTFPF1 Field Descriptions

Field	Description
6–4 F1	<b>Port F1 Pin Mux Controls.</b> 000 PTF1 001 LCD27 010–111Reserved
2–0 F0	<b>Port F0 Pin Mux Controls.</b> 000 PTF0 001 LCD26 010–111Reserved

6.7.22 Port F Pin Function Register 2 (PTFPF2)

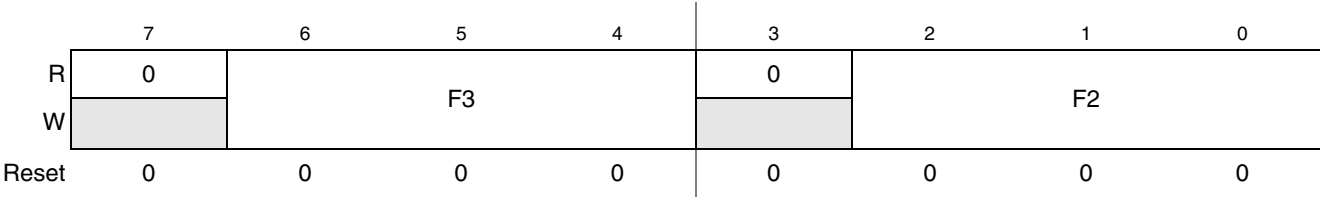


Figure 6-67. Port F Pin Function Register 2 (PTFPF2)

Table 6-68. PTFPF2 Field Descriptions

Field	Description
6–4 F3	<b>Port F3 Pin Mux Controls.</b> 000 PTF3 001 LCD29 010–111Reserved
2–0 F2	<b>Port F2 Pin Mux Controls.</b> 000 PTF2 001 LCD28 010–111Reserved



### 6.7.23 Port F Pin Function Register 3 (PTFPF3)



Figure 6-68. Port F Pin Function Register 3 (PTFPF3)

Table 6-69. PTFPF3 Field Descriptions

Field	Description
6–4 F5	<b>Port F5 Pin Mux Controls.</b> 000 PTF5 001 LCD31 010–111Reserved
2–0 F4	<b>Port F4 Pin Mux Controls.</b> 000 PTF4 001 LCD30 010 –111Reserved

### 6.7.24 Port F Pin Function Register 4 (PTFPF4)



Figure 6-69. Port F Pin Function Register 4 (PTFPF4)

Table 6-70. PTFPF4 Field Descriptions

Field	Description
6–4 F7	<b>Port F7 Pin Mux Controls.</b> 000 PTF7 001 FTMCLK 010 AD5 011 LCD33 100–111Reserved
2–0 F6	<b>Port F6 Pin Mux Controls.</b> 000 PTF6 001 MTIMCLK 010 AD4 011 LCD32 100–111Reserved

## 6.7.25 Port G Pin Function Register 1 (PTGPF1)



Figure 6-70. Port G Pin Function Register 1 (PTGPF1)

Table 6-71. PTGPF1 Field Descriptions

Field	Description
6–4 G1	<b>Port G1 Pin Mux Controls.</b> 000 PTG1 001 MISO1 010 AD7 011 LCD35 100–1111Reserved
2–0 G0	<b>Port G0 Pin Mux Controls.</b> 000 PTG0 001 MOSI1 010 AD6 011 LCD34 100–111Reserved

## 6.7.26 Port G Pin Function Register 2 (PTGPF2)

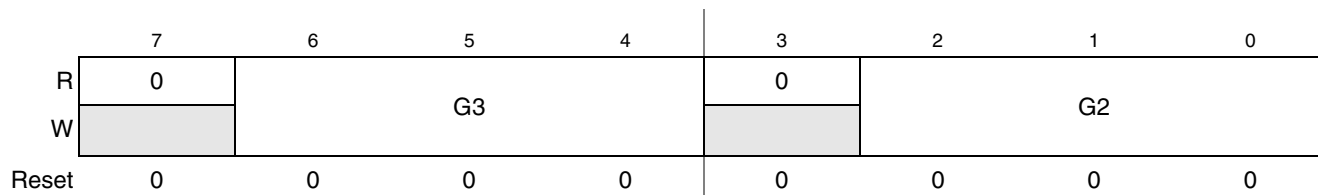


Figure 6-71. Port G Pin Function Register 2 (PTGPF2)

Table 6-72. PTGPF2 Field Descriptions

Field	Description
6–4 G3	<b>Port G3 Pin Mux Controls.</b> 000 PTG3 001 SS1 010 AD9 011 LCD37 100–1111Reserved
2–0 G2	<b>Port G2 Pin Mux Controls.</b> 000 PTG2 001 SCLK1 010 AD8 011 LCD36 100–111Reserved

### 6.7.27 Port G Pin Function Register 3 (PTGPF3)

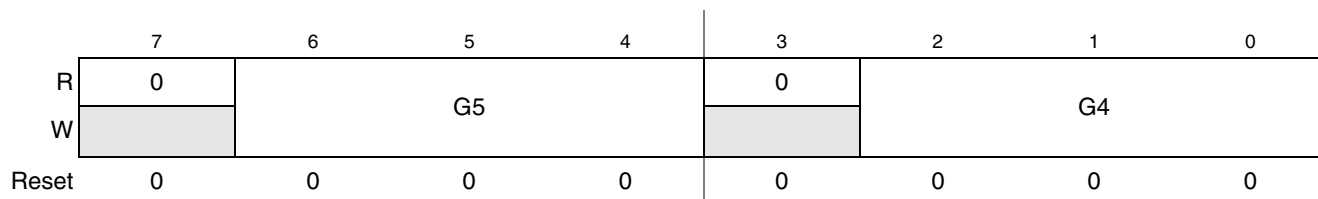


Figure 6-72. Port G Pin Function Register 3 (PTGPF3)

Table 6-73. PTGPF3 Field Descriptions

Field	Description
6–4 G5	<b>Port G5 Pin Mux Controls.</b> 000 PTG5 001 CMPOUT2 010 TxD3 011 AD11 100 LCD39 101–111 Reserved
2–0 G4	<b>Port G4 Pin Mux Controls.</b> 000 PTG4 001 CMPOUT1 010 RxD3 011 AD10 100 LCD38 101–111 Reserved

6.7.28 Port G Pin Function Register 4 (PTGPF4)

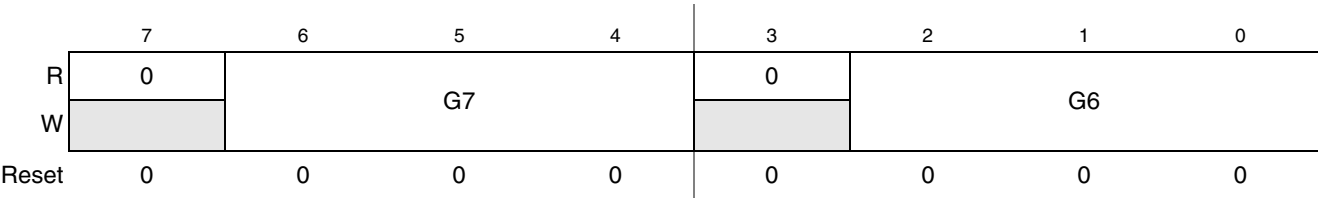


Figure 6-73. Port G Pin Function Register 4 (PTGPF4)

Table 6-74. PTGPF4 Field Descriptions

Field	Description
6–4 G7	<b>Port G7 Pin Mux Controls.</b> 000 PTG7 001 CMPP4 010 AD13 011 PCNT1 100 LCD41 101–111Reserved
2–0 G6	<b>Port G6 Pin Mux Controls.</b> 000 PTG6 001 CMPP3 010 AD12 011 PCNT0 100 LCD40 101–111Reserved

## 6.7.29 Port H Pin Function Register 1 (PTHPF1)



Figure 6-74. Port H Pin Function Register 1 (PTHPF1)

Table 6-75. PTHPF1 Field Descriptions

Field	Description
6–4 H1	<b>Port H1 Pin Mux Controls.</b> 000 PTH1 001 RTCCLKOUT 010 AD15 011 LCD43 100–111 Reserved
2–0 H0	<b>Port H0 Pin Mux Controls.</b> 000 PTH0 001 CMPP5 010 AD14 011 PCNT2 100 LCD42 101–111 Reserved



## Chapter 7

# Central Processor Unit (S08CPUV6)

### 7.1 Introduction

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

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

#### 7.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- 64-KB CPU address space with banked memory management unit for greater than 64 KB
- 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

## 7.2 Programmer's Model and CPU Registers

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

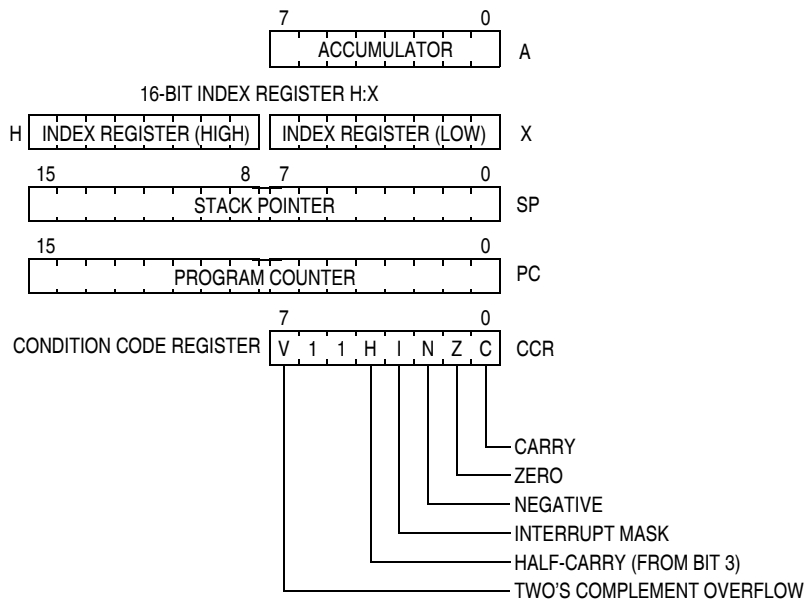


Figure 7-1. CPU Registers

### 7.2.1 Accumulator (A)

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

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

### 7.2.2 Index Register (H:X)

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

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

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



### 7.2.3 Stack Pointer (SP)

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

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

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

### 7.2.4 Program Counter (PC)

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

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

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

### 7.2.5 Condition Code Register (CCR)

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

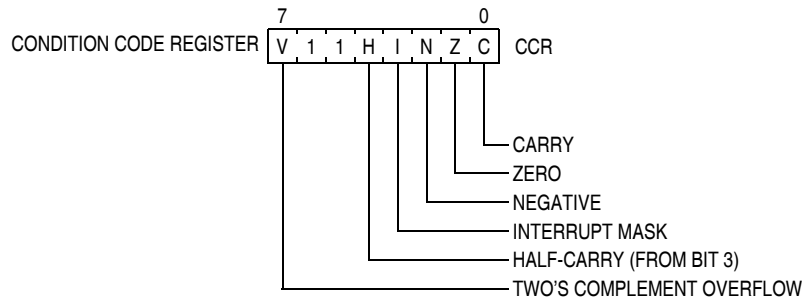


Figure 7-2. Condition Code Register

Table 7-1. CCR Register Field Descriptions

Field	Description
7 V	<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

## 7.3 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.

### NOTE

For more information about extended addressing modes, see the Memory Management Unit section in the Memory chapter.

MCU derivatives with more than 64-Kbytes of memory also include a memory management unit (MMU) to support extended memory space. A PPAGE register is used to manage 16-Kbyte pages of memory which can be accessed by the CPU through a 16-Kbyte window from 0x8000 through 0xBFFF. The CPU includes two special instructions (CALL and RTC). CALL operates like the JSR instruction except that CALL saves the current PPAGE value on the stack and provides a new PPAGE value for the destination. RTC works like the RTS instruction except RTC restores the old PPAGE value in addition to the PC during the return from the called routine. The MMU also includes a linear address pointer register and data access registers so that the extended memory space operates as if it was a single linear block of memory. For additional information about the MMU, refer to the Memory chapter of this data sheet.

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

### 7.3.1 Inherent Addressing Mode (INH)

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

### 7.3.2 Relative Addressing Mode (REL)

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

### 7.3.3 Immediate Addressing Mode (IMM)

In immediate addressing mode, the operand needed to complete the instruction is included in the object code immediately following the instruction opcode in memory. In the case of a 16-bit immediate operand,

the high-order byte is located in the next memory location after the opcode, and the low-order byte is located in the next memory location after that.

### 7.3.4 Direct Addressing Mode (DIR)

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

### 7.3.5 Extended Addressing Mode (EXT)

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

### 7.3.6 Indexed Addressing Mode

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 7.4 Special Operations

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

### 7.4.1 Reset Sequence

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

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

### 7.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.

3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.
6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

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

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

### 7.4.3 Wait Mode Operation

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

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

### 7.4.4 Stop Mode Operation

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

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface

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

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

### 7.4.5 BGND Instruction

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

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

The CALL is similar to a jump-to-subroutine (JSR) instruction, but the subroutine that is called can be located anywhere in the normal 64-Kbyte address space or on any page of program expansion memory. When CALL is executed, a return address is calculated, then it and the current program page register value are stacked, and a new instruction-supplied value is written to PPAGE. The PPAGE value controls which of the possible 16-Kbyte pages is visible through the window in the 64-Kbyte memory map. Execution continues at the address of the called subroutine.

The actual sequence of operations that occur during execution of CALL is:

1. CPU calculates the address of the next instruction after the CALL instruction (the return address) and pushes this 16-bit value onto the stack, low byte first.
2. CPU reads the old PPAGE value and pushes it onto the stack.
3. CPU writes the new instruction-supplied page select value to PPAGE. This switches the destination page into the program overlay window in the CPU address range 0x8000 0xBFFF.
4. Instruction queue is refilled starting from the destination address, and execution begins at the new address.

This sequence of operations is an uninterruptable CPU instruction. There is no need to inhibit interrupts during CALL execution. In addition, a CALL can be performed from any address in memory to any other address. This is a big improvement over other bank-switching schemes, where the page switch operation can be performed only by a program outside the overlay window.

For all practical purposes, the PPAGE value supplied by the instruction can be considered to be part of the effective address. The new page value is provided by an immediate operand in the instruction.

The RTC instruction is used to terminate subroutines invoked by a CALL instruction. RTC unstacks the PPAGE value and the return address, the queue is refilled, and execution resumes with the next instruction after the corresponding CALL.

The actual sequence of operations that occur during execution of RTC is:

1. The return value of the 8-bit PPAGE register is pulled from the stack.
2. The 16-bit return address is pulled from the stack and loaded into the PC.
3. The return PPAGE value is written to the PPAGE register.
4. The queue is refilled and execution begins at the new address.

Since the return operation is implemented as a single uninterruptable CPU instruction, the RTC can be executed from anywhere in memory, including from a different page of extended memory in the overlay window.

The CALL and RTC instructions behave like JSR and RTS, except they have slightly longer execution times. Since extra execution cycles are required, routinely substituting CALL/RTC for JSR/RTS is not recommended. JSR and RTS can be used to access subroutines that are located outside the program overlay window or on the same memory page. However, if a subroutine can be called from other pages, it must be terminated with an RTC. In this case, since RTC unstacks the PPAGE value as well as the return address, all accesses to the subroutine, even those made from the same page, must use CALL instructions.



## 7.5 HCS08 Instruction Set Summary

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

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

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

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
BCC <i>rel</i>	Branch if Carry Bit Clear (if C = 0)	REL	24 rr	3	ppp	- 1 1 -	- - - -
BCLR <i>n,opr8a</i>	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	r fwpp r fwpp r fwpp r fwpp r fwpp r fwpp r fwpp r fwpp	- 1 1 -	- - - -
BCS <i>rel</i>	Branch if Carry Bit Set (if C = 1) (Same as BLO)	REL	25 rr	3	ppp	- 1 1 -	- - - -
BEQ <i>rel</i>	Branch if Equal (if Z = 1)	REL	27 rr	3	ppp	- 1 1 -	- - - -
BGE <i>rel</i>	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+	fp...ppp	- 1 1 -	- - - -
BGT <i>rel</i>	Branch if Greater Than (if Z   (N ⊕ V) = 0) (Signed)	REL	92 rr	3	ppp	- 1 1 -	- - - -
BHCC <i>rel</i>	Branch if Half Carry Bit Clear (if H = 0)	REL	28 rr	3	ppp	- 1 1 -	- - - -
BHCS <i>rel</i>	Branch if Half Carry Bit Set (if H = 1)	REL	29 rr	3	ppp	- 1 1 -	- - - -
BHI <i>rel</i>	Branch if Higher (if C   Z = 0)	REL	22 rr	3	ppp	- 1 1 -	- - - -
BHS <i>rel</i>	Branch if Higher or Same (if C = 0) (Same as BCC)	REL	24 rr	3	ppp	- 1 1 -	- - - -
BIH <i>rel</i>	Branch if IRQ Pin High (if IRQ pin = 1)	REL	2F rr	3	ppp	- 1 1 -	- - - -
BIL <i>rel</i>	Branch if IRQ Pin Low (if IRQ pin = 0)	REL	2E rr	3	ppp	- 1 1 -	- - - -
BIT # <i>opr8i</i> BIT <i>opr8a</i> BIT <i>opr16a</i> BIT <i>opr16,X</i> BIT <i>opr8,X</i> BIT <i>,X</i> BIT <i>opr16,SP</i> BIT <i>opr8,SP</i>	Bit Test (A) & (M) (CCR Updated but Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 ii B5 dd C5 hh ll D5 ee ff E5 ff F5 9E D5 ee ff 9E E5 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 -	- ↑ ↓ -
BLE <i>rel</i>	Branch if Less Than or Equal To (if Z   (N ⊕ V) = 1) (Signed)	REL	93 rr	3	ppp	- 1 1 -	- - - -
BLO <i>rel</i>	Branch if Lower (if C = 1) (Same as BCS)	REL	25 rr	3	ppp	- 1 1 -	- - - -
BLS <i>rel</i>	Branch if Lower or Same (if C   Z = 1)	REL	23 rr	3	ppp	- 1 1 -	- - - -
BLT <i>rel</i>	Branch if Less Than (if N ⊕ V = 1) (Signed)	REL	91 rr	3	ppp	- 1 1 -	- - - -
BMC <i>rel</i>	Branch if Interrupt Mask Clear (if I = 0)	REL	2C rr	3	ppp	- 1 1 -	- - - -
BMI <i>rel</i>	Branch if Minus (if N = 1)	REL	2B rr	3	ppp	- 1 1 -	- - - -
BMS <i>rel</i>	Branch if Interrupt Mask Set (if I = 1)	REL	2D rr	3	ppp	- 1 1 -	- - - -
BNE <i>rel</i>	Branch if Not Equal (if Z = 0)	REL	26 rr	3	ppp	- 1 1 -	- - - -

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
BPL <i>rel</i>	Branch if Plus (if N = 0)	REL	2A rr	3	ppp	- 1 1 -	- - - -
BRA <i>rel</i>	Branch Always (if I = 1)	REL	20 rr	3	ppp	- 1 1 -	- - - -
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> 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 <i>rel</i>	Branch Never (if I = 0)	REL	21 rr	3	ppp	- 1 1 -	- - - -
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> 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 <i>n,opr8a</i>	Set Bit <i>n</i> 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	rffwpp rffwpp rffwpp rffwpp rffwpp rffwpp rffwpp rffwpp	- 1 1 -	- - - -
BSR <i>rel</i>	Branch to Subroutine PC ← (PC) + \$0002 push (PCL); SP ← (SP) - \$0001 push (PCH); SP ← (SP) - \$0001 PC ← (PC) + <i>rel</i>	REL	AD rr	5	ssppp	- 1 1 -	- - - -
CALL <i>page, opr16a</i>	Call Subroutine	EXT	AC pg hhll	8	ppsspppp	- 1 1 -	- - - -
CBEQ <i>opr8a,rel</i> CBEQA # <i>opr8i,rel</i> CBEQX # <i>opr8i,rel</i> CBEQ <i>opr8,X+,rel</i> CBEQ <i>,X+,rel</i> CBEQ <i>opr8,SP,rel</i>	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 rffppp prpppp	- 1 1 -	- - - -
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	- 1 1 -	- - - 0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	- 1 1 -	0 - - -
CLR <i>opr8a</i> CLRA CLR X CLR H CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	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	rffwpp p p p rffwpp rffw prffwpp	0 1 1 -	- 0 1 -

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
CMP #opr8i CMP opr8a CMP opr16a CMP oprx16,X CMP oprx8,X CMP ,X CMP oprx16,SP CMP oprx8,SP	Compare Accumulator with Memory $A - M$ (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	$\uparrow 1 1 -$	$- \uparrow \uparrow \uparrow$
COM opr8a COMA COMX COM oprx8,X COM ,X COM oprx8,SP	Complement $M \leftarrow (\overline{M}) = \$FF - (M)$ (One's Complement) $A \leftarrow (\overline{A}) = \$FF - (A)$ $X \leftarrow (\overline{X}) = \$FF - (X)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$	DIR INH INH IX1 IX SP1	33 dd 43 53 63 ff 73 9E 63 ff	5 1 1 5 4 6	rffwpp p p rffwpp rffw prffwpp	$0 1 1 -$	$- \uparrow \uparrow 1$
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	prrrfpp ppp rrfpp prrrfpp	$\uparrow 1 1 -$	$- \uparrow \uparrow \uparrow$
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	$\uparrow 1 1 -$	$- \uparrow \uparrow \uparrow$
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p	$U 1 1 -$	$- \uparrow \uparrow \uparrow$
DBNZ opr8a,rel DBNZA rel DBNZX rel DBNZ oprx8,X,rel DBNZ ,X,rel DBNZ oprx8,SP,rel	Decrement A, X, or M and Branch if Not Zero (if (result) $\neq 0$ ) DBNZX Affects X Not H	DIR INH INH IX1 IX SP1	3B dd rr 4B rr 5B rr 6B ff rr 7B rr 9E 6B ff rr	7 4 4 7 6 8	rffwpppp fppp fppp rffwpppp rffwppp prffwpppp	$- 1 1 -$	$- - - -$
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	rffwpp p p rffwpp rffw prffwpp	$\uparrow 1 1 -$	$- \uparrow \uparrow -$
DIV	Divide $A \leftarrow (H:A) \div (X)$ ; $H \leftarrow$ Remainder	INH	52	6	fffffp	$- 1 1 -$	$- - \uparrow \uparrow$
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	$0 1 1 -$	$- \uparrow \uparrow -$

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

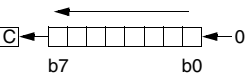
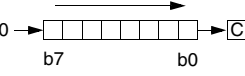
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V I 1 H	I N Z C
INC <i>opr8a</i> INCA INCX INC <i>opr8,X</i> INC ,X INC <i>opr8,SP</i>	Increment $M \leftarrow (M) + \$01$ $A \leftarrow (A) + \$01$ $X \leftarrow (X) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$	DIR INH INH IX1 IX SP1	3C dd 4C 5C 6C ff 7C 9E 6C ff	5 1 1 5 4 6	r fwpp p p r fwpp r fwp p rfwpp	$\uparrow 1 1 -$	$- \uparrow \uparrow -$
JMP <i>opr8a</i> JMP <i>opr16a</i> JMP <i>opr16,X</i> JMP <i>opr8,X</i> JMP ,X	Jump $PC \leftarrow \text{Jump Address}$	DIR EXT IX2 IX1 IX	BC dd CC hh ll DC ee ff EC ff FC	3 4 4 3 3	ppp pppp pppp ppp ppp	$- 1 1 -$	$- - - -$
JSR <i>opr8a</i> JSR <i>opr16a</i> JSR <i>opr16,X</i> JSR <i>opr8,X</i> JSR ,X	Jump to Subroutine $PC \leftarrow (PC) + n$ ( $n = 1, 2, \text{ or } 3$ ) Push (PCL); $SP \leftarrow (SP) - \$0001$ Push (PCH); $SP \leftarrow (SP) - \$0001$ $PC \leftarrow \text{Unconditional Address}$	DIR EXT IX2 IX1 IX	BD dd CD hh ll DD ee ff ED ff FD	5 6 6 5 5	ssppp pssppp pssppp ssppp ssppp	$- 1 1 -$	$- - - -$
LDA # <i>opr8i</i> LDA <i>opr8a</i> LDA <i>opr16a</i> LDA <i>opr16,X</i> LDA <i>opr8,X</i> LDA ,X LDA <i>opr16,SP</i> LDA <i>opr8,SP</i>	Load Accumulator from Memory $A \leftarrow (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 ii B6 dd C6 hh ll D6 ee ff E6 ff F6 9E D6 ee ff 9E E6 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	$0 1 1 -$	$- \uparrow \uparrow -$
LDHX # <i>opr16i</i> LDHX <i>opr8a</i> LDHX <i>opr16a</i> LDHX ,X LDHX <i>opr16,X</i> LDHX <i>opr8,X</i> LDHX <i>opr8,SP</i>	Load Index Register (H:X) $H:X \leftarrow (M:M + \$0001)$	IMM DIR EXT IX IX2 IX1 SP1	45 jj kk 55 dd 32 hh ll 9E AE 9E BE ee ff 9E CE ff 9E FE ff	3 4 5 5 6 5 5	ppp rrpp prrrpp prrrp pprrpp prrrpp prrrpp	$0 1 1 -$	$- \uparrow \uparrow -$
LDX # <i>opr8i</i> LDX <i>opr8a</i> LDX <i>opr16a</i> LDX <i>opr16,X</i> LDX <i>opr8,X</i> LDX ,X LDX <i>opr16,SP</i> LDX <i>opr8,SP</i>	Load X (Index Register Low) from Memory $X \leftarrow (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE ii BE dd CE hh ll DE ee ff EE ff FE 9E DE ee ff 9E EE ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	$0 1 1 -$	$- \uparrow \uparrow -$
LSL <i>opr8a</i> LSLA LSLX LSL <i>opr8,X</i> LSL ,X LSL <i>opr8,SP</i>	Logical Shift Left  (Same as ASL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	r fwpp p p r fwpp r fwp p rfwpp	$\uparrow 1 1 -$	$- \uparrow \uparrow \uparrow$
LSR <i>opr8a</i> LSRA LSRX LSR <i>opr8,X</i> LSR ,X LSR <i>opr8,SP</i>	Logical Shift Right 	DIR INH INH IX1 IX SP1	34 dd 44 54 64 ff 74 9E 64 ff	5 1 1 5 4 6	r fwpp p p r fwpp r fwp p rfwpp	$\uparrow 1 1 -$	$- 0 \uparrow \uparrow$

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

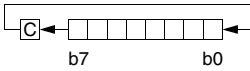
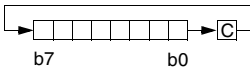
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
MOV <i>opr8a,opr8a</i> MOV <i>opr8a,X+</i> MOV <i>#opr8i,opr8a</i> MOV <i>,X+,opr8a</i>	Move $(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ In IX+/DIR and DIR/IX+ Modes, $H:X \leftarrow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

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V I 1 H	I N Z C
RSP	Reset Stack Pointer (Low Byte) SPL ← \$FF (High Byte Not Affected)	INH	9C	1	p	- 1 1 -	- - - -
RTC	Return from CALL	INH	8D	7	uuufppp	- 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	↑ 1 1 ↑	↑ ↓ ↑ ↓
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	↑ 1 1 -	- ↑ ↓ ↓
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 -	- ↑ ↓ ↓
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 -	- ↑ ↓ ↓
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 -	- ↑ ↓ ↓

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
SUB #opr8i SUB opr8a SUB opr16a SUB oprx16,X SUB oprx8,X SUB ,X SUB oprx16,SP SUB oprx8,SP	Subtract $A \leftarrow (A) - (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 ii B0 dd C0 hh ll D0 ee ff E0 ff F0 9E D0 ee ff 9E E0 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	$\uparrow 1 1 -$	$- \uparrow \uparrow \uparrow$
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	$\uparrow 1 1 \uparrow$	$\uparrow \uparrow \uparrow \uparrow$
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	$0 1 1 -$	$- \uparrow \uparrow -$
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 7-2. Instruction Set Summary (Sheet 9 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V I 1 H	I N Z C
TXS	Transfer Index Reg. to SP $SP \leftarrow (H:X) - \$0001$	INH	94	2	f <sub>p</sub>	- 1 1 -	- - - -
WAIT	Enable Interrupts; Wait for Interrupt I bit $\leftarrow$ 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.  
*opr8* Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.  
*opr16* Any label or expression that evaluates to a 16-bit value, used for indexed addressing.  
*rel* Any label or expression that refers to an address that is within -128 to +127 locations from the start of the next instruction.

**Operation Symbols:**

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

**CCR Bits:**

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

**Addressing Modes:**

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

**Cycle-by-Cycle Codes:**

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

**CCR Effects:**

- ↑ Set or cleared  
- Not affected  
U Undefined

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

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

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

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

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

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

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

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

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



## Chapter 8

# Keyboard Interrupt (S08KBIV2)

### 8.1 Introduction

The keyboard interrupt (KBI) module provides up to eight independently enabled external interrupt sources. All KBI pins share KBI functionality with GPIO pins. Each keyboard interrupt has independent pin enable bit and edge select bit for different usages.

#### 8.1.1 KBI Clock Gating

The bus clock to the KBI module can be gated on and off using the KBI bit in SCGC1. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the KBI bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

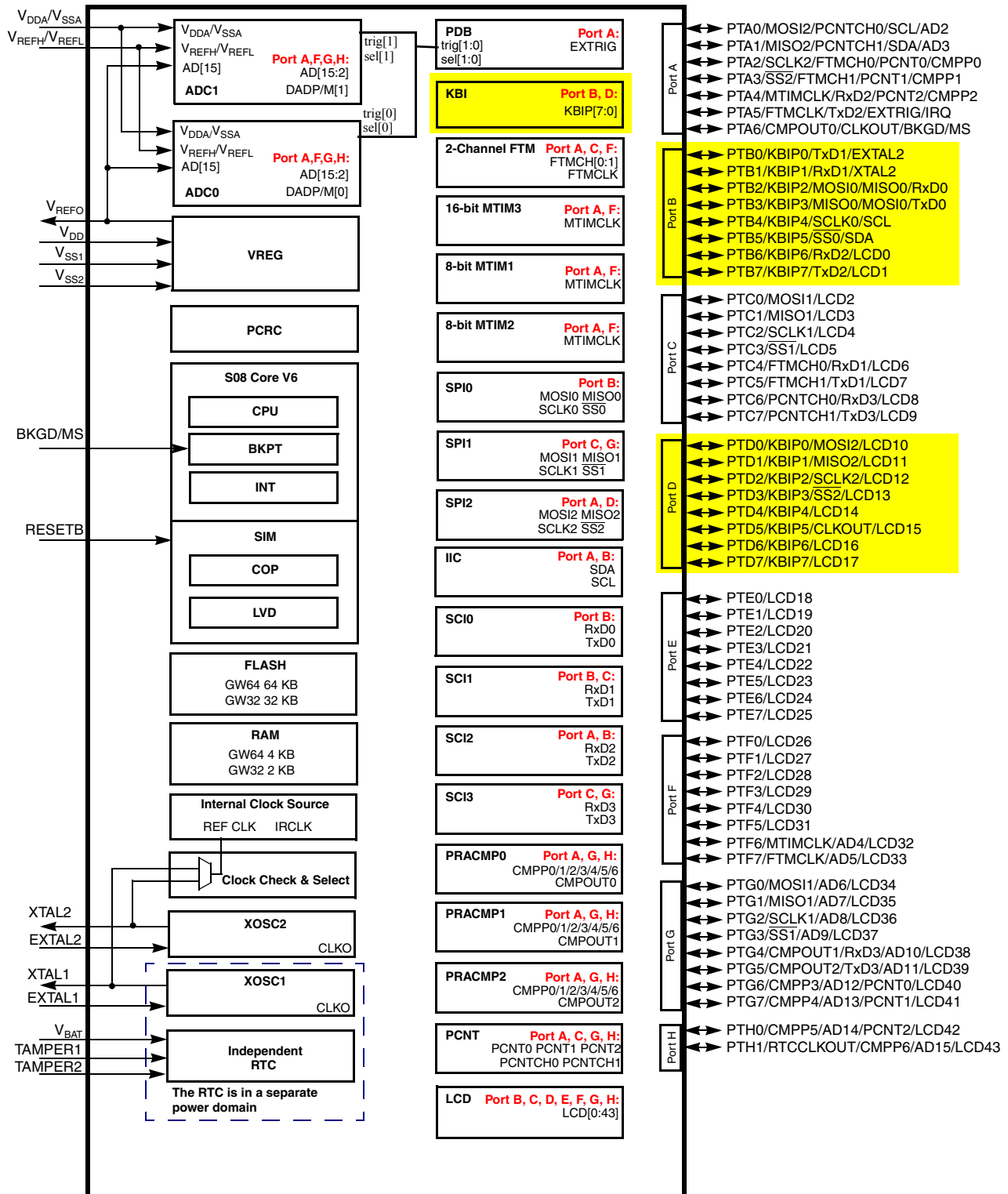


Figure 8-1. MC9S08GW64 Series Block Diagram Highlighting KBI Module and Pins

## 8.1.2 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.

## 8.1.3 Modes of Operation

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

### 8.1.3.1 KBI in Wait Mode

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

### 8.1.3.2 KBI in Stop Modes

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

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

### 8.1.3.3 KBI in Active Background Mode

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

## 8.1.4 Block Diagram

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

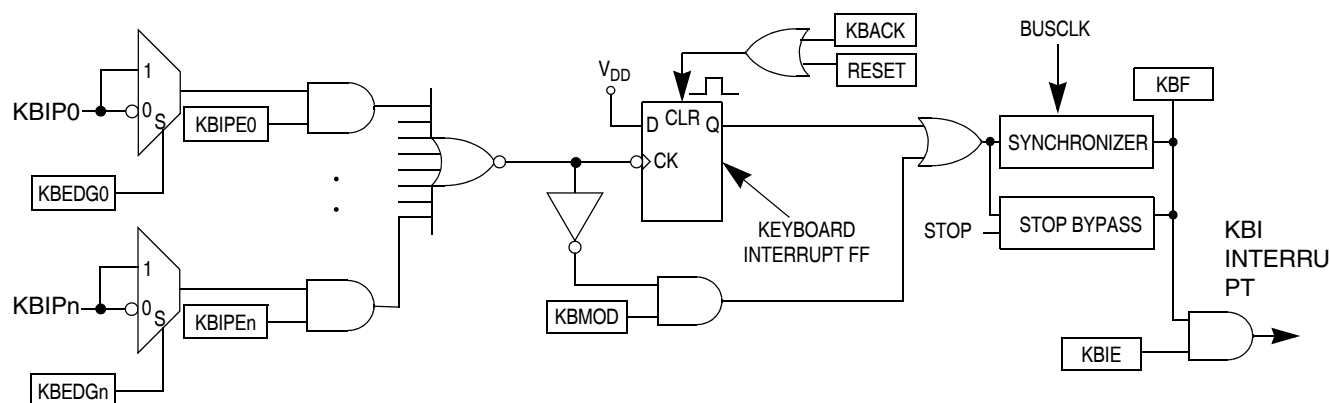


Figure 8-2. KBI Block Diagram

## 8.2 External Signal Description

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

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

Table 8-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

## 8.3 Register Definition

The KBI includes three registers:

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

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

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

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

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



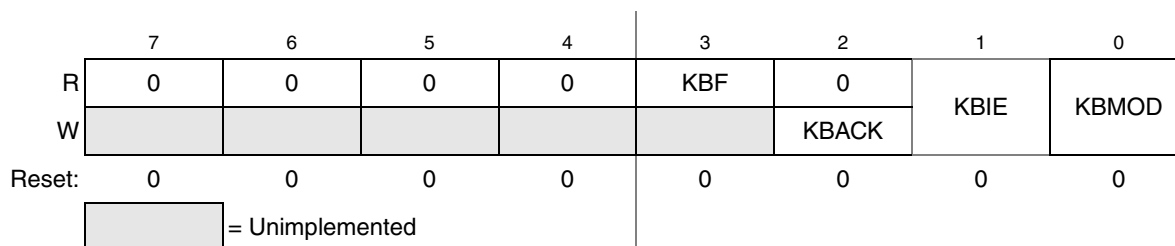


Figure 8-3. KBI Status and Control Register

Table 8-2. KBISC Register Field Descriptions

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

### 8.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.

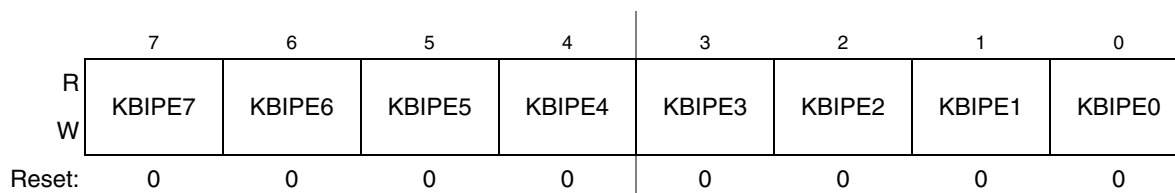


Figure 8-4. KBI Pin Enable Register

Table 8-3. KBIPE Register Field Descriptions

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

### 8.3.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

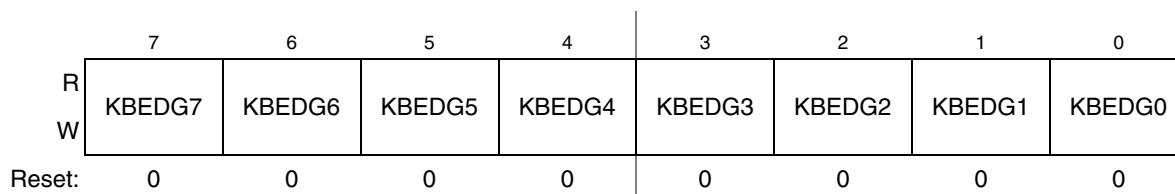


Figure 8-5. KBI Edge Select Register

Table 8-4. KBIES Register Field Descriptions

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

## 8.4 Functional Description

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

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

### 8.4.1 Edge Only Sensitivity

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

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

### 8.4.2 Edge and Level Sensitivity

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

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

### 8.4.3 KBI Pullup/Pulldown Resistors

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

### 8.4.4 KBI Initialization

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

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



## Chapter 9

# 8-Bit Serial Peripheral Interface (S08SPIV4)

### 9.1 Introduction

There are three 8-bit serial peripheral interface (SPI) modules in MC9S08GW64 series MCUs.

#### 9.1.1 AMR SPI0

SPI0 is designed for AMR operation. It's  $\overline{SS0}$ , SCLK0 and MISO0 in Slave mode or MOSI0 in Master mode are open drain and can be compatible with 5V devices. See [Section 2.3.8, “AMR Pins Compatible with 5 V Interface,”](#) for more information.

#### 9.1.2 SPI Clock Gating

The bus clock to the SPIx modules can be gated on and off using the SPIx bit in SCGC2. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the SPIx bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### NOTE

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it. For details on low-power mode operation, refer to [Table 6-5 in Chapter 6, “Parallel Input/Output Control.”](#)

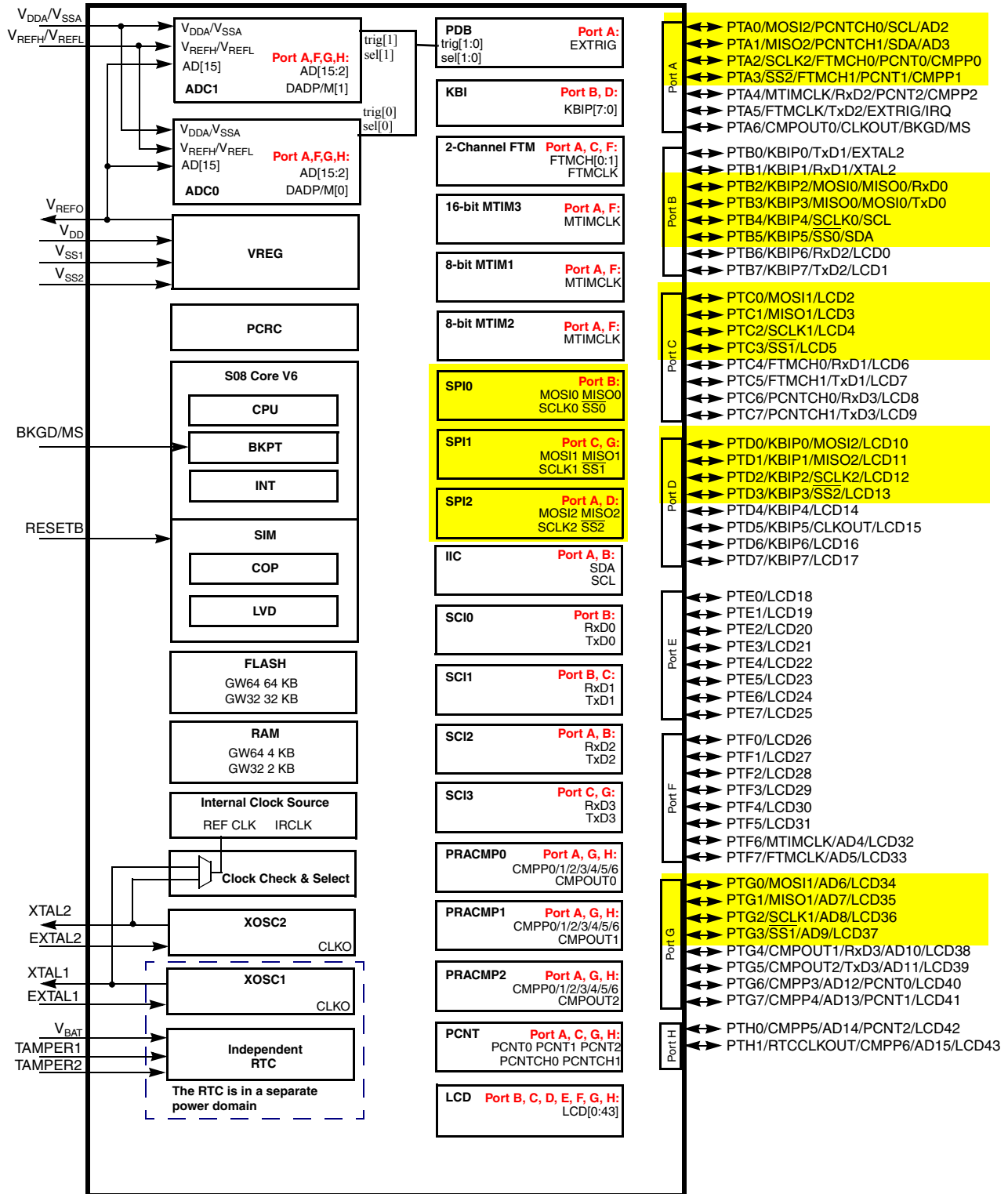


Figure 9-1. MC9S08GW64 Series Block Diagram Highlighting SPI Modules and Pins

### 9.1.3 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

### 9.1.4 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.

#### 9.1.4.1 SPI System Block Diagram

Figure 9-2 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ( $\overline{SS}$  pin). In this system, the master device has configured its  $\overline{SS}$  pin as an optional slave select output.

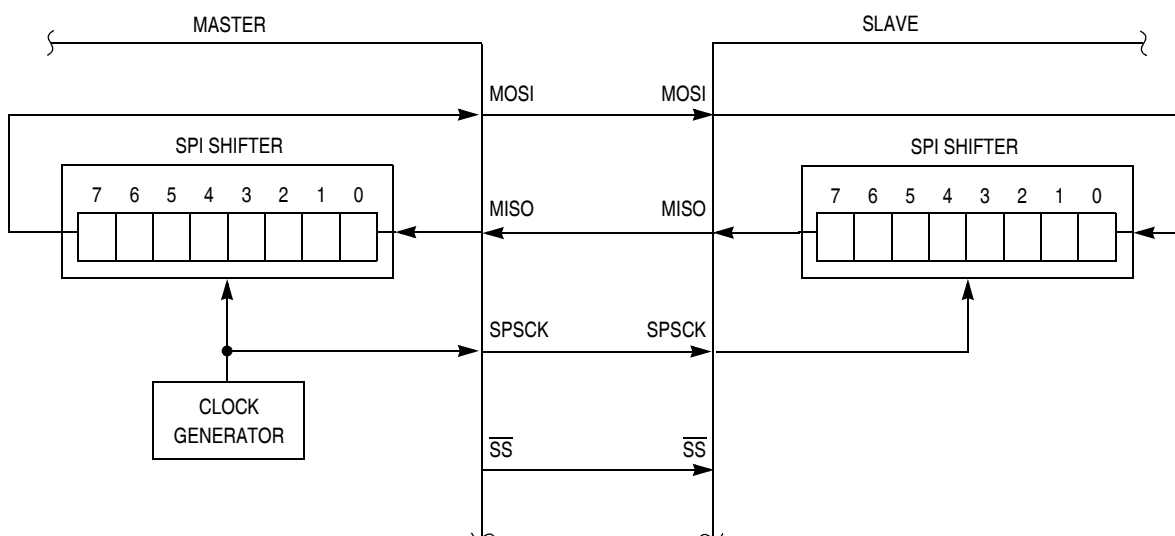


Figure 9-2. SPI System Connections

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 9-2](#) 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.

#### 9.1.4.2 SPI Module Block Diagram

[Figure 9-3](#) is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPID) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.





### 9.1.5 SPI Baud Rate Generation

As shown in [Figure 9-4](#), 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.

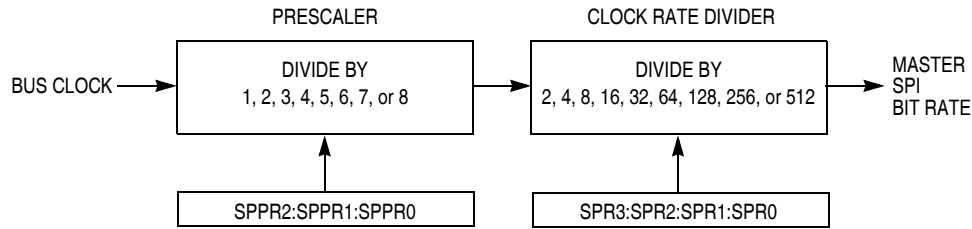


Figure 9-4. SPI Baud Rate Generation

## 9.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled ( $SPE = 0$ ), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 9.2.1 SPCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 9.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero ( $SPC0$ ) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and  $SPC0 = 0$ , this pin is the serial data input. If  $SPC0 = 1$  to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input ( $BIDIROE = 0$ ) or an output ( $BIDIROE = 1$ ). If  $SPC0 = 1$  and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 9.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero ( $SPC0$ ) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and  $SPC0 = 0$ , this pin is the serial data output. If  $SPC0 = 1$  to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input ( $BIDIROE = 0$ ) or an output ( $BIDIROE = 1$ ). If  $SPC0 = 1$  and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 9.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off ( $MODFEN = 0$ ), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and  $MODFEN = 1$ , the slave select output enable bit determines whether this pin acts as the mode fault input ( $SSOE = 0$ ) or as the slave select output ( $SSOE = 1$ ).

## 9.3 Modes of Operation

### 9.3.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

## 9.4 Register Definition

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data.

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 9.4.1 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								
Reset	0	0	0	0	0	1	0	0

Figure 9-5. SPI Control Register 1 (SPIC1)

Table 9-1. 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 9-1. 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 9.5.4, “SPI Clock Formats”</a> for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	<b>Clock Phase</b> — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to <a href="#">Section 9.5.4, “SPI Clock Formats”</a> for more details. 0 First edge on SPSCCK occurs at the middle of the first cycle of an 8-cycle data transfer 1 First edge on SPSCCK 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 9-2</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 9-2.  $\overline{SS}$  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.

**9.4.2 SPI Control Register 2 (SPIC2)**

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

	7	6	5	4	3	2	1	0
R	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

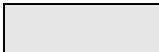
 = Unimplemented or Reserved

Figure 9-6. SPI Control Register 2 (SPIC2)

Table 9-3. SPIC2 Register Field Descriptions

Field	Description
4 MODFEN	<b>Master Mode-Fault Function Enable</b> — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to <a href="#">Table 9-2</a> for more details). 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	<b>Bidirectional Mode Output Enable</b> — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	<b>SPI Stop in Wait Mode</b> 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	<b>SPI Pin Control 0</b> — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

### 9.4.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

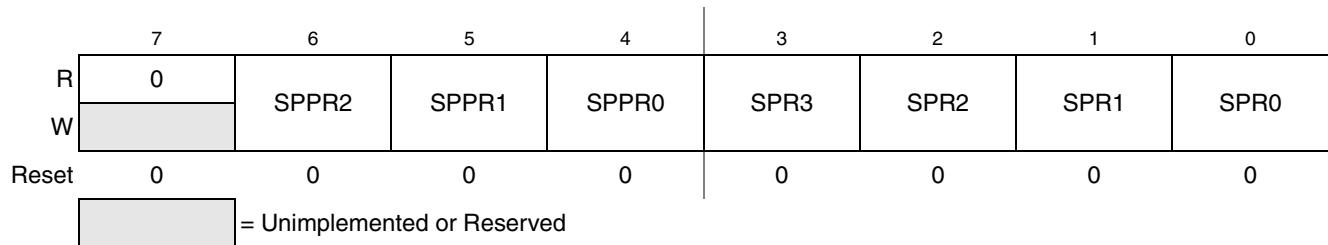


Figure 9-7. SPI Baud Rate Register (SPIBR)

Table 9-4. 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 9-5</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 9-4</a> ).
2:0 SPR[3:0]	<b>SPI Baud Rate Divisor</b> — This 4-bit field selects one of nine divisors for the SPI baud rate divider as shown in <a href="#">Table 9-6</a> . The input to this divider comes from the SPI baud rate prescaler (see <a href="#">Figure 9-4</a> ). The output of this divider is the SPI bit rate clock for master mode.

Table 9-5. 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 9-6. 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

#### 9.4.4 SPI Status Register (SPIS)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0. Writes have no meaning or effect.

	7	6	5	4	3	2	1	0
R	SPRF	0	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-8. SPI Status Register (SPIS)

Table 9-7. 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 $\overline{SS}$ 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

### 9.4.5 SPI Data Register (SPID)

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

Figure 9-9. 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.

## 9.5 Functional Description

### 9.5.1 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.

### 9.5.2 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 SPSCCK 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 SPSCCK, 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 SPSCCK-cycle delay. After the delay, SPSCCK 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 9.5.4, “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.

## 9.5.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register1 is clear.

- SPSCCK

In slave mode, SPSCCK 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 SPSCCK 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 SPSCCK 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 SPSCCK 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.

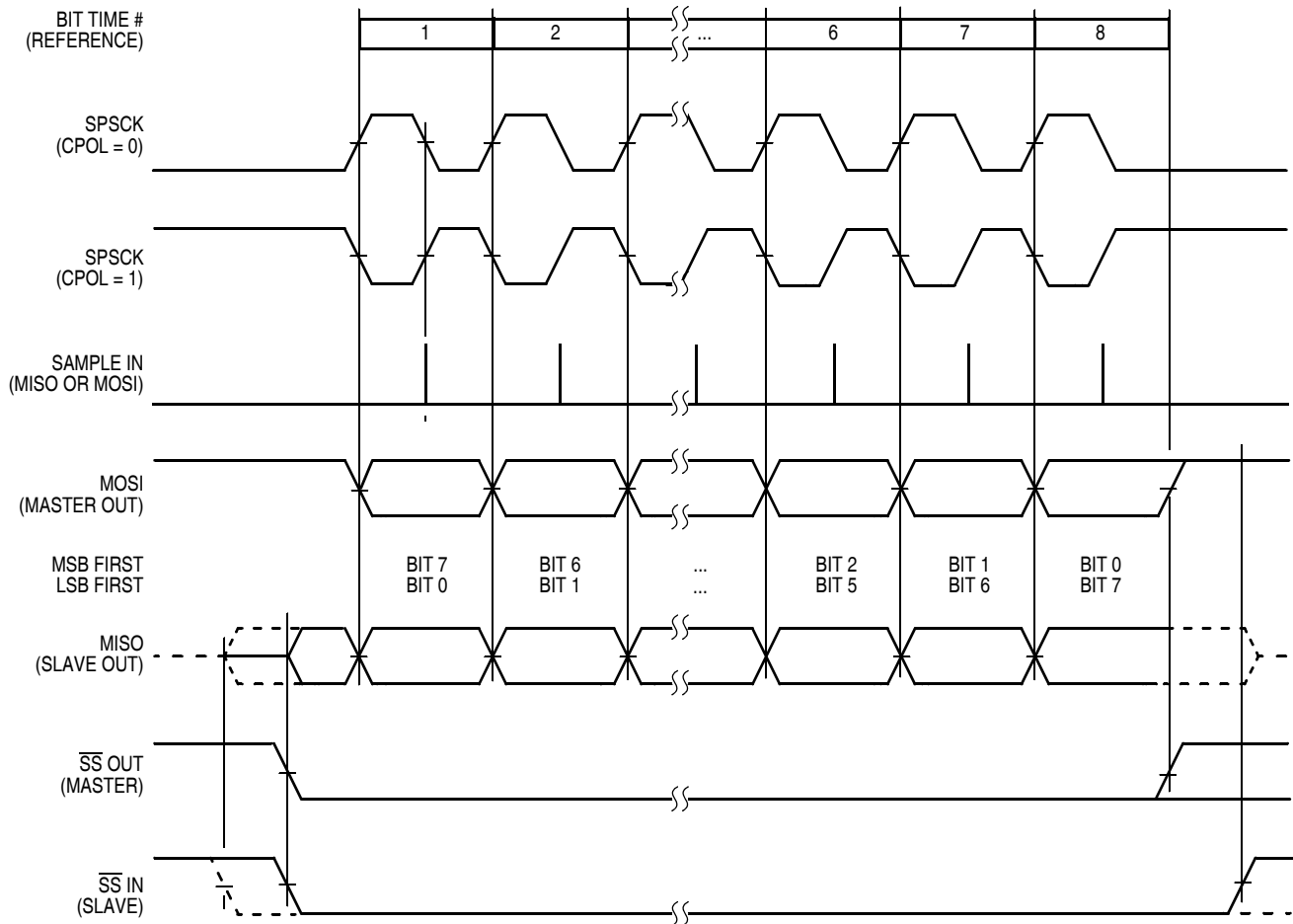
**NOTE**

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.

## 9.5.4 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 9-10 shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCCK edge and bit 8 ending one-half SPSCCK cycle after the sixteenth SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The 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 SPSCCK 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 9-10. SPI Clock Formats (CPHA = 1)**

When CPHA = 1, the slave begins to drive its MISO output when  $\overline{SS}$  goes to active low, but the data is not defined until the first SPSCCK edge. The first SPSCCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's  $\overline{SS}$  input is not required to go to its inactive high level between transfers.

Figure 9-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ( $\overline{SS}$  IN goes low), and bit 8 ends at the last SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the 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 SPSCCK 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.

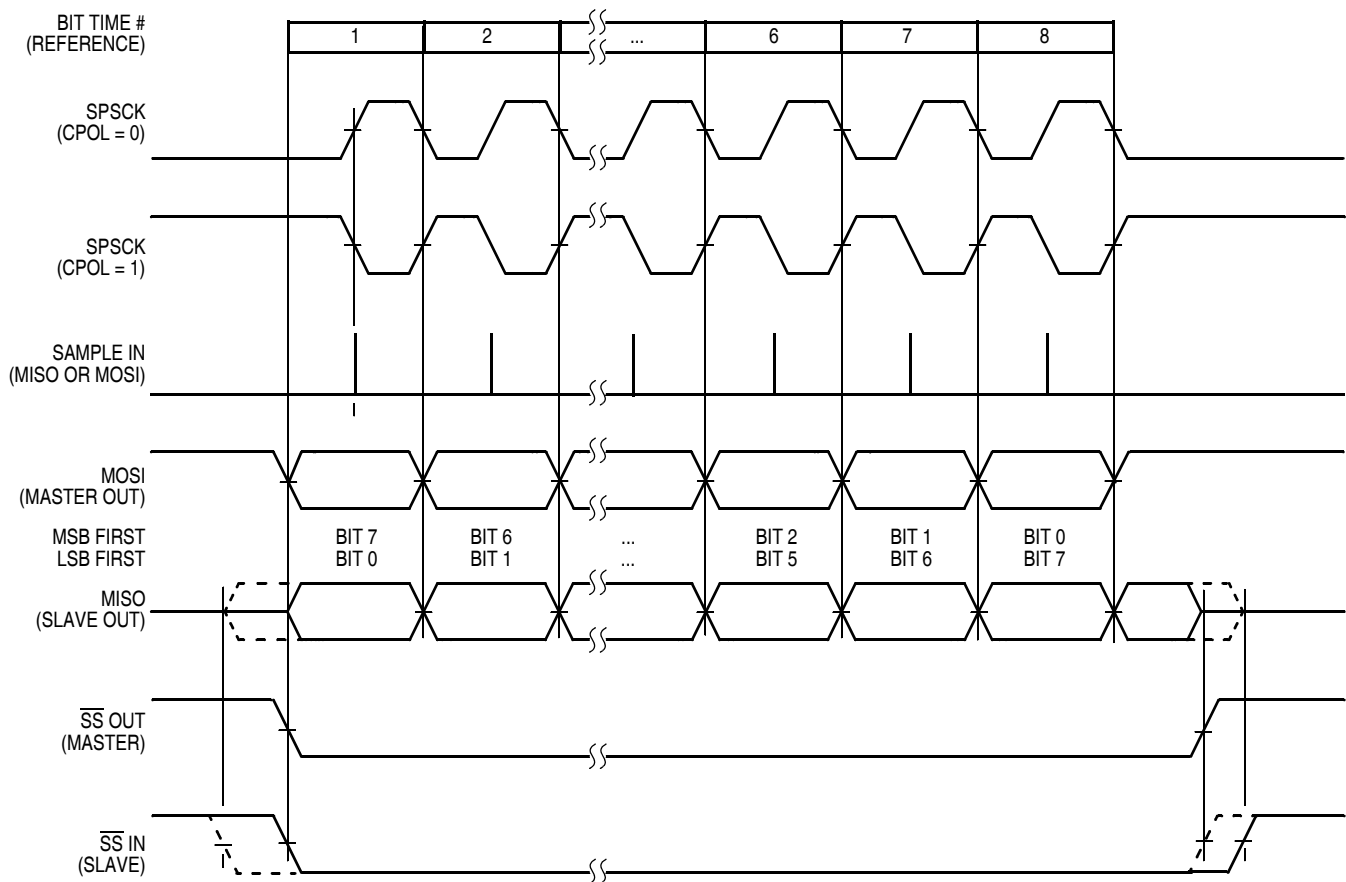


Figure 9-11. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when  $\overline{SS}$  goes to active low. The first SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's  $\overline{SS}$  input must go to its inactive high level between transfers.

## 9.5.5 Special Features

### 9.5.5.1 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 9-2.](#), “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.

### 9.5.5.2 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 9-8. Normal Mode and Bidirectional Mode**

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
<b>Normal Mode</b> SPC0 = 0		
<b>Bidirectional Mode</b> SPC0 = 1		

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 SPSCCK 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 SPSCCK 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.

**9.5.6 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).

**9.5.7 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 SPSCCK, 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 10

# Serial Communication Interface (S08SCIV4)

### 10.1 Introduction

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs.

There are four SCI modules in MC9S08GW64 series MCU. SCI0 is for AMR, whose TxD0 is open-drain and can be compatible with 5 V devices. SCI2 is a normal SCI with double drive strength to facilitate IR communication. SCI1 and SCI3 are normal SCIs.

#### 10.1.1 AMR SCI0

SCI0 is for AMR, whose TxD0 is open-drain and can be compatible with 5 V devices. See [Section 2.3.8, “AMR Pins Compatible with 5 V Interface,”](#) for more information.

#### 10.1.2 SCI1 and SCI2 Internal Connection

The SCI2 is double-drive strength SCI module on the TxD pin. Both SCI1 and SCI2 can connect to the FTM channel, PCNT channel, MTIMx output and PRACMP0 and PRACMP1 in order to facilitate IR communication modulation/demodulation. For more information, refer to [Section 2.3.10, “Interfacing the SCIs to Off-Chip Circuits.”](#)

#### 10.1.3 SCI Clock Gating

The bus clock to the SCIx module can be gated on and off using the SCIx bit in SCGC1. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the SCIx bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### NOTE

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it.

For details on low-power mode operation, refer to [Table 6-5 in Chapter 6 , “Parallel Input/Output Control”](#).

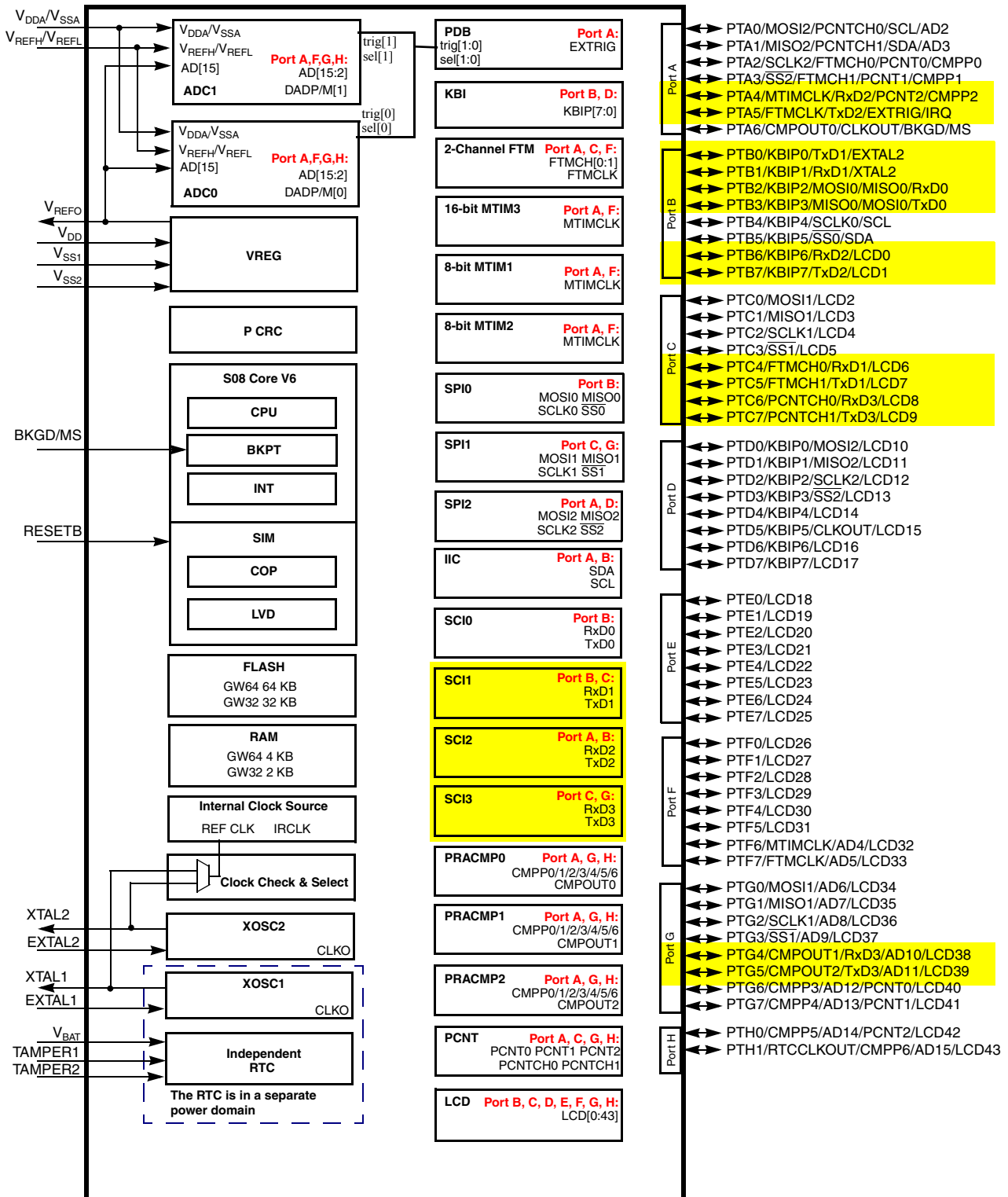


Figure 10-1. MC9S08GW64 Series Block Diagram Highlighting SCI Modules and Pins



## 10.1.4 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

## 10.1.5 Modes of Operation

See [Section 10.3, “Functional Description,”](#) for details concerning SCI operation in these modes:

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

## 10.1.6 Block Diagram

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

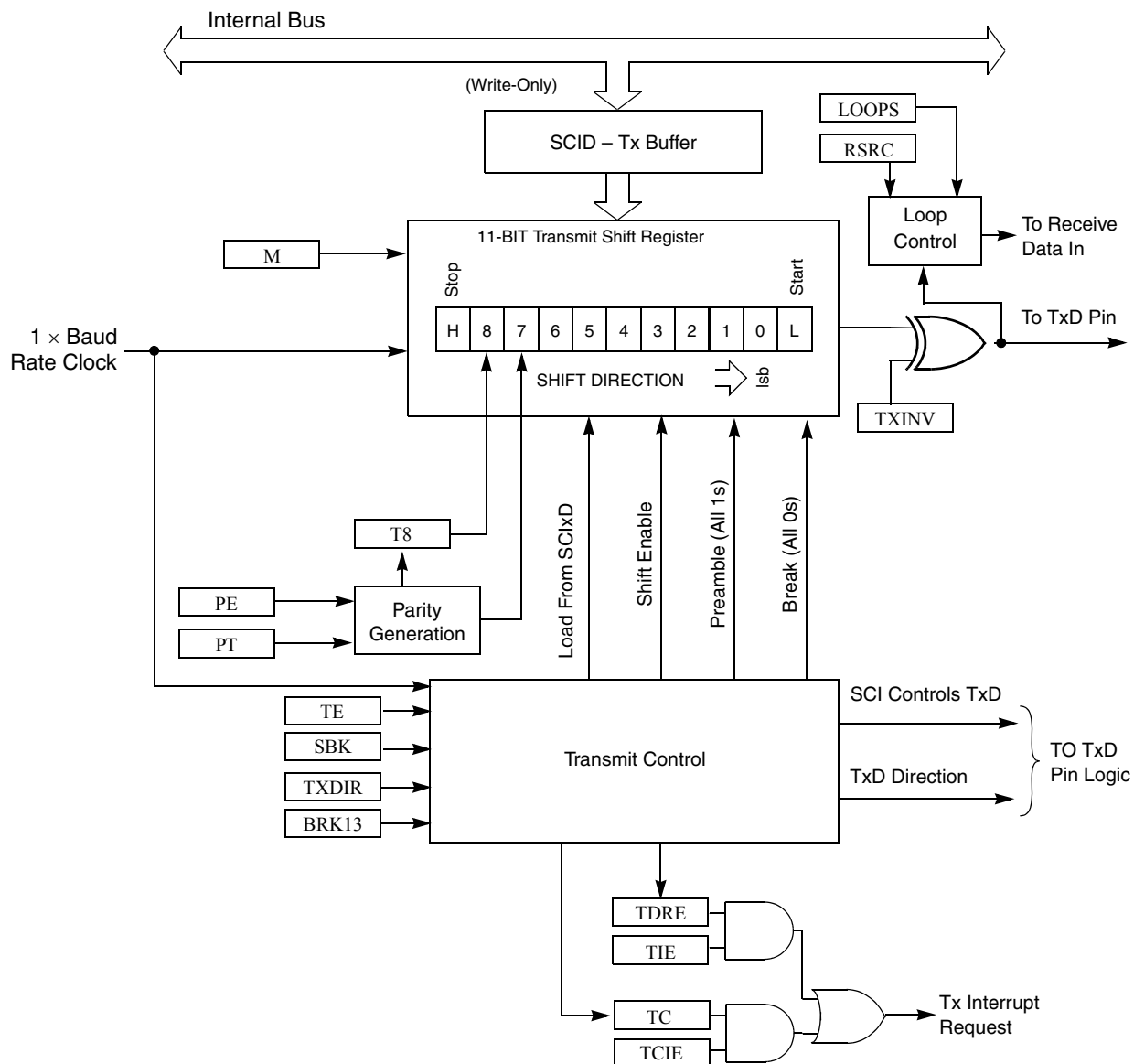


Figure 10-2. SCI Transmitter Block Diagram

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

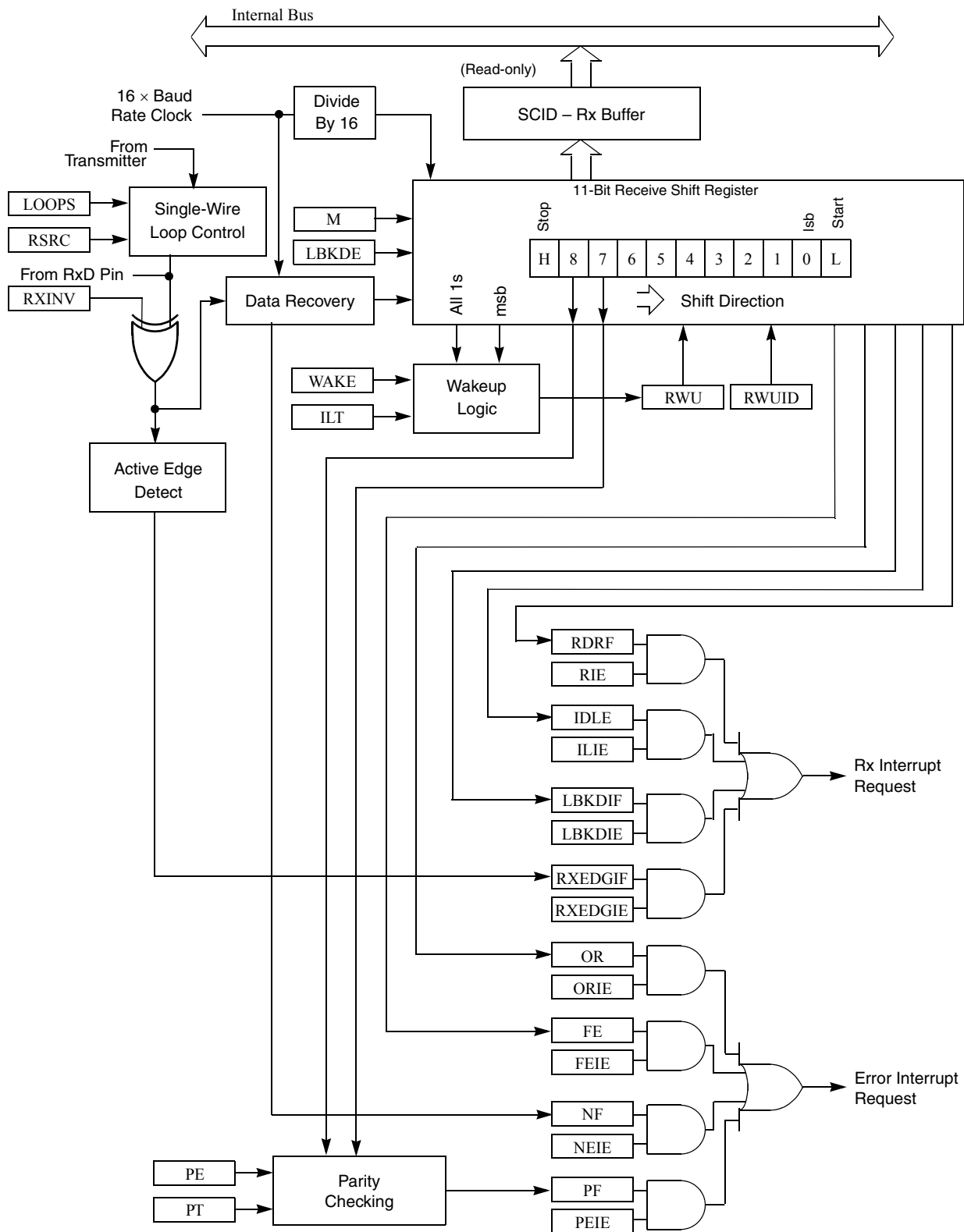


Figure 10-3. SCI Receiver Block Diagram

## 10.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the [memory](#) chapter of this document or the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 10.2.1 SCI Baud Rate Registers (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).

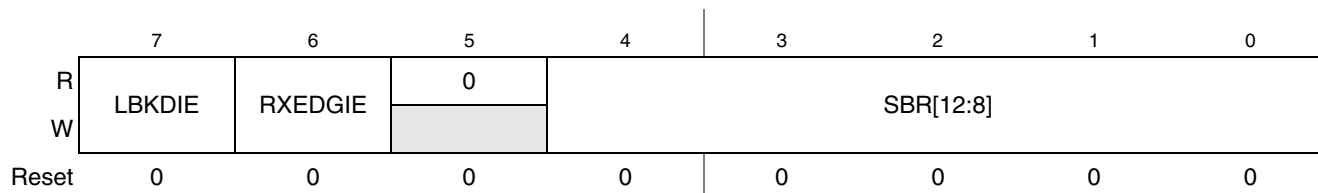


Figure 10-4. SCI Baud Rate Register (SCIxBDH)

Table 10-1. SCIxBDH Field Descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable (for LBKDIF) 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable (for RXEDGIF) 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4–0 SBR[12:8]	Baud Rate Modulo Divisor. The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR is cleared, the SCI baud rate generator is disabled to reduce supply current. When BR is 1 – 8191, the SCI baud rate equals $BUSCLK/(16 \times BR)$ . See also BR bits in <a href="#">Table 10-2</a> .

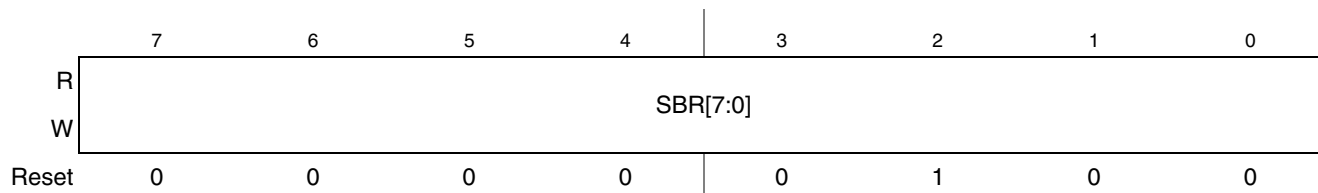


Figure 10-5. SCI Baud Rate Register (SCIxBDL)

**Table 10-2. SCIBDL Field Descriptions**

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

## 10.2.2 SCI Control Register 1 (SCIC1)

This read/write register controls various optional features of the SCI system.

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

**Figure 10-6. SCI Control Register 1 (SCIC1)****Table 10-3. SCIC1 Field Descriptions**

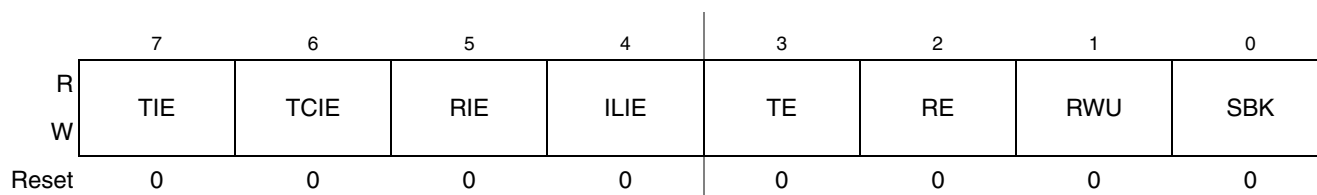
Field	Description
7 LOOPS	Loop Mode Select. Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS is set, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See <a href="#">RSRC</a> bit.) RxD pin is not used by SCI.
6 SCISWAI	SCI Stops in Wait Mode 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	Receiver Source Select. This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS is set, 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 is set, RSRC is cleared, selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Normal — start + 8 data bits (lsb first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (lsb first) + 9th data bit + stop.
3 WAKE	Receiver Wakeup Method Select. Refer to <a href="#">Section 10.3.3.2, “Receiver Wakeup Operation”</a> for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	Idle Line Type Select. Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to <a href="#">Section 10.3.3.2.1, “Idle-Line Wakeup”</a> for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

**Table 10-3. SC1xC1 Field Descriptions (continued)**

Field	Description
1 PE	Parity Enable. Enables hardware parity generation and checking. When parity is enabled, the most significant bit (msb) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type. Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

### 10.2.3 SCI Control Register 2 (SC1xC2)

This register can be read or written at any time.

**Figure 10-7. SCI Control Register 2 (SC1xC2)****Table 10-4. SC1xC2 Field Descriptions**

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

Table 10-4. SC1xC2 Field Descriptions (continued)

Field	Description
3 TE	<p>Transmitter Enable</p> <p>0 Transmitter off.</p> <p>1 Transmitter on.</p> <p>TE must be 1 to use the SCI transmitter. When TE is set, the SCI forces the TxD pin to act as an output for the SCI system.</p> <p>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).</p> <p>TE can also queue an idle character by clearing TE then setting TE while a transmission is in progress. Refer to <a href="#">Section 10.3.2.1, “Send Break and Queued Idle”</a> for more details.</p> <p>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.</p>
2 RE	<p>Receiver Enable. When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS is set the RxD pin reverts to being a general-purpose I/O pin even if RE is set.</p> <p>0 Receiver off.</p> <p>1 Receiver on.</p>
1 RWU	<p>Receiver Wakeup Control. This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is 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 10.3.3.2, “Receiver Wakeup Operation,”</a> for more details.</p> <p>0 Normal SCI receiver operation.</p> <p>1 SCI receiver in standby waiting for wakeup condition.</p>
0 SBK	<p>Send Break. Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK is set. 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 10.3.2.1, “Send Break and Queued Idle”</a> for more details.</p> <p>0 Normal transmitter operation.</p> <p>1 Queue break character(s) to be sent.</p>

## 10.2.4 SCI Status Register 1 (SC1xS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) clear these status flags.

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

Figure 10-8. SCI Status Register 1 (SC1xS1)

Table 10-5. SC1xS1 Field Descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag. TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SC1xS1 with TDRE set and then write to the SCI data register (SC1xD). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	Transmission Complete Flag. TC is set out of reset and when TDRE is set 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 SC1xS1 with TC set and then doing one of the following: <ul style="list-style-type: none"> <li>• Write to the SCI data register (SC1xD) 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 SC1xC2</li> </ul>
5 RDRF	Receive Data Register Full Flag. RDRF becomes set when a character transfers from the receive shifter into the receive data register (SC1xD). To clear RDRF, read SC1xS1 with RDRF set and then read the SCI data register (SC1xD). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	Idle Line Flag. IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. 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 is set, the receiver doesn't start counting idle bit times until after the stop bit. 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 SC1xS1 with IDLE set and then read the SCI data register (SC1xD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE is set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	Receiver Overrun Flag. OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SC1xD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SC1xD. To clear OR, read SC1xS1 with OR set and then read the SCI data register (SC1xD). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	Noise Flag. The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF is set at the same time as RDRF is set for the character. To clear NF, read SC1xS1 and then read the SCI data register (SC1xD). 0 No noise detected. 1 Noise detected in the received character in SC1xD.



**Table 10-5. SCIS1 Field Descriptions (continued)**

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

### 10.2.5 SCI Status Register 2 (SCIS2)

This register contains one read-only status flag.

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

**Figure 10-9. SCI Status Register 2 (SCIS2)****Table 10-6. SCIS2 Field Descriptions**

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag. LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag. RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a 1 to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV <sup>1</sup>	Receive Data Inversion. Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	Receive Wake Up Idle Detect. RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length. BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

**Table 10-6. SCIxS2 Field Descriptions (continued)**

Field	Description
1 LBKDE	LIN Break Detection Enable. LBKDE selects a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	Receiver Active Flag. RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

<sup>1</sup> Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold 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 running 14% faster than the master. This would trigger normal break detection circuitry designed to detect a 10-bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

## 10.2.6 SCI Control Register 3 (SClxC3)

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

**Figure 10-10. SCI Control Register 3 (SClxC3)****Table 10-7. SClxC3 Field Descriptions**

Field	Description
7 R8	Ninth Data Bit for Receiver. When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the msb of the buffered data in the SClxD register. When reading 9-bit data, read R8 before reading SClxD because reading SClxD completes automatic flag clearing sequences that could allow R8 and SClxD to be overwritten with new data.
6 T8	Ninth Data Bit for Transmitter. When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the msb of the data in the SClxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SClxD is written so T8 should be written (if it needs to change from its previous value) before SClxD 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 SClxD is written.
5 TXDIR	TxD Pin Direction in Single-Wire Mode. When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

**Table 10-7. SCIC3 Field Descriptions (continued)**

Field	Description
4 TXINV <sup>1</sup>	Transmit Data Inversion. Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	Overrun Interrupt Enable. This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR is set.
2 NEIE	Noise Error Interrupt Enable. This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF is set.
1 FEIE	Framing Error Interrupt Enable. This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE is set.
0 PEIE	Parity Error Interrupt Enable. This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF is set.

<sup>1</sup> Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

## 10.2.7 SCI Data Register (SCID)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

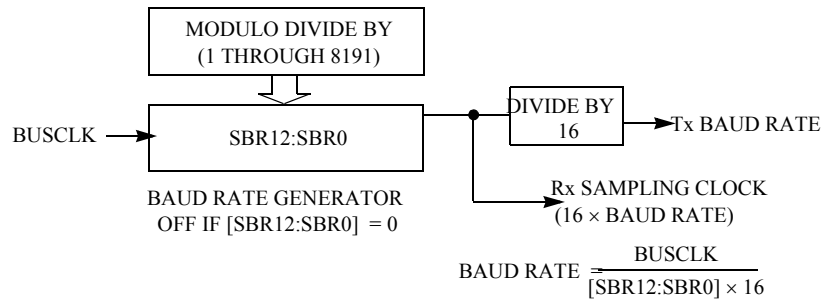
**Figure 10-11. SCI Data Register (SCID)**

## 10.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

### 10.3.1 Baud Rate Generation

As shown in [Figure 10-12](#), the clock source for the SCI baud rate generator is the bus-rate clock.



**Figure 10-12. SCI Baud Rate Generation**

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

The MCU resynchronizes to bit boundaries on every high-to-low transition. 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.

### 10.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in [Figure 10-2](#).

The transmitter output (TxD) idle state defaults to logic high (TXINV is cleared following reset). The transmitter output is inverted by setting TXINV. 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 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, assume M is cleared, 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 in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

### 10.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIx2 sends break characters 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. 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 remains 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 are 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 is cleared, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE is cleared, set the general-purpose I/O controls so the pin shared with TxD is an output driving a logic 1. This ensures that the TxD line looks like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

**Table 10-8. 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

### 10.3.3 Receiver Functional Description

In this section, the receiver block diagram ([Figure 10-3](#)) is 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. The receiver is enabled by setting the RE bit in SCIx2. 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 •, “8- and 9-bit data modes”](#). For the remainder of this discussion, assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF)

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading SCiXD. The RDRF flag is cleared automatically by a two-step sequence normally satisfied in the course of the user's program that manages receive data. Refer to [Section 10.3.4, "Interrupts and Status Flags,"](#) for more details about flag clearing.

### 10.3.3.1 Data Sampling Technique

The SCI receiver uses a  $16\times$  baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The  $16\times$  baud rate clock divides 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) is set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges. 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 remains set.

### 10.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message 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 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.

#### 10.3.3.2.1 Idle-Line Wakeup

When wake is cleared, 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 that sets the RDRF flag and generates 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 is cleared, 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 is set, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

#### 10.3.3.2.2 Address-Mark Wakeup

When wake is set, 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 when M is cleared and ninth bit when M is set).

Address-mark wakeup allows messages to contain idle characters, but requires the msb be reserved for use in address frames. The logic 1 msb of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case, the character with the msb set is received even though the receiver was sleeping during most of this character time.

### 10.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF, and LBKDIF events. 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 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 can optionally 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 is requested when TDRE is set. 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 is requested when TC is set. 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 cleared.

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  $RDRF$  is set 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 is set 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) — are 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 is set instead of the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the  $RxD$  serial data input pin causes the  $RXEDGIF$  flag to set. The  $RXEDGIF$  flag is cleared by writing a 1 to it. This function does depend on the receiver being enabled ( $RE = 1$ ).

### 10.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

#### 10.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in  $SCIxC1$ . 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.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.



### 10.3.5.2 Stop Mode Operation

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

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

The receive input active edge detect circuit remains 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).

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

### 10.3.5.3 Loop Mode

When LOOPS is set, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

### 10.3.5.4 Single-Wire Operation

When LOOPS is set, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode implements 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 is cleared, 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 is set, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.



# Chapter 11

## LCD Module (S08LCDLPV1)

### 11.1 Introduction

On the MC9S08GW64 series, the LCD module (LCD) controls up to 44 LCD pins to generate the waveforms necessary to drive a liquid crystal display. On the 80 pin package, the 44 LCD pins can be used to generate 4x40 or 8x36 configuration. On the 64-pin package there are 32 LCD pins available for driving 4x28 or 8x24 configurations.

#### NOTE

$V_{LCD}$  pin is not available in MC9S08GW64 series, please ignore any content relates with  $V_{LCD}$ .

#### NOTE

MC9S08GW64 series support only up to 44 LCD pins (LCD[43:0]), please ignore the other LCD pins in the following sections of this chapter.

For GPIO muxed with LCD pins, full complimentary or open drain drive is controlled by the LCD controller. When LCD pins are configured as open drain GPIOs, then the internal pullup is not disabled in output mode and is controlled by the GPIO pull control register. This can cause some leakage from the pads if a pullup is enabled and a zero is being driven.

For some peripheral ports shared with LCD pad such as SPI, SCI and CLKOUT, in some cases, such as VLL3 powered by charge pump, it may not reach a high frequency i.e. 20 MHz.

In the Charge Pump mode when the VLL3 is being generated internally, the LCD/digital functional I/O that presents digital functionality only can drive limited loads which is less than respective ones. If the VLL3 is being provided externally, I/O can be used to drive respective loads.

When LCD clocked from OSCOUT1 and run in the STOP3 mode, please never do not let the VDD drop under 1.8V, else you need enable LVD function in the STOP3 mode.

#### 11.1.1 LCD Clock Sources

The LCD module on MC9S08GW64 Series can be clocked from the OSCOUT1 or OSCOUT2.

## 11.1.2 LCD Modes of Operation

The LCD module can be configured to operate in stop modes by setting the LCDSTP bit. All clock sources are available in all modes except stop2. In stop2 mode, only OSCOUT1 or OSCOUT2 is available.

## 11.1.3 LCD Status after Stop2 Wakeup

The LCD control registers are set to their default state. These registers must be re-initialized before setting the PPDACK. The contents of the LCD data registers (LCD waveform, LCD pin enable, and LCD backplane enable) are retained.

## 11.1.4 LCD Clock Gating

The bus clock to the LCD can be gated on and off using the LCD bit in SCGC2. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the LCD bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

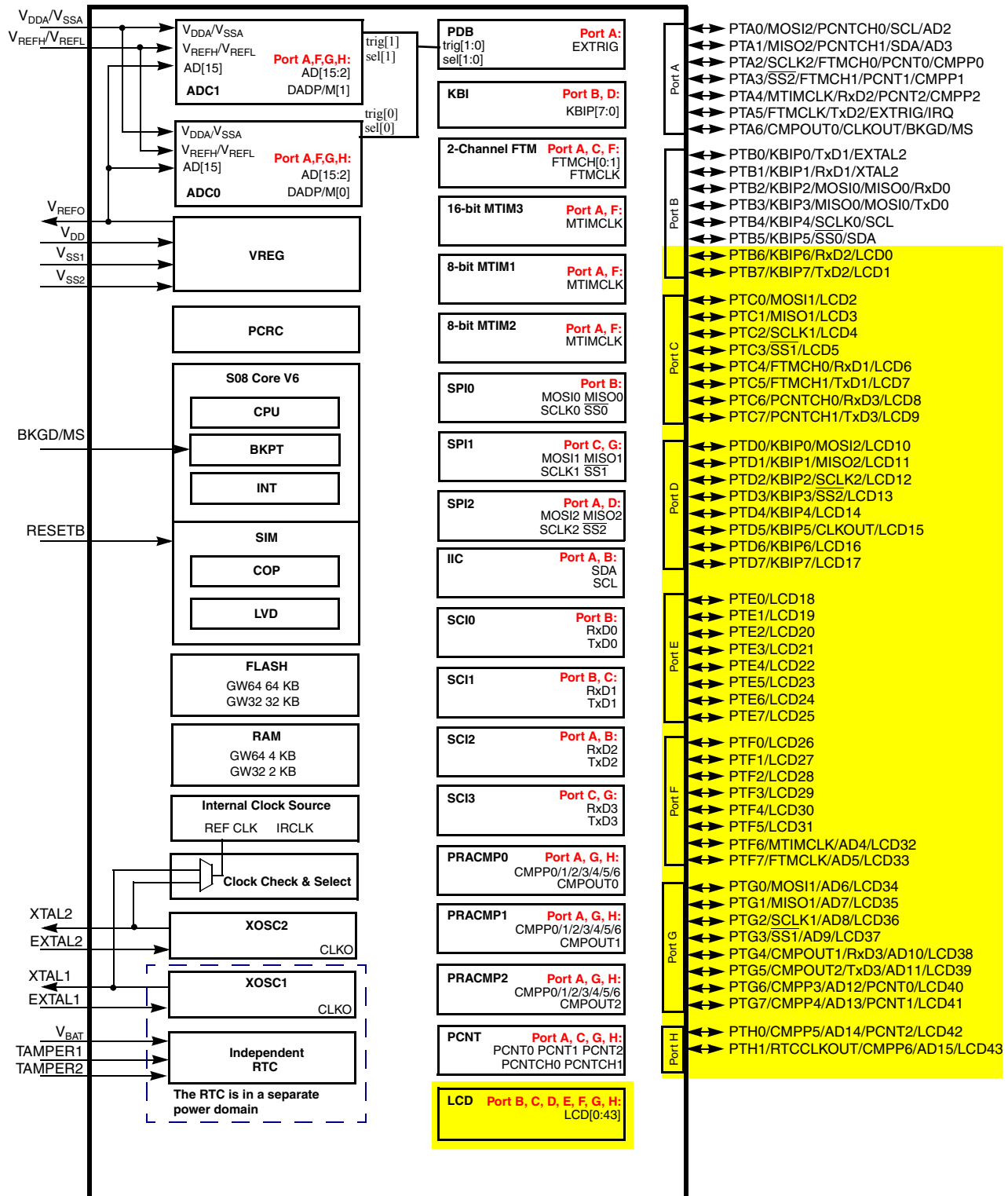


Figure 11-1. MC9S08GW64 Series Block Diagram Highlighting LCD Module and Pins

### 11.1.5 Features

The LCD module driver features include:

- LCD waveforms functional in  $LP_{Run}$ ,  $LP_{Wait}$ , wait, stop2 and stop3 low-power modes
- 64 LCD (LCD[63:0]) pins with selectable frontplane/backplane configuration
  - Generate up to 63 frontplane signals
  - Generate up to 8 backplanes signals
- Programmable LCD frame frequency
- Programmable blink modes and frequency
  - All segments blank during blink period
  - Alternate display for each LCD segment in x4 or less mode
  - Blink operation in low-power modes
- Programmable LCD power supply switch, making it an ideal solution for battery-powered and board-level applications
  - Charge pump requires only four external capacitors
  - Internal LCD power using  $V_{DD}$  (1.8 to 3.6 V)
  - External  $V_{LCD}$  power supply option (.9 to 1.8 V)
  - Internal  $V_{IREG}$  regulated power supply option for 3 or 5V LCD glass
  - External  $V_{LL3}$  power supply option (3V) Internal regulated voltage source with a 4-bit trim register to apply contrast control
- Integrated charge pump for generating LCD bias voltages
  - Hardware configurable to drive 3-V or 5-V LCD panels
  - On-chip generation of bias voltages
- Waveform storage registers LCDWF
- Backplane reassignment to assist in vertical scrolling on dot-matrix displays
- Software configurable LCD frame frequency interrupt
- Internal ADC channels are connected to  $V_{LL1}$  and  $V_{LCD}$  to monitor their magnitudes. This feature allows software to adjust the contrast.

### 11.1.6 Modes of Operation

The LCD module supports the following operation modes:

**Table 11-1. LCD-Module Operation Modes**

Mode	Operation
Stop2	<p>Depending on the state of the LCDSTP bit, the LCD module can operate an LCD panel in stop2 mode. If LCDSTP = 1, LCD module clock generation is turned off and the LCD module enters a power conservation state and is disabled. If LCDSTP = 0, the LCD module can operate an LCD panel in stop2, and the LCD module continues to display the current LCD panel contents based on the LCD operation prior to the stop2 event.</p> <p>If the LCD is enabled in stop2, the selected LCD clock source, OSCOUT, must be enabled to operate in stop2.</p> <p>The LCD frame interrupt does not cause the MCU to exit stop2.</p>
Stop3	<p>Depending on the state of the LCDSTP bit, the LCD module can operate an LCD panel in stop3 mode. If LCDSTP = 1, LCD module clock generation is turned off and the LCD module enters a power conservation state and is disabled. If LCDSTP = 0, the LCD module can operate an LCD panel in stop3, and the LCD module continues displaying the current LCD panel contents based on the LCD operation prior to the stop3 event.</p> <p>If the LCD is enabled in stop3, the selected LCD clock source, OSCOUT or the Alternate Clock, must be enabled to operate in stop3.</p> <p>In stop3 mode the LCD frame interrupt can cause the MCU to exit stop3.</p>
LP <sub>Wait</sub> , Wait	<p>Depending on the configuration, the LCD module can operate an LCD panel in wait mode. If LCDWAI = 1, the LCD module clock generation is turned off and the LCD module enters a power-conservation state and is disabled. If LCDWAI = 0, the LCD module can operate an LCD panel in wait, and the LCD module continues displaying the current LCD panel contents base on the LCDWF registers.</p> <p>In wait mode, the LCD frame interrupt can cause the MCU to exit wait.</p>

Stop2 provides the lowest power consumption state where the LCD module is functional. To operate the LCD in stop2 mode, use an external crystal.

## 11.1.7 Block Diagram

Figure 11-2 is a block diagram of the LCD module.

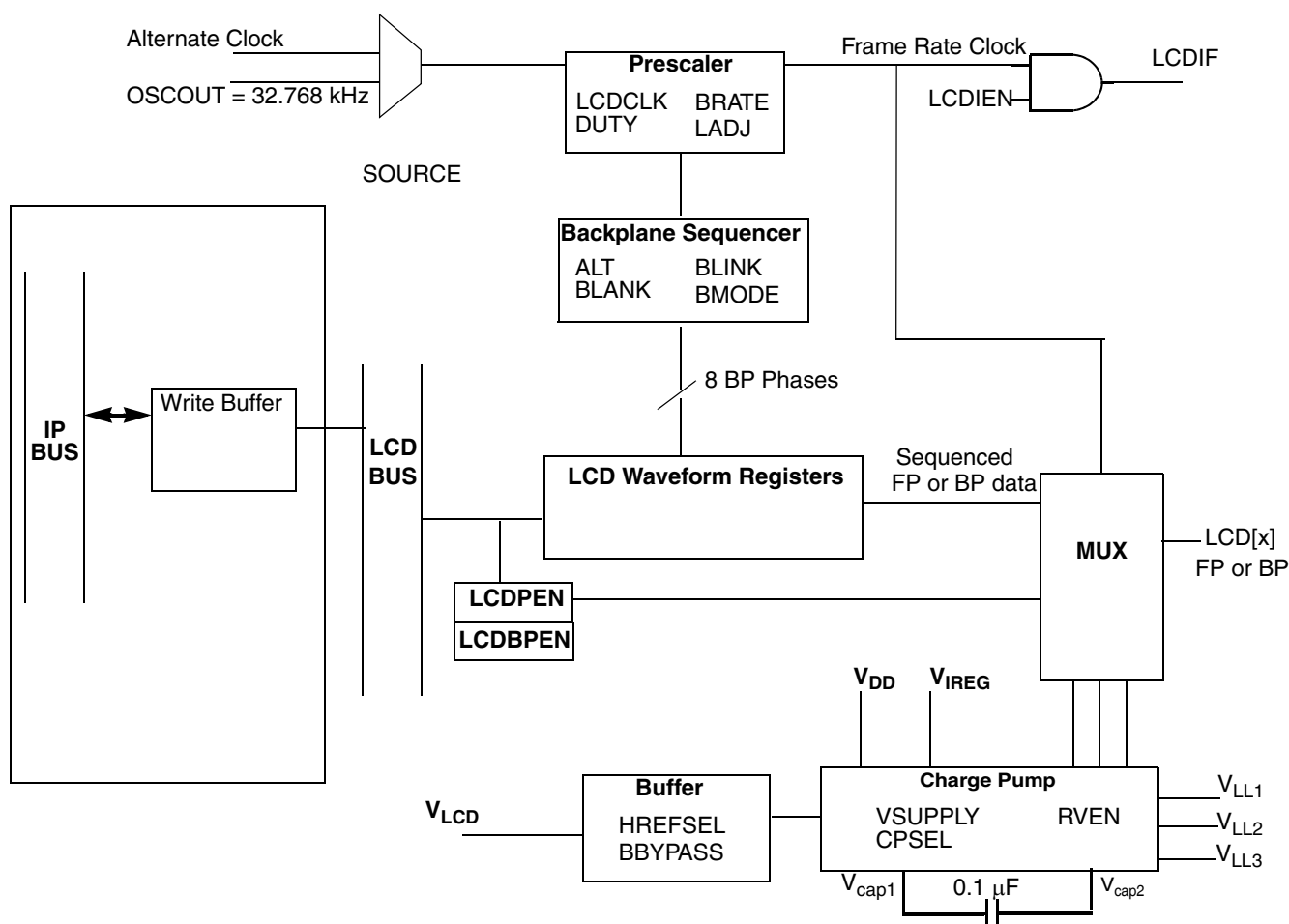


Figure 11-2. LCD Driver Block Diagram

## 11.2 External Signal Description

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

Table 11-2. Signal Properties

Name	Port	Function	Reset State
64 LCD frontplane/backplane	LCD[63:0]	Switchable frontplane/backplane driver that connects directly to the display LCD[63:0] can operate as GPIO pins	High impedance
LCD voltage	V <sub>LCD</sub>	LCD supply voltage	—



Table 11-2. Signal Properties (continued)

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

### 11.2.1 LCD[63:0]

When LCD functionality is enabled by the PEN[63:0] bits in the LCDPEN registers, the corresponding LCD[63:0] pin will generate a frontplane or backplane waveform depending on the configuration of the backplane-enable bit field (BPEN[63:0]).

### 11.2.2 $V_{LCD}$

$V_{LCD}$  can be a source for LCD module-waveform generation.  $V_{LCD}$  is connected to a switch capacitor charge pump DC/DC converter that can generate double or triple  $V_{LCD}$  to support 3-V or 5-V LCD glass. A voltage source in the range from .9V to 1.8 V can be placed on  $V_{LCD}$ .  $V_{LCD}$  is also connected internally to an ADC channel in order to monitor the  $V_{LCD}$  magnitude.

### 11.2.3 $V_{LL1}$ , $V_{LL2}$ , $V_{LL3}$

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

### 11.2.4 $V_{cap1}$ , $V_{cap2}$

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

## 11.3 Register Definition

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

### 11.3.1 LCD Control Register 0 (LCDC0)

	7	6	5	4	3	2	1	0
R	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
W								
Reset	0	0	0	0	0	0	1	1


 = Unimplemented or Reserved

Figure 11-3. LCD Control Register 0 (LCDC0)

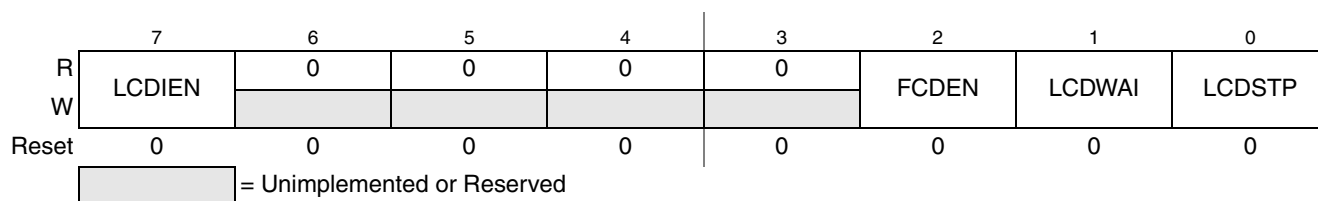
Read: anytime

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

**Table 11-3. LCDC0 Field Descriptions**

Field	Description
7 LCDEN	<b>LCD Driver Enable</b> — LCDEN starts LCD-module-waveform generator. 0 All frontplane and backplane pins are disabled. The LCD module system is also disabled, and all LCD waveform generation clocks are stopped. $V_{LL3}$ is connected to $V_{DD}$ internally 1 LCD module driver system is enabled and frontplane and backplane waveforms are generated. All LCD pins enabled using the LCD pin enable register (LCDPEN[x]) will output an LCD module driver waveform. The backplane pins will output an LCD module driver backplane waveform based on the settings of DUTY[2:0]. Chargepump or resistor bias is enabled.
6 SOURCE	<b>LCD Clock Source Select</b> — The LCD module has two possible clock sources. This bit is used to select which clock source is the basis for LCDCLK. 0 Selects the OSCOUT (external clock reference) as the LCD clock source. 1 Selects the alternate clock as the LCD clock source.
5:3 LCLK[2:0]	<b>LCD Clock Prescaler</b> — Used as a clock divider to generate the LCD module frame frequency as shown in <a href="#">Equation 11-1</a> . LCD-module-duty-cycle configuration is used to determine the LCD module frame frequency. LCD module frame frequency calculations are provided in <a href="#">Table 11-13/p.266</a> . <div style="text-align: right;"><b>Eqn. 11-1</b></div> $\text{LCD Module Frame Frequency} = \frac{\text{LCDCLK}}{((\text{DUTY}+1) \times 8 \times (4 + \text{LCLK}[2:0]) \times Y)}$ <p style="text-align: right;">where <math>30 &lt; \text{LCDCLK} &lt; 39.063 \text{ kHz}</math>            where <math>Y = 2, 2, 3, 3, 4, 5, 8, 16</math> chosen by module duty cycle configuration</p>
2:0 DUTY[2:0]	<b>LCD Duty Select</b> — DUTY[2:0] bits select the duty cycle of the LCD module driver. 000 Use 1 BP (1/1 duty cycle). 001 Use 2 BP (1/2 duty cycle). 010 Use 3 BP (1/3 duty cycle). 011 Use 4 BP (1/4 duty cycle). (Default) 100 Use 5 BP (1/5 duty cycle). 101 Use 6 BP (1/6 duty cycle). 110 Use 7 BP (1/7 duty cycle). 111 Use 8 BP (1/8 duty cycle).

### 11.3.2 LCD Control Register 1 (LCDC1)

**Figure 11-4. LCD Control Register 1 (LCDC1)**

Read: anytime Write: anytime

Table 11-4. LCDC1 Field Descriptions

Field	Description
7 LCDIEN	<b>LCD Module Frame Frequency Interrupt Enable</b> — Enables an LCD interrupt event that coincides with the LCD module frame frequency. 0 No interrupt request is generated by this event. 1 The start of the LCD module frame causes an LCD module frame frequency interrupt request.
2 FCDEN	<b>Full Complementary Drive Enable</b> — This bit allows GPIO that are shared with LCD pins to operate as full complementary if the other conditions necessary have been met. The other conditions are: VSUPPLY = 11 and RVEN = 0. 0 GPIO shared with LCD operate as open drain outputs, input levels and internal pullup resistors are referenced to $V_{DD}$ . 1 If VSUPPLY = 11 and RVEN = 0, GPIO shared with LCD operate as full complementary outputs. Input levels and internal pullup resistors are referenced to $V_{LL3}$ .
1 LCDWAI	<b>LCD Module Driver and Charge Pump Stop While in Wait Mode</b> 0 Allows the LCD driver and charge pump to continue running during wait mode. 1 Disables the LCD driver and charge pump when MCU goes into wait mode.
0 LCDSTP	<b>LCD Module Driver and Charge Pump Stop While in Stop2 or Stop3 Mode</b> 0 Allows LCD module driver and charge pump to continue running during stop2 or stop3. 1 Disables LCD module driver and charge pump when MCU goes into stop2 or stop3.

### 11.3.3 LCD Voltage Supply Register (LCDSUPPLY)

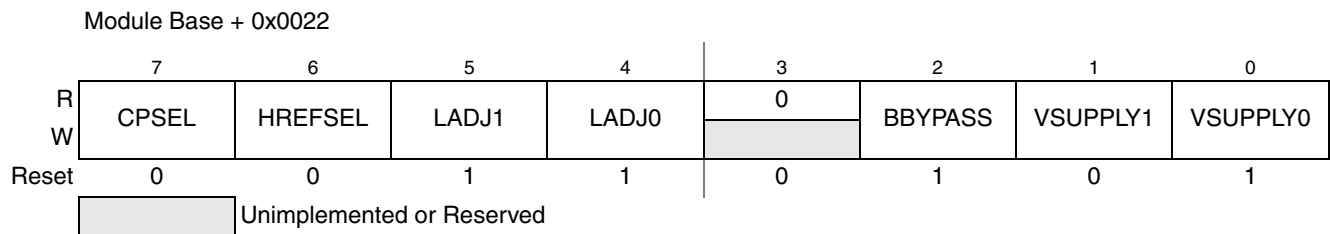


Figure 11-5. LCD Voltage Supply Register (LCDSUPPLY)

Read: anytime

Write: anytime.

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

Table 11-5. LCDSUPPLY Field Descriptions

Field	Description
7 CPSEL	<b>Charge Pump or Resistor Bias Select</b> — Selects LCD module charge pump or a resistor network to supply the LCD voltages $V_{LL1}$ , $V_{LL2}$ , and $V_{LL3}$ . See Figure 11-16 for more detail. 0 LCD charge pump is disabled. Resistor network selected (The internal 1/3-bias is forced.) 1 LCD charge pump is selected. Resistor network disabled (The internal 1/3-bias is forced.)
6 HREFSEL	<b>High Reference Select</b> — When using the $V_{LCD}$ or $V_{IREG}$ inputs, this bit configures internal circuits to supply $V_{LL1}$ . 0 Divide input, $V_{LCD\ IN} = V_{LCDEXT} * 2/3$ , $V_{IREG} = 1.0V$ 1 Do not divide the input, $V_{LCD\ IN} = V_{LCDEXT} * 3/3$ , $V_{IREG} = 1.67\ V$
5:4 LADJ[1:0]	<b>LCD Module Load Adjust</b> — The LCD load adjust bits are used to configure the LCD module to handle different LCD glass capacitance.  For CPSEL = 1 Adjust the clock source for the charge pump. Higher loads require higher charge pump clock rates. 00 - Fastest clock source for charge pump (LCD glass capacitance 8000pf or lower) 01 - Intermediate clock source for charge pump (LCD glass capacitance 6000pf or lower) 10 - Intermediate clock source for charge pump (LCD glass capacitance 4000pf or lower) 11 - Slowest clock source for charge pump (LCD glass capacitance 2000pf or lower)  For CPSEL = 0 Adjust the resistor bias network for different LCD glass capacitance 00 - Low Load (LCD glass capacitance 2000pf or lower) 01 - Low Load (LCD glass capacitance 2000pf or lower) 10 - High Load (LCD glass capacitance 8000pf or lower) 11 - High Load (LCD glass capacitance 8000pf or lower)
2 BBYPASS	<b>Op Amp Control</b> — Determines whether the internal LCD op amp buffer is bypassed. 0 Buffered mode 1 Unbuffered mode
1:0 VSUPPLY[1:0]	<b>Voltage Supply Control</b> — Configures whether the LCD module power supply is external or internal. Avoid modifying this bit field while the LCD module is enabled (e.g., LCDEN = 1). See Figure 11-16 for more detail. 00 Drive $V_{LL2}$ internally from $V_{DD}$ 01 Drive $V_{LL3}$ internally from $V_{DD}$ 10 Drive $V_{LL1}$ internally from $V_{LCD}$ 11 Drive $V_{LL3}$ externally Or $V_{IREG}$

### 11.3.4 LCD Regulated Voltage Control Register (LCDRVC)

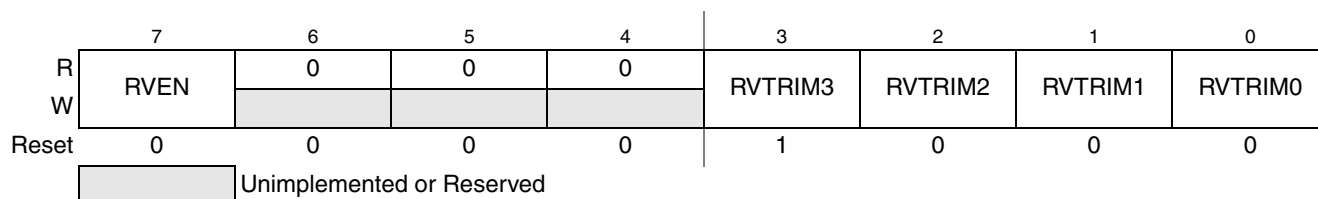


Figure 11-6. LCD Regulated Voltage Control Register (LCDRVC)

Read: anytime.

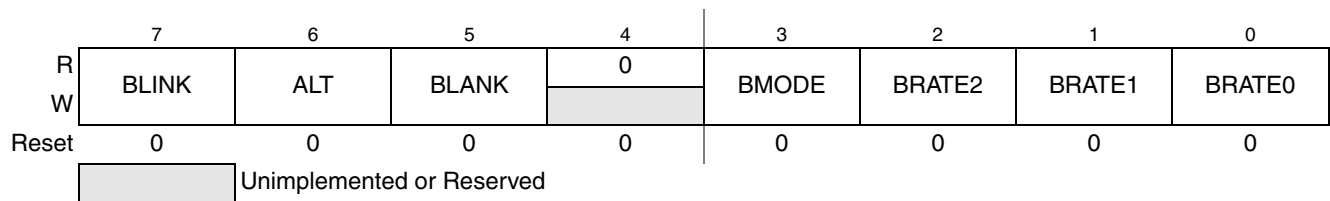
Write: anytime.{73}

The regulated voltage can be used to generate a reference signal to the LCD charge pump for 3V or 5V LCD operation dependant on the HREFSEL bit.

**Table 11-6. LCDRVC Field Descriptions**

Field	Description
7 RVEN	<b>Regulated Voltage Enable</b> — Enables internal voltage regulator, Must have charge pump enabled. 0 Regulated voltage disabled. 1 Regulated voltage enabled.
3:0 RVTRIM[3:0]	<b>Regulated Voltage Trim</b> —This 4 bit trim register is used to adjust the regulated input. Each bit in the register has equal weight. The Regulated input is changed by 1.5% for each count.

### 11.3.5 LCD Blink Control Register (LCDBCTL)



**Figure 11-7. LCD Blink Control Register (LCDBCTL)**

Read: anytime

Write: anytime

**Table 11-7. LCDBCTL Field Descriptions**

Field	Description
7 BLINK	<b>Blink Command</b> — Starts or stops LCD module blinking 0 Disables blinking 1 Starts blinking at blinking frequency specified by LCD blink rate calculation (see <a href="#">Equation 11-2</a> )
6 ALT	<b>Alternate Display Mode</b> — For four backplanes or less the LCD backplane sequencer changes to output an alternate display. ALT bit is ignored if Duty is 5 or greater. 0 Normal Display 1 Alternate display mode
5 BLANK	<b>Blank Display Mode</b> — Asserting this bit clears all segments in the LCD display. 0 Normal or Alternate Display 1 Blank Display Mode

Table 11-7. LCDBCTL Field Descriptions (continued)

Field	Description
3 BMODE	<b>Blink Mode</b> — Selects the blink mode displayed during the blink period. See Table 11-7 for more information on how BMODE affects the LCD display. 0 Display blank during the blink period 1 Display alternate display during blink period (Ignored if duty is 5 or greater)
2:0 BRATE[2:0]	<b>Blink-Rate Configuration</b> — Selects frequency at which the LCD display blinks when the BLINK is asserted. Equation 11-2 shows how BRATE[2:0] bit field is used in the LCD blink-rate calculation.  Equation 11-2 provides an expression for the LCD module blink rate  <div style="text-align: right;"><b>Eqn. 11-2</b></div> $\text{LCD module blink rate} = \frac{\text{LCDCLK}}{2^{(12 + \text{BRATE}[2:0])}}$ LCD module blink rate calculations are provided in 11.4.3.2/p.272.

### 11.3.6 LCD Status Register (LCDS)

	7	6	5	4	3	2	1	0
R	LCDIF	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0
		Unimplemented or Reserved						

Figure 11-8. LCD Status Register (LCDS)

Read: anytime

Write: anytime

Table 11-8. LCDS Field Descriptions

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

### 11.3.7 LCD Pin Enable Registers 0–7 (LCDPEN0–LCDPEN7)

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

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

		7	6	5	4	3	2	1	0
LCDPEN0	R	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
	W								
	Reset	Indeterminate after reset							
LCDPEN1	R	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
	W								
	Reset	Indeterminate after reset							
LCDPEN2	R	PEN23	PEN22	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
	W								
	Reset	Indeterminate after reset							
LCDPEN3	R	PEN31	PEN30	PEN29	PEN28	PEN27	PEN26	PEN25	PEN24
	W								
	Reset	Indeterminate after reset							
LCDPEN4	R	PEN39	PEN38	PEN37	PEN36	PEN35	PEN34	PEN33	PEN32
	W								
	Reset	Indeterminate after reset							
LCDPEN5	R	PEN47	PEN46	PEN45	PEN44	PEN43	PEN42	PEN41	PEN40
	W								
	Reset	Indeterminate after reset							
LCDPEN6	R	PEN55	PEN54	PEN53	PEN52	PEN51	PEN50	PEN49	PEN48
	W								
	Reset	Indeterminate after reset							
LCDPEN7	R	PEN63	PEN62	PEN61	PEN60	PEN59	PEN58	PEN57	PEN56
	W								
	Reset	Indeterminate after reset							


 Unimplemented or Reserved

Figure 11-9. LCD Pin Enable Registers 0–7 (LCDPEN0–LCDPEN7)

Read: anytime

Write: anytime

Table 11-9. LCDPEN0–LCDPEN7 Field Descriptions

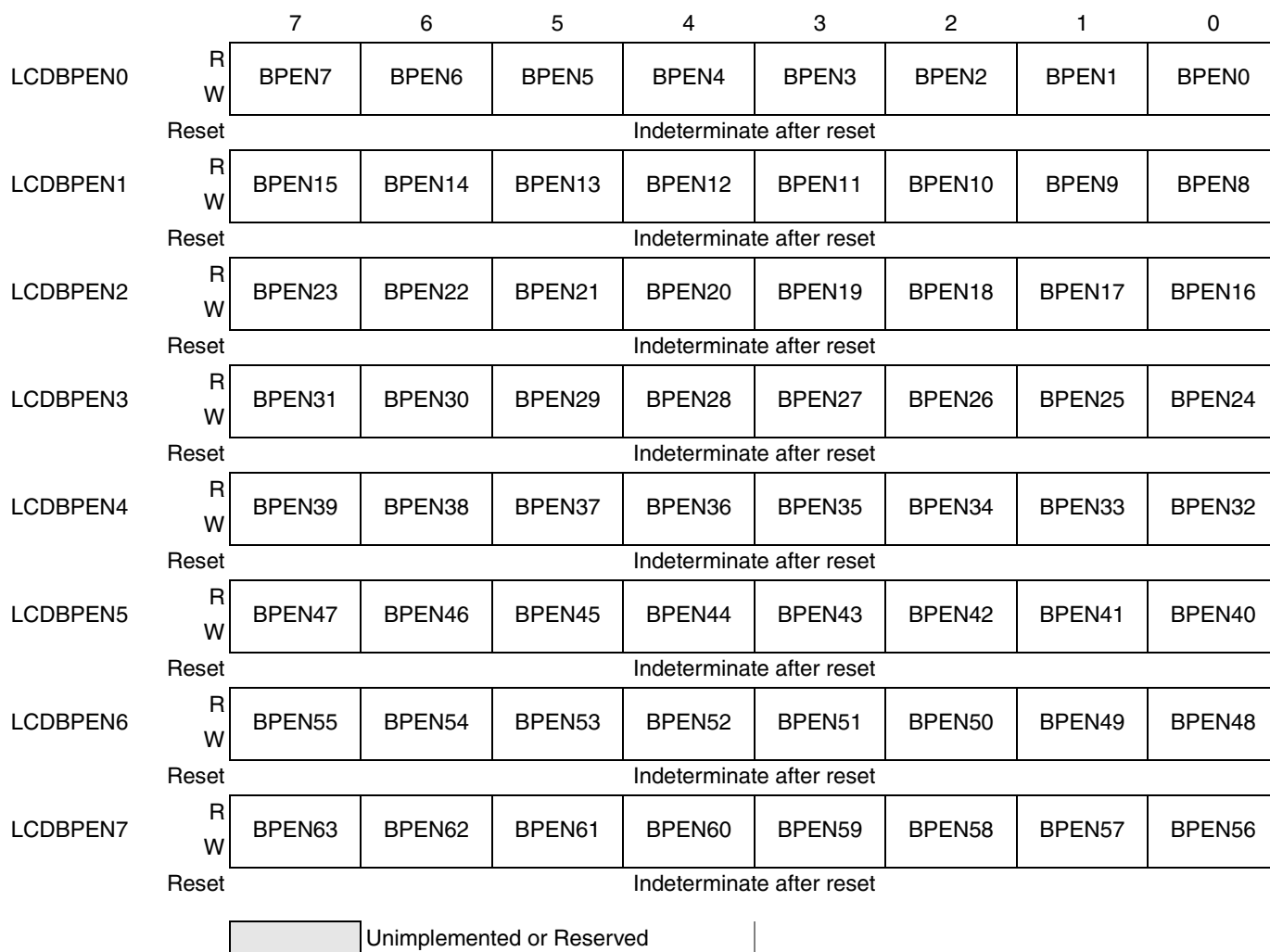
Field	Description
PEN[63:0]	<b>LCD Pin Enable</b> — The PEN[63:0] bit enables the LCD[63:0] pin for LCD operation. Each LCD[63:0] pin can be configured as a backplane or a frontplane based on the corresponding BPEN[n] bit in the Backplane Enable Register (LCDBPEN[7:0]). If LCDEN = 0, these bits have no effect on the state of the I/O pins. Set PEN[63:0] bits before LCDEN is set. 0 LCD operation disabled on LCDnn. 1 LCD operation enabled on LCDnn.

### 11.3.8 Backplane Enable Registers 0–7 (BPEN0–BPEN7)

When LCDPEN[n] = 1, these bits configure the corresponding LCD pin to operate as an LCD backplane or an LCD frontplane. Most applications set a maximum of eight of these bits. Initialize these registers

before enabling the LCD module. Exiting Stop2 mode does not require reinitializing the LCDBPEN registers.

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



**Figure 11-10. Backplane Enable Registers 0–7 (LCDBPEN0–LCDBPEN7)**

Read: anytime

Write: anytime

**Table 11-10. LCDBPEN0–LCDBPEN7 Field Descriptions**

Field	Description
BPEN[63:0]	<p><b>Backplane Enable</b> — The BPEN[63:0] bit configures the LCD[63:0] pin to operate as an LCD backplane or LCD frontplane. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set BPEN[63:0] bits before LCDEN is set.</p> <p>0 Frontplane operation enabled on LCD[n].</p> <p>1 Backplane operation enabled on LCD[n].</p>



### 11.3.9 LCD Waveform Registers (LCDWF[63:0])

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

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

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

After reset, the LCDWF contents are indeterminate as indicated by [Figure 11-11](#). Exiting Stop2 mode does not require reinitializing the LCDWF registers.

		7	6	5	4	3	2	1	0
<b>LCDWF0</b>	R								
	W	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
	Reset	Indeterminate after reset							
<b>LCDWF1</b>	R								
	W	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
	Reset	Indeterminate after reset							
<b>LCDWF2</b>	R								
	W	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
	Reset	Indeterminate after reset							
<b>LCDWF3</b>	R								
	W	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
	Reset	Indeterminate after reset							
<b>LCDWF4</b>	R								
	W	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
	Reset	Indeterminate after reset							
<b>LCDWF5</b>	R								
	W	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
	Reset	Indeterminate after reset							
<b>LCDWF6</b>	R								
	W	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
	Reset	Indeterminate after reset							
<b>LCDWF7</b>	R								
	W	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
	Reset	Indeterminate after reset							
<b>LCDWF8</b>	R								
	W	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
	Reset	Indeterminate after reset							

**Figure 11-11. LCD Waveform Registers (LCDWF[63:0])**

<b>LCDWF9</b>	R	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
	W								
	Reset	Indeterminate after reset							
<b>LCDWF10</b>	R	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
	W								
	Reset	Indeterminate after reset							
<b>LCDWF11</b>	R	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
	W								
	Reset	Indeterminate after reset							
<b>LCDWF12</b>	R	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
	W								
	Reset	Indeterminate after reset							
<b>LCDWF13</b>	R	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
	W								
	Reset	Indeterminate after reset							
<b>LCDWF14</b>	R	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
	W								
	Reset	Indeterminate after reset							
<b>LCDWF15</b>	R	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
	W								
	Reset	Indeterminate after reset							
<b>LCDWF16</b>	R	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
	W								
	Reset	Indeterminate after reset							
<b>LCDWF17</b>	R	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
	W								
	Reset	Indeterminate after reset							
<b>LCDWF18</b>	R	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
	W								
	Reset	Indeterminate after reset							
<b>LCDWF19</b>	R	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
	W								
	Reset	Indeterminate after reset							
<b>LCDWF20</b>	R	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
	W								
	Reset	Indeterminate after reset							
<b>LCDWF21</b>	R	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21
	W								
	Reset	Indeterminate after reset							
<b>LCDWF22</b>	R	BPHLCD22	BPGLCD22	BPFLCD22	BPELCD22	BPDLCD22	BPCLCD22	BPBLCD22	BPALCD22
	W								

Figure 11-11. LCD Waveform Registers (LCDWF[63:0]) (continued)

<b>LCDWF23</b>	Reset	Indeterminate after reset						
	R W	BPHLCD23	BPGLCD23	BPFLCD23	BPELCD23	BPDLCD23	BPCLCD23	BPBLCD23
<b>LCDWF24</b>	Reset	Indeterminate after reset						
	R W	BPHLCD24	BPGLCD24	BPFLCD24	BPELCD24	BPDLCD24	BPCLCD24	BPBLCD24
<b>LCDWF25</b>	Reset	Indeterminate after reset						
	R W	BPHLCD25	BPGLCD25	BPFLCD25	BPELCD25	BPDLCD25	BPCLCD25	BPBLCD25
<b>LCDWF26</b>	Reset	Indeterminate after reset						
	R W	BPHLCD26	BPGLCD26	BPFLCD26	BPELCD26	BPDLCD26	BPCLCD26	BPBLCD26
<b>LCDWF27</b>	Reset	Indeterminate after reset						
	R W	BPHLCD27	BPGLCD27	BPFLCD27	BPELCD27	BPDLCD27	BPCLCD27	BPBLCD27
<b>LCDWF28</b>	Reset	Indeterminate after reset						
	R W	BPHLCD28	BPGLCD28	BPFLCD28	BPELCD28	BPDLCD28	BPCLCD28	BPBLCD28
<b>LCDWF29</b>	Reset	Indeterminate after reset						
	R W	BPHLCD29	BPGLCD29	BPFLCD29	BPELCD29	BPDLCD29	BPCLCD29	BPBLCD29
<b>LCDWF30</b>	Reset	Indeterminate after reset						
	R W	BPHLCD30	BPGLCD30	BPFLCD30	BPELCD30	BPDLCD30	BPCLCD30	BPBLCD30
<b>LCDWF31</b>	Reset	Indeterminate after reset						
	R W	BPHLCD31	BPGLCD31	BPFLCD31	BPELCD31	BPDLCD31	BPCLCD31	BPBLCD31
<b>LCDWF32</b>	Reset	Indeterminate after reset						
	R W	BPHLCD32	BPGLCD32	BPFLCD32	BPELCD32	BPDLCD32	BPCLCD32	BPBLCD32
<b>LCDWF33</b>	Reset	Indeterminate after reset						
	R W	BPHLCD33	BPGLCD33	BPFLCD33	BPELCD33	BPDLCD33	BPCLCD33	BPBLCD33
<b>LCDWF34</b>	Reset	Indeterminate after reset						
	R W	BPHLCD34	BPGLCD34	BPFLCD34	BPELCD34	BPDLCD34	BPCLCD34	BPBLCD34
<b>LCDWF35</b>	Reset	Indeterminate after reset						
	R W	BPHLCD35	BPGLCD35	BPFLCD35	BPELCD35	BPDLCD35	BPCLCD35	BPBLCD35
	Reset	Indeterminate after reset						

Figure 11-11. LCD Waveform Registers (LCDWF[63:0]) (continued)

<b>LCDWF36</b>	R	BPHLCD36	BPGLCD36	BPFLCD36	BPELCD36	BPDLCD36	BPCLCD36	BPBLCD36	BPALCD36
	W								
	Reset	Indeterminate after reset							
<b>LCDWF37</b>	R	BPHLCD37	BPGLCD37	BPFLCD37	BPELCD37	BPDLCD37	BPCLCD37	BPBLCD37	BPALCD37
	W								
	Reset	Indeterminate after reset							
<b>LCDWF38</b>	R	BPHLCD38	BPGLCD38	BPFLCD38	BPELCD38	BPDLCD38	BPCLCD38	BPBLCD38	BPALCD38
	W								
	Reset	Indeterminate after reset							
<b>LCDWF39</b>	R	BPHLCD39	BPGLCD39	BPFLCD39	BPELCD39	BPDLCD39	BPCLCD39	BPBLCD39	BPALCD39
	W								
	Reset	Indeterminate after reset							
<b>LCDWF40</b>	R	BPHLCD40	BPGLCD40	BPFLCD40	BPELCD40	BPDLCD40	BPCLCD40	BPBLCD40	BPALCD40
	W								
	Reset	Indeterminate after reset							
<b>LCDWF41</b>	R	BPHLCD41	BPGLCD41	BPFLCD41	BPELCD41	BPDLCD41	BPCLCD41	BPBLCD41	BPALCD41
	W								
	Reset	Indeterminate after reset							
<b>LCDWF42</b>	R	BPHLCD42	BPGLCD42	BPFLCD42	BPELCD42	BPDLCD42	BPCLCD42	BPBLCD42	BPALCD42
	W								
	Reset	Indeterminate after reset							
<b>LCDWF43</b>	R	BPHLCD43	BPGLCD43	BPFLCD43	BPELCD43	BPDLCD43	BPCLCD43	BPBLCD43	BPALCD43
	W								
	Reset	Indeterminate after reset							
<b>LCDWF44</b>	R	BPHLCD44	BPGLCD44	BPFLCD44	BPELCD44	BPDLCD44	BPCLCD44	BPBLCD44	BPALCD44
	W								
	Reset	Indeterminate after reset							
<b>LCDWF45</b>	R	BPHLCD45	BPGLCD45	BPFLCD45	BPELCD45	BPDLCD45	BPCLCD45	BPBLCD45	BPALCD45
	W								
	Reset	Indeterminate after reset							
<b>LCDWF46</b>	R	BPHLCD46	BPGLCD46	BPFLCD46	BPELCD46	BPDLCD46	BPCLCD46	BPBLCD46	BPALCD46
	W								
	Reset	Indeterminate after reset							
<b>LCDWF47</b>	R	BPHLCD47	BPGLCD47	BPFLCD47	BPELCD47	BPDLCD47	BPCLCD47	BPBLCD47	BPALCD47
	W								
	Reset	Indeterminate after reset							
<b>LCDWF48</b>	R	BPHLCD48	BPGLCD48	BPFLCD48	BPELCD48	BPDLCD48	BPCLCD48	BPBLCD48	BPALCD48
	W								
	Reset	Indeterminate after reset							
<b>LCDWF49</b>	R	BPHLCD49	BPGLCD49	BPFLCD49	BPELCD49	BPDLCD49	BPCLCD49	BPBLCD49	BPALCD49
	W								

Figure 11-11. LCD Waveform Registers (LCDWF[63:0]) (continued)

<b>LCDWF50</b>	Reset	Indeterminate after reset						
	R W	BPHLCD50	BPGLCD50	BPFLCD50	BPELCD50	BPDLCD50	BPCLCD50	BPBLCD50
<b>LCDWF51</b>	Reset	Indeterminate after reset						
	R W	BPHLCD51	BPGLCD51	BPFLCD51	BPELCD51	BPDLCD51	BPCLCD51	BPBLCD51
<b>LCDWF52</b>	Reset	Indeterminate after reset						
	R W	BPHLCD52	BPGLCD52	BPFLCD52	BPELCD52	BPDLCD52	BPCLCD52	BPBLCD52
<b>LCDWF53</b>	Reset	Indeterminate after reset						
	R W	BPHLCD53	BPGLCD53	BPFLCD53	BPELCD53	BPDLCD53	BPCLCD53	BPBLCD53
<b>LCDWF54</b>	Reset	Indeterminate after reset						
	R W	BPHLCD54	BPGLCD54	BPFLCD54	BPELCD54	BPDLCD54	BPCLCD54	BPBLCD54
<b>LCDWF55</b>	Reset	Indeterminate after reset						
	R W	BPHLCD55	BPGLCD55	BPFLCD55	BPELCD55	BPDLCD55	BPCLCD55	BPBLCD55
<b>LCDWF56</b>	Reset	Indeterminate after reset						
	R W	BPHLCD56	BPGLCD56	BPFLCD56	BPELCD56	BPDLCD56	BPCLCD56	BPBLCD56
<b>LCDWF57</b>	Reset	Indeterminate after reset						
	R W	BPHLCD57	BPGLCD57	BPFLCD57	BPELCD57	BPDLCD57	BPCLCD57	BPBLCD57
<b>LCDWF58</b>	Reset	Indeterminate after reset						
	R W	BPHLCD58	BPGLCD58	BPFLCD58	BPELCD58	BPDLCD58	BPCLCD58	BPBLCD58
<b>LCDWF59</b>	Reset	Indeterminate after reset						
	R W	BPHLCD59	BPGLCD59	BPFLCD59	BPELCD59	BPDLCD59	BPCLCD59	BPBLCD59
<b>LCDWF60</b>	Reset	Indeterminate after reset						
	R W	BPHLCD60	BPGLCD60	BPFLCD60	BPELCD60	BPDLCD60	BPCLCD60	BPBLCD60
<b>LCDWF61</b>	Reset	Indeterminate after reset						
	R W	BPHLCD61	BPGLCD61	BPFLCD61	BPELCD61	BPDLCD61	BPCLCD61	BPBLCD61
	Reset	Indeterminate after reset						

Figure 11-11. LCD Waveform Registers (LCDWF[63:0]) (continued)

<b>LCDWF62</b>	R	BPHLCD62	BPGLCD62	BPFLCD62	BPELCD62	BPDLCD62	BPCLCD62	BPBLCD62	BPALCD62
	W								
Reset		Indeterminate after reset							
<b>LCDWF63</b>	R	BPHLCD63	BPGLCD63	BPFLCD63	BPELCD63	BPDLCD63	BPCLCD63	BPBLCD63	BPALCD63
	W								
Reset		Indeterminate after reset							

Figure 11-11. LCD Waveform Registers (LCDWF[63:0]) (continued)

Table 11-11. LCDWF Field Descriptions

Field	Description
BP[y]LCD[x]	<p><b>Segment-on-Frontplane Operation</b> — If the LCD[x] pin is enabled and configured to operate as a frontplane, the BP[y]LCD[x] bit in the LCDWF registers controls the on/off state for the LCD segment connected between LCD[x] and BP[y]. BP[y] corresponds to an LCD[63:0] pin enabled and configured to operate as a backplane that is active in phase [y]. Asserting BP[y]LCD[x] displays (turns on) the LCD segment connected between LCD[x] and BP[y].</p> <p>0 LCD segment off 1 LCD segment on</p> <p><b>Segment-on-Backplane Operation</b> — If the LCD[x] pin is enabled and configured to operate as a backplane, the BP[y] LCD[x] bit in the LCDWF registers controls the phase (A-H) in which the LCD[x] pin is active. Backplane phase assignment is done using this method.</p> <p>0 LCD backplane inactive for phase[y] 1 LCD backplane active for phase[y].</p>

## 11.4 Functional Description

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

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

- Frame frequency
- Duty cycle (number of backplanes)
- Backplane assignment (which LCD[63:0] pins operate as backplanes)
- Frame frequency interrupt enable
- Blinking frequency and options
- Power-supply configurations

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

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

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

For pins enabled as backplane, the phase of the backplane (A-H) is assigned by the LCDWF register for the corresponding backplane pin. For a detailed description of LCD module operation for a basic seven-segment LCD display, see [Section 11.6.1, “LCD Seven Segment Example Description.”](#)

## 11.4.1 LCD Driver Description

The LCD module driver has 8 modes of operation:

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

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

### 11.4.1.1 LCD Duty Cycle

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

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

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

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

The duty cycle is used by the backplane phase generator to set the phase outputs. The phase outputs A-H are driven according to the sequence shown below. The sequence is repeated at the LCD frame frequency. The duty cycle is configured using the DUTY[2:0] bit field in the LCDC0 register, as shown in [Table 11-12](#).

**Table 11-12. LCD Module Duty Cycle Modes**

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

#### 11.4.1.2 LCD Bias

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

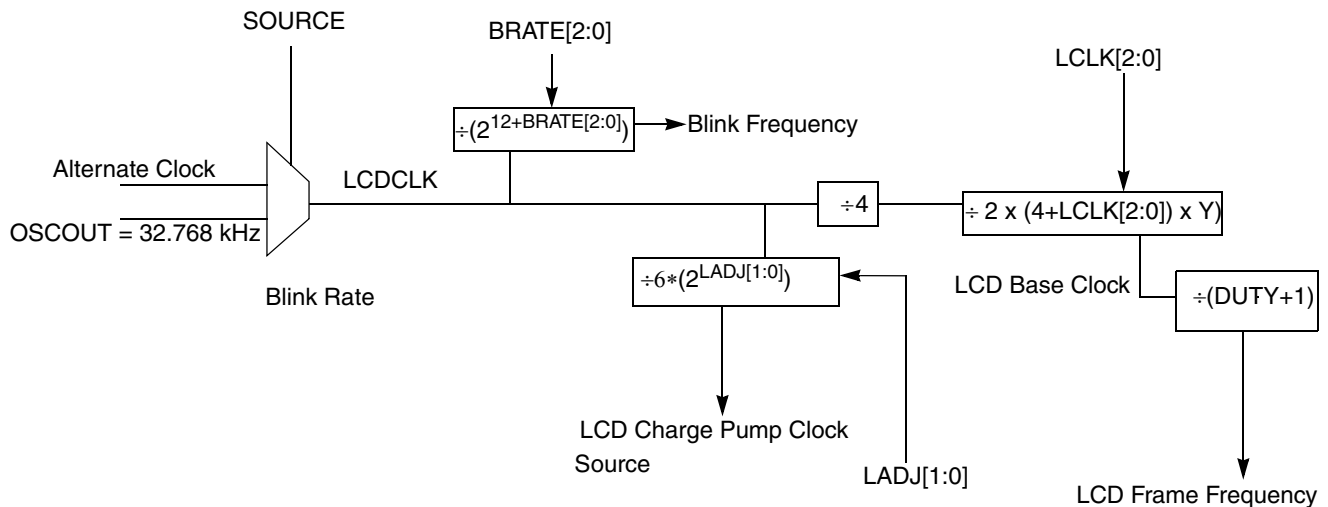
#### 11.4.1.3 LCD Module Base Clock and Frame Frequency

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

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

[Figure 11-12](#) shows the LCD clock tree. The clock tree shows the two possible clock sources and the LCD frame frequency and blink frequency clock source. The LCD blink frequency is discussed in [Section 11.4.3.2, “Blink Frequency.”](#)





**Figure 11-12. LCD Clock Tree**

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

The value of **LCDCLK** is important because it is used to generate the LCD module frame frequency. Equation 11-1 provides an expression for the LCD module frame frequency calculation.

The LCD module frame frequency is a function of the LCD module duty cycle as shown in Equation 11-1. Table 11-14 and Table 11-13 show LCD module frame frequency calculations that consider several possible LCD module configurations of **LCLK[2:0]** and **DUTY[2:0]**.

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

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

**Table 11-13. LCD Module Frame Frequency Calculations<sup>1</sup>**

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

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

Shaded table entries are out of specification and invalid.

**Table 11-14. LCD Module Frame Frequency Calculations<sup>1</sup>**

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

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

Shaded table entries are out of specification and invalid.

### 11.4.1.4 LCD Waveform Examples

This section shows the timing examples of the LCD output waveforms for the several modes of operation. As shown in [Table 11-15](#), all examples use 1/3 bias mode.

**Table 11-15. Configurations for Example LCD Waveforms**

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

#### 11.4.1.4.1 1/2 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

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

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

BPEN0=1 and BPEN1=1 in the LCDBPEN0

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

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

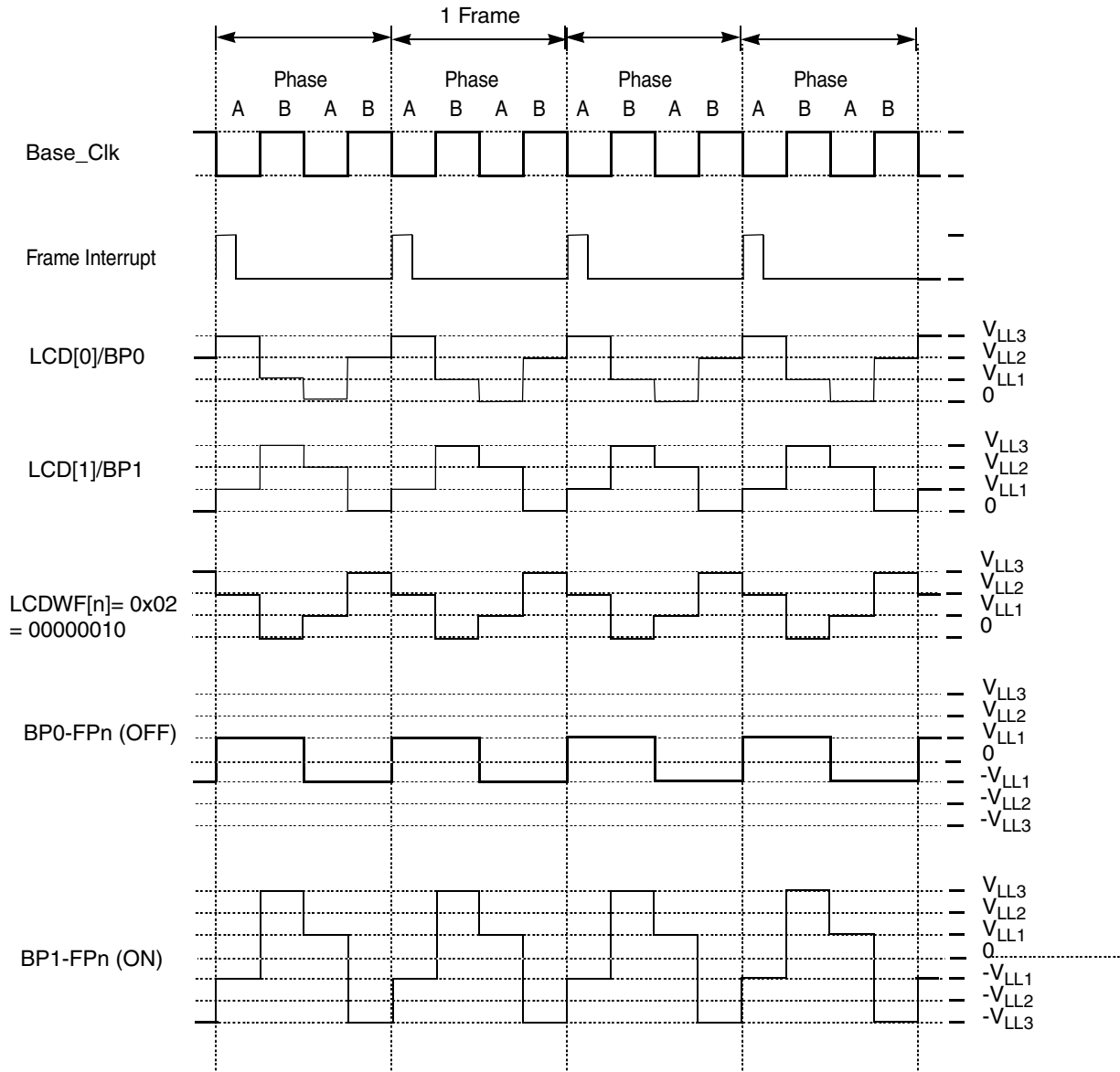


Figure 11-13. 1/2 Duty and 1/3 Bias (Low-Power Waveform)

### 11.4.1.4.2 1/4 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

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

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

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

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

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

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

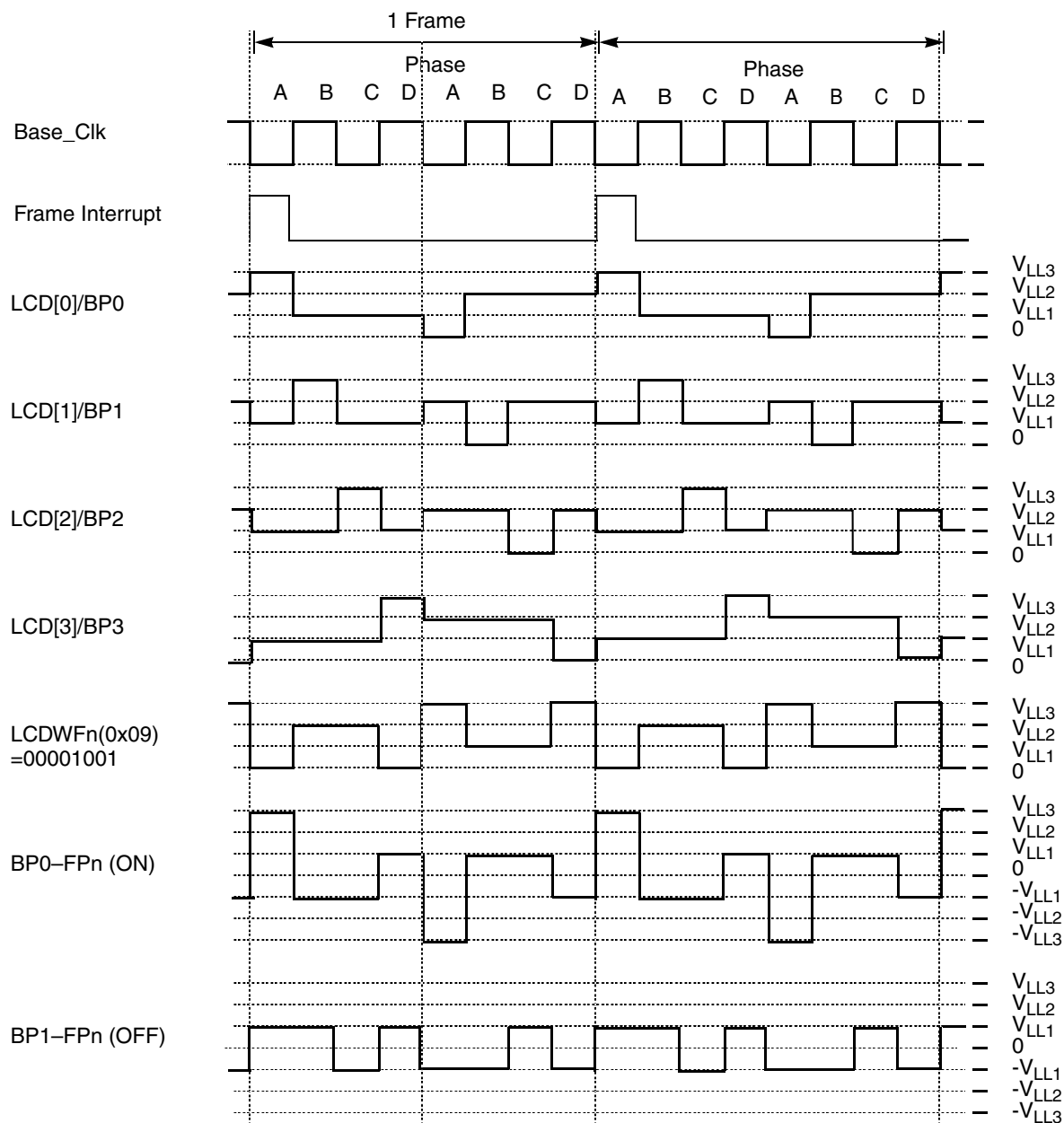


Figure 11-14. 1/4 Duty and 1/3 Bias (Low-Power Waveform)

#### 11.4.1.4.3 1/8 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

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

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

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

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

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

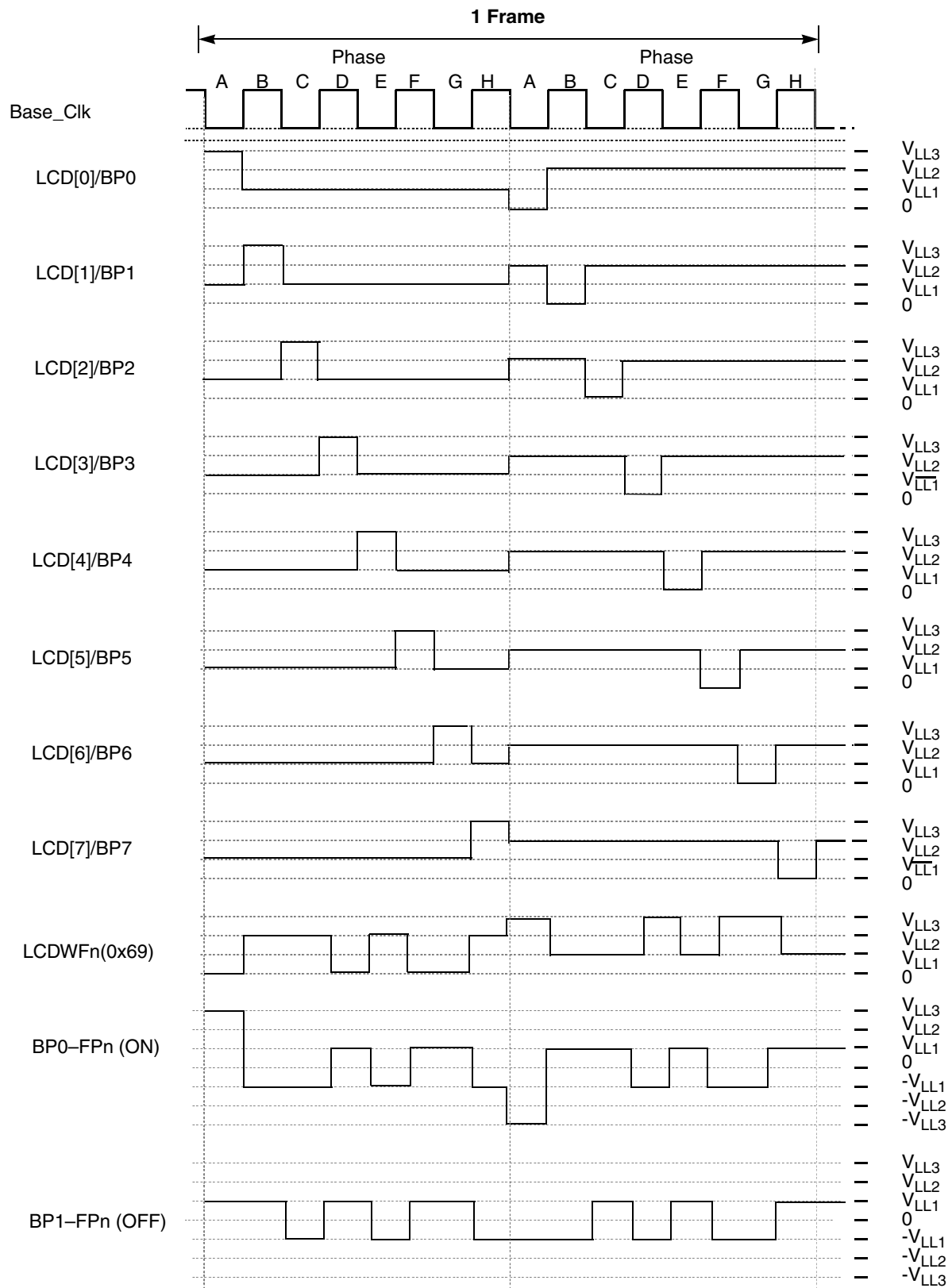
LCD[3] assigned to Phase D: LCDWF3 = 0x08

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

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

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

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



## 11.4.2 LCDWF Registers

For a segment on the LCD panel to be displayed, data must be written to the LCDWF registers. For LCD pins enabled as frontplanes, each bit in the LCDWF registers corresponds to a segment on an LCD panel. The different phases A-H represent the different backplanes of the LCD panel. The selected LCD duty cycle controls the number of implemented phases. Refer to [Table 11-12](#) for normal LCD operation the phases follow the sequence shown.

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

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

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

## 11.4.3 LCD Display Modes

The LCD module can be configured to implement several different display modes. The bits ALT and BLANK in the LCD-blink-control register (LCDBCTL) configure the different display modes. In normal display mode (default), LCD segments are controlled by the data placed in the LCDWF registers, as described in [Section 11.4.2, “LCDWF Registers.”](#) For blank-display mode, the LCDWF data is bypassed and the frontplane and backplane pins are configured to clear all segments.

For alternate-display mode, the backplane sequence is modified for duty cycles of 1/4, 1/3, 1/2, and 1/1. For four backplanes or less, the backplane sequence is modified as shown below. The altered sequence allows two complete displays to be placed in the LCDWF registers. The first display is placed in phases A-D and the second in phases E-H in the case of four backplanes. If the LCD duty cycle is five backplanes or greater, the ALT bit is ignored and creates a blank display. Refer to [Table 11-17](#) for additional information.

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

**Table 11-16. Alternate Display Backplane Sequence**

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

**Table 11-16. Alternate Display Backplane Sequence (continued)**

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

### 11.4.3.1 LCD Blink Modes

The blink mode is used as a means of alternating among different LCD display modes at a defined frequency. The LCD module can be configured to implement two blink modes. The BMODE bit in the LCD-blink-control register (LCDBCTL) configures the different blink modes. Blink modes are activated by setting the BLINK bit in the LCDBCTL register.

- If BLINK = 0, the LCD module operates normally as described [Section 11.4.3, “LCD Display Modes”](#).
- If BLINK = 1, BMODE bit configures the blinking operation. During a blink, the display data driven by the LCD module changes to the mode selected by the BMODE bit.

The BMODE bit selects two different blink modes, blank and alternate modes operate in the same way, as defined in [Section 11.4.3, “LCD Display Modes.”](#) The table below shows the interaction between display modes and blink modes. If the LCD duty cycle is five backplanes or greater, BMODE = 1 is ignored and will revert to create a blank display during the blink period.

**Table 11-17. Display Mode Interaction**

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

### 11.4.3.2 Blink Frequency

The LCD clock is the basis for the calculation of the LCD module blink frequency. The LCD module blink frequency is equal to the LCD clock (LCDCLK) divided by the factor selected by the BRATE[2:0] bits. [Table 11-18](#) shows LCD module blink frequency calculations for all values of BRATE[2:0] at a few common LCDCLK selections.



**Table 11-18. Blink Frequency Calculations**  
**(Blink Rate = LCD Clock(Hz) ÷ Blink Divider)**

BRATE[2:0]	0	1	2	3	4	5	6	7
LCD Clock	Blink Frequency (Hz)							
30 khz	7.32	3.66	1.831	.916	.46	.23	.11	.06
32.768 khz	8	4	2	1	.5	.25	.13	.06
39.063 khz	9.54	4.77	2.38	1.19	.6	.30	.15	.075

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

This section describes the LCD charge pump, voltage divider, and LCD power supply configuration options. [Figure 11-16](#) provides a block diagram for the LCD charge pump and the resistor divider network.

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

VSUPPLY[1:0] indicates the state of internal signals used to configure power switches as shown in the table in [Figure 11-16](#). The block diagram in [Figure 11-16](#) illustrates several potential operational modes for the LCD module including configuration of the LCD module power supply source using  $V_{DD}$ , internal regulated voltage  $V_{IREG}$  or an external supply on the  $V_{LL3}$  or  $V_{LCD}$  pins.  $V_{LL3}$  should never exceed  $V_{DD}$ .

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

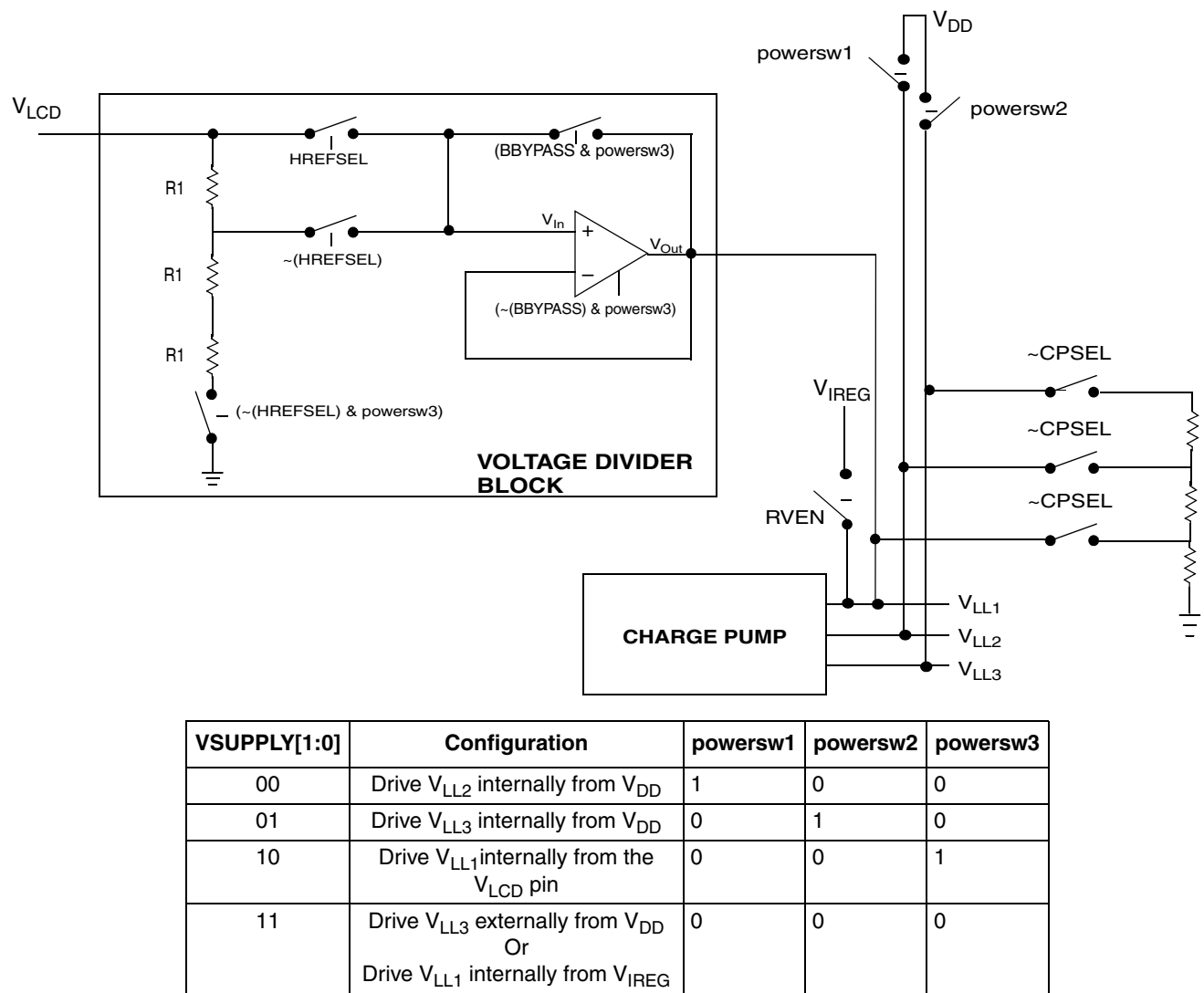


Figure 11-16. LCD Charge Pump and  $V_{LCD}$  Voltage Divider Block Diagram

Figure 11-16 also illustrates a buffer, a voltage follower with an ideal op amp. The buffer, if enabled, gives  $V_{In} = V_{Out}$ , and, because the input impedance of the op amp is very high,  $V_{In}$  is isolated from  $V_{Out}$ . This isolation can protect  $V_{In}$  from current draw from  $V_{Out}$ ; however, if the buffer is disabled ( $(\sim BBYPASS \& powersw3) = 0$ ), the output and input will be configured in a tri-state condition; that is, they will be floating.

#### NOTE:

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

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

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

#### 11.4.4.1 LCD Charge Pump and Voltage Divider

Using the voltage divider and charge pump, the LCD module can effectively double or triple the input voltage placed on the  $V_{LCD}$  pin. This LCD module configurability makes the LCD module compatible with both 3-V or 5-V LCD glass.

The LCD module high reference select bit (HREFSEL) in the LCDSUPPLY register configures the LCD module operational mode as a voltage doubler or a voltage tripler. In Figure 11-16, HREFSEL bit signal is used to control switches within the voltage divider block to enable or disable the two-thirds ( $2/3 * V_{LCD}$ ) voltage divider. If HREFSEL = 0, the LCD module is configured as a voltage doubler, by enabling the voltage divider. With this configuration if  $V_{LCD} = 1.5$  V the bias voltages generated will be:

$$V_{LCD} * 2/3 = 1.5 \text{ V} * 2/3 = 1 \text{ V} = V_{LL1}$$

$$V_{LL1} * 2 = 2 \text{ V} = V_{LL2}$$

$$V_{LL1} * 3 = 3 \text{ V} = V_{LL3}$$

If HREFSEL = 1, the LCD module is configured as a voltage tripler by disabling the voltage divider. ( $V_{LL1} = V_{LCD}$ ) The HREFSEL configuration depends on the LCD panel operating voltage specification in the application.

##### 11.4.4.1.1 CPSEL: LCD Charge Pump or Resistor Bias Enable

The CPSEL bit in the LCDSUPPLY register selects the charge pump. When the charge pump is selected (CPSEL = 1), an internal charge pump is used to generate the LCD bias voltages;  $V_{LL1}$ ,  $V_{LL2}$ , and  $V_{LL3}$ .

When the charge pump is unselected (CPSEL = 0),  $V_{LL3}$  must be supplied and  $V_{LL1}$ ,  $V_{LL2}$  are generated by a resistor bias network.

### 11.4.4.2 LCD Power Supply and Voltage Buffer Configuration

The LCD bias voltages can be

- internally derived from  $V_{DD}$ ,
- internally derived from a voltage source (must not exceed  $V_{DD}$ ) connected to  $V_{LL3}$ ,
- internally derived from a regulated voltage source that can be configured to supply 1.0 or 1.67 V ( $V_{IREG}$ ),
- internally derived from a voltage source in the range between .9 to 1.8 V that is applied to the  $V_{LCD}$  pin.

Table 11-20 provides a more detailed description of the power state of the LCD module which depends on the configuration of the VSUPPLY[1:0], HREFSEL, BBYPASS, CPSEL and RVEN bits.

Table 11-20 shows all possible configurations of the LCD Power Supply. All other combinations of the configuration bits above are not permissible LCD power supply modes and should be avoided.

**Table 11-20. LCD Power Supply Options**

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	HREFSEL	BBYPASS	CPSEL	RVEN
$V_{LL2}$ connected to $V_{DD}$ internally for 3 or 5 V glass operation.	For 3 V glass operation $V_{DD}$ must equal 2 V. For 5 V glass operation $V_{DD}$ must equal 3.33 V Charge pump is used to generate $V_{LL1}$ and $V_{LL3}$ $V_{LCD}$ pin is not connected	00	X	0	1	0
$V_{LL3}$ connected to $V_{DD}$ internally for 3 V or 5 V glass operation	For 3 V glass operation $V_{DD}$ must equal 3 V. For 5 V glass operation $V_{DD}$ must equal 5 V. Charge pump is used to generate $V_{LL1}$ and $V_{LL2}$ $V_{LCD}$ pin is not connected	01	X	0	1	0
$V_{LCD} * 3/3$ connected to $V_{LL1}$ for 3 V glass operation. Unbuffered mode.	For 3 V glass operation $V_{LCD}$ must equal 1 V. $V_{LCD}$ (1 V) is connected to $V_{LL1}$ HREFSEL = 1 to set $V_{LL1} = V_{LCD} * 3/3$ Charge pump is used to generate $V_{LL2}$ , and $V_{LL3}$	10	1	1	1	0

Table 11-20. LCD Power Supply Options (continued)

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	HREFSEL	BBYPASS	CPSEL	RVEN
$V_{LCD} * 2/3$ connected to $V_{LL1}$ for 3 V glass operation. Buffered mode.	For 3 V glass operation $V_{LCD}$ must equal 1.5 V.  $2/3 V_{LCD}$ (1 V) is connected to $V_{LL1}$  $HREFSEL = 0$ to set $V_{LL1} = V_{LCD} * 2/3$  Charge pump is used to generate $V_{LL2}$ , and $V_{LL3}$	10	0	0	1	0
$V_{LCD} * 3/3$ connected to $V_{LL1}$ for 3 V glass operation. Buffered mode.	For 3 V glass operation $V_{LCD}$ must equal 1 V.  $V_{LCD}$ (1 V) is connected to $V_{LL1}$  $HREFSEL = 1$ to set $V_{LL1} = V_{LCD} * 3/3$  Charge pump is used to generate $V_{LL2}$ , and $V_{LL3}$	10	1	0	1	0
$V_{LCD}$ connected to $V_{LL1}$ for 5 V glass operation. Unbuffered mode.	For 5 V glass operation $V_{LCD}$ must equal 1.67 V.  $V_{LCD}$ (1.67 V) is connected to $V_{LL1}$  $HREFSEL = 1$ to set $V_{LL1} = V_{LCD} * 3/3$  Charge pump is used to generate $V_{LL2}$ , and $V_{LL3}$	10	1	1	1	0
$V_{LCD}$ connected to $V_{LL1}$ for 5 V glass operation. Buffered mode.	For 5 V glass operation $V_{LCD}$ must equal 1.67 V.  $V_{LCD}$ (1.67 V) is connected to $V_{LL1}$  $HREFSEL = 1$ to set $V_{LL1} = V_{LCD} * 3/3$  Charge pump is used to generate $V_{LL2}$ , and $V_{LL3}$	10	1	0	1	0
$V_{LL3}$ is driven externally for 3 V LCD Glass operation.	For 3 V glass operation $V_{LL3}$ must equal 3 V.  $V_{LCD}$ is not connected  Charge pump is used to generate $V_{LL1}$ and $V_{LL2}$	11	X	0	1	0
$V_{LL3}$ is driven externally for 5 V LCD Glass operation.  $V_{LL3}$ must equal $V_{DD}$  This operation is not allowed for 1.8 V to 3.6 V parts	For 5 V glass operation $V_{LL3}$ must equal 5 V.  $V_{LCD}$ is not connected  Charge pump is used to generate $V_{LL1}$ and $V_{LL2}$	11	X	0	1	0

Table 11-20. LCD Power Supply Options (continued)

LCD Operational State	LCD Power Supply Configuration	VSUPPLY[1:0]	HREFSEL	BBYPASS	CPSEL	RVEN
$V_{LL3}$ is driven externally for 3 V LCD Glass operation.  Resistor Bias Network enabled.	For 3 V glass operation $V_{LL3}$ must equal 3 V.  $V_{LCD}$ is not connected.  Charge pump is disabled.  Resistor Bias network is used to create $V_{LL1}$ and $V_{LL2}$	11	X	0	0	0
$V_{LL3}$ is driven externally for 5 V LCD Glass operation. Resistor Bias network enabled.  $V_{LL3}$ must equal $V_{DD}$  This operation is not allowed for 1.8 V to 3.6 V parts	For 5 V glass operation $V_{LL3}$ must equal 5 V.  $V_{LCD}$ is not connected.  Charge pump is disabled.  Resistor Bias network is used to create $V_{LL1}$ and $V_{LL2}$ .	11	X	0	0	0
$V_{IREG}$ is connected to $V_{LL1}$ for 5 V glass operation.  The HREFSEL bit is used to select 1.0 or 1.67 V range for $V_{IREG}$	For 5 V glass operation HREFSEL = 1, $V_{IREG}$ = 1.67 V.  $V_{IREG}$ is connected $V_{LL1}$ internally.  $V_{LCD}$ is not connected.  Charge pump is used to generate $V_{LL2}$ and $V_{LL3}$ .	11	1	0	1	1
$V_{IREG}$ is connected to $V_{LL1}$ for 3 V glass operation.  The HREFSEL bit is used to select 1.0 or 1.67 V range for $V_{IREG}$	For 3 V glass operation HREFSEL = 0, $V_{IREG}$ = 1 V.  $V_{IREG}$ is connected $V_{LL1}$ internally.  $V_{LCD}$ is not connected.  Charge pump is used to generate $V_{LL2}$ and $V_{LL3}$ .	11	0	0	1	1

#### 11.4.4.2.1 VSUPPLY[1:0] = 10

If VSUPPLY[1:0] = 10, the LCD power supply is configured to be internally derived from the  $V_{LCD}$  pin.

#### External $V_{LCD}$ supply

When VSUPPLY[1:0] = 10, only the powersw3 signal is asserted, and the LCD module is configured to be powered via an input ( $V_{LCD}$  in the range from .9 V to 1.8 V). {statement} Figure 11-16 shows that  $V_{LCD}$  can be an input to the voltage divider block and is related to  $V_{LL1}$ . The voltage-divider block uses the states of HREFSEL, BBYPASS, and powersw3 to derive a state for  $V_{LL1}$ .

The output of the voltage divider block is be connected to  $V_{LL1}$ .  $V_{LL1}$  is connected to the internal charge pump. Using the charge pump, the value of  $V_{LL1}$  is tripled and output as  $V_{LL3}$ .  $V_{LL3}$ , an LCD bias voltage, is equal to the voltage required to energize the LCD panel,  $V_{LCDON}$ . For 3-V LCD glass,  $V_{LL3}$  should be approximately 3 V; while for 5-V LCD glass,  $V_{LL3}$  should be approximately 5 V.

Depending on the HREFSEL bit configuration,  $V_{LL3}$  will be equal to  $3 \times V_{LCD}$  or  $3 \times (2/3 \times V_{LCD})$  (see Section 11.4.4, “LCD Charge Pump, Voltage Divider, and Power Supply Operation”). Table 11-21 shows the selected  $V_{LL1}$  and  $V_{LL3}$  values based on the input value of  $V_{LCD}$ .

**Table 11-21.  $V_{LL1}$  Typical Values**

$V_{LCD}$	HREFSEL = 0 Voltage Doubler		HREFSEL = 1 Voltage Tripler	
	$V_{LL1} = V_{ref}$	$V_{LL3} = 3 \times V_{ref}$	$V_{LL1} = V_{ref}$	$V_{LL3} = 3 \times V_{ref}$
1.4 V	$(2/3) \times 1.4 \text{ V}$	2.8 V	1.4 V	4.2 V
1.5 V	$(2/3) \times 1.5 \text{ V}$	3.0 V	1.5 V	4.5 V
1.7 V	$(2/3) \times 1.7 \text{ V}$	3.4 V	1.7 V	5.1 V
1.8 V	$(2/3) \times 1.8 \text{ V}$	3.6 V	1.8 V	5.4 V

In addition to  $V_{LL1}$  and  $V_{LL3}$ ,  $V_{LL2}$  is also generated internally when the charge pump is enabled (CPSEL = 1). For a typical LCD panel, the bias voltages in 1/3 bias mode are:

- $V_3 = V_{LL3} = V_{LCDON} = 3 \times V_{ref}$
- $V_2 = V_{LL2} = 2 \times V_{ref}$
- $V_1 = V_{LL1} = V_{ref}$
- $V_0 = V_{SS}$

#### NOTE

$V_{LCDON}$  is the LCD panel driving voltage required to turn on an LCD segment. Since  $V_{LL3}$  and  $V_{LCDON}$  are equivalent,  $V_{LL3}$  should be configured so that it is 3 V or 5 V, depending on the LCD panel specification.

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

When VSUPPLY[1:0] = 11, powersw1, powersw2, and powersw3 are deasserted. With powersw3 deasserted, the buffer is disabled ( $(\sim \text{BBYPASS} \ \& \ \text{powersw3}) = 0$ ), so  $V_{LCD}$  will be configured in a tri-state condition.  $V_{DD}$  is not available to power the LCD module internally, so the LCD module requires an external power source for  $V_{LL1}$ ,  $V_{LL2}$ , and  $V_{LL3}$  when the charge pump is disabled.

If the charge pump is enabled, external power must be applied to  $V_{LL3}$ . With this configuration, the charge pump will generate the other LCD bias voltages  $V_{LL1}$  and  $V_{LL2}$ .

#### Internal $V_{IREG}$

If the charge pump is enabled the internal regulated voltage,  $V_{IREG}$ , can be used as an input to generate the LCD bias voltages. In this state external voltage source should not be connected to  $V_{LL1}$ .  $V_{IREG}$  is controlled by the LCD regulated voltage control (LCDRVC) register. The figure above, Figure 11-16, shows that  $V_{IREG}$  is connected to  $V_{LL1}$  when the RVEN bit is set. {statement}

$V_{LL1}$  is connected to the internal charge pump. Using the charge pump, the value of  $V_{LL1}$  is tripled and output as  $V_{LL3}$ .  $V_{LL3}$ , an LCD bias voltage, is equal to the voltage required to energize the LCD panel,

$V_{LCDON}$ . For 3-V LCD glass,  $V_{LL3}$  should be approximately 3 V; while for 5-V LCD glass,  $V_{LL3}$  should be approximately 5 V.

The HREFSEL bit in the LCDSUPPLY register is used to set  $V_{IREG}$  to approximately 1.0 V or 1.67 V as shown in Table 11-22. The LCDRVC contains trim bits that can be used to make changes to the regulated voltage. The trim can be used to increase or decrease the regulated voltage by 1.5% for each count. A total of  $\pm 12\%$  of change can be done to the regulated voltage.

Table 11-22 shows the selected LCD bias voltages  $V_{LL1}$ ,  $V_{LL2}$ , and  $V_{LL3}$  values based on the value of  $V_{IREG}$ .

**Table 11-22. Bias Voltage Typical Values**

HREFSEL	$V_{IREG}$	$V_{LL1} = V_{ref}$	$V_{LL2} = 2 \times V_{ref}$	$V_{LL3} = 3 \times V_{ref}$
HREFSEL = 0	1.0 V	1.0 V	2.0 V	3.0 V
HREFSEL = 1	1.67 V	1.67 V	3.33 V	5.0 V

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

$V_{DD}$  is used as the LCD module power supply when VSUPPLY[1:0] = 00 or 11 (Table 11-23). When powering the LCD module using  $V_{DD}$ , the charge pump must be enabled (CPSEL = 1). Table 11-23 provides recommendations regarding configuration of the VSUPPLY[1:0] bit field when using both 3-V and 5-V LCD panels.

**Table 11-23.  $V_{DD}$  Switch Option**

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

### 11.4.5 Resets

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

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



## 11.4.6 Interrupts

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

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

## 11.5 Initialization Section

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

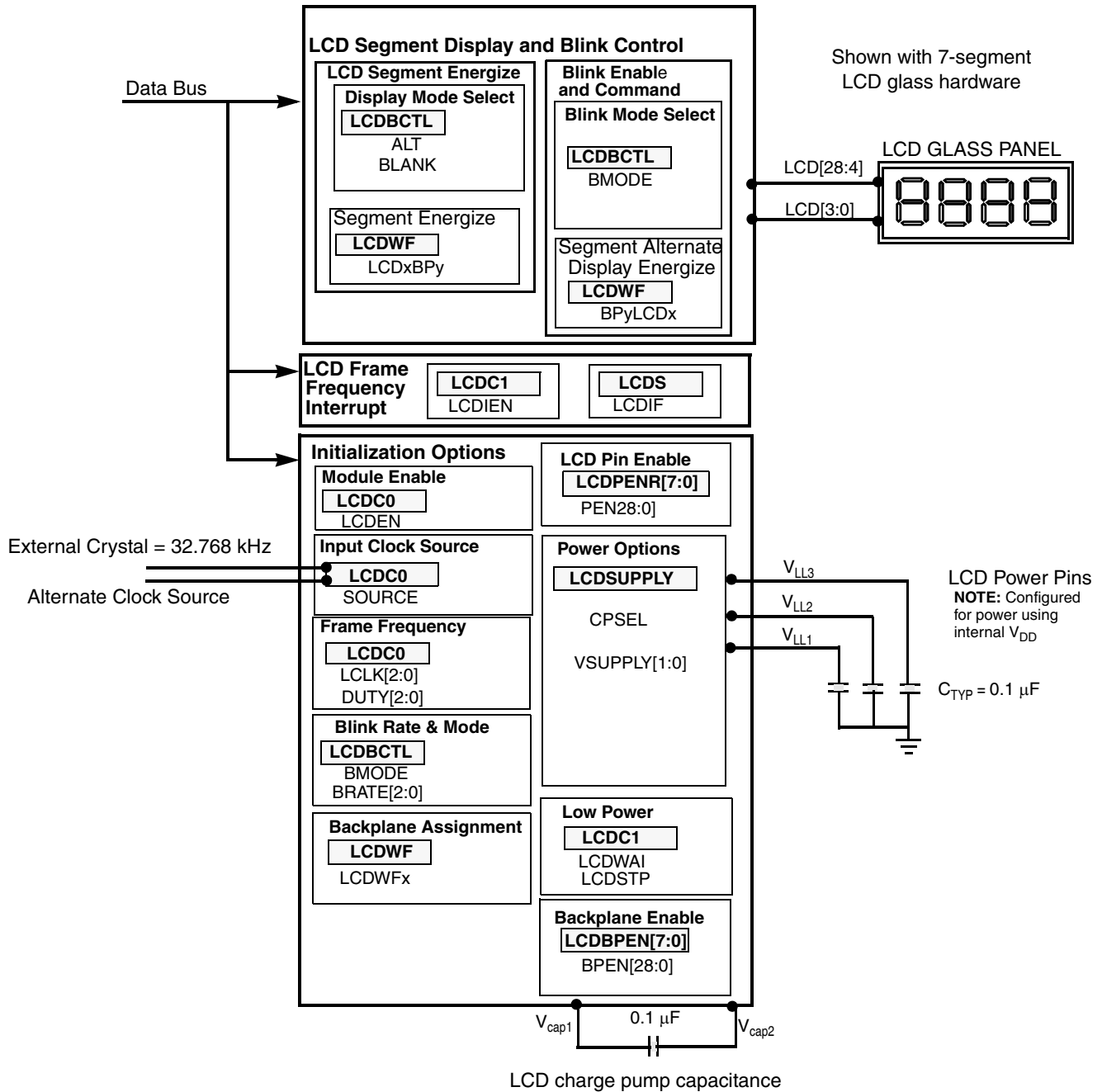
### 11.5.1 Initialization Sequence

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

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

1. LCDC0 register
  - a) Configure LCD clock source (SOURCE bit).
2. LCDRVC register (If the application uses the internally regulated voltage)
  - a) Select 1.0 V or 1.67 V for 3 or 5 V glass (HREFSEL).
  - b) Enable regulated voltage (RVEN).
  - c) Trim the regulated voltage (RVTRIM).
3. LCDSUPPLY register
  - a) Enable charge pump (CPSEL bit).
  - b) Configure LCD module for doubler or tripler mode (HREFSEL bit).
  - c) Configure charge pump clock (LADJ[1:0]).
  - d) Configure op amp switch (BBYPASS bit).
  - e) Configure LCD power supply (VSUPPLY[1:0]).
4. LCDC1 register
  - a) Configure LCD frame frequency interrupt (LCDIEN bit).
  - b) Configure LCD behavior in low power mode (LCDWAI and LCDSTP bits).
5. LCDC0 register
  - a) Configure LCD duty cycle (DUTY[2:0]).
  - b) Select and configure LCD frame frequency (LCLK[2:0]).
6. LCDBCTL register
  - a) Configure display mode (ALT and BLANK bits).
  - b) Configure blink mode (BMODE).
  - c) Configure blink frequency (BRATE[2:0]).

7. LCDPEN[7:0] register
  - a) Enable LCD module pins (PEN[63:0] bits).
8. LCDBPEN[7:0]
  - a) Enable LCD pins to operate as an LCD backplane (BPEN[63:0]).
9. LCDC0 register
  - a) Enable LCD module (LCDEN bit).



## 11.5.2 Initialization Examples

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

**Table 11-24. LCD Application Scenario**

Example	Operating Voltage, $V_{DD}$	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in WAIT/STOP modes	LCD Power Input
1	1.8 V	External 32.768 kHz	3 V	128	30 Hz	None	WAIT: on STOP: on	Power via $V_{LCD}$
2	3.6 V	Internal 39.063 kHz	3 V	100	50 Hz	Alternate 0.5 Hz	WAIT: on STOP: off	Power via $V_{DD}$
3	3.0 V	External 32.768 kHz	5 V	168	30 Hz	Blank 2.0 Hz	WAIT: off STOP: off	Power via $V_{IREG}$

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

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

### 11.5.2.1 Initialization Example 1

**Table 11-25. LCD Setup Requirements for Example 1**

Example	Operating Voltage, $V_{DD}$	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in STOP and WAIT modes	LCD Power Input
1	1.8-V	External 32.768 kHz	3-V	128	30 Hz	None	WAIT: on STOP: on	Power via $V_{LCD}$

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

**Table 11-26. Initialization Register Values for Example 1**

Register	bit or bit field	Binary Value	Comment
LCDSUPPLY 1000-010	CPSEL	1	Enable charge pump
	HREFSEL	0	For 3-V LCD glass, select doubler mode; Doubler mode = 0; Tripler mode = 1
	LADJ[1:0]	00	Configure LCD charge pump clock source
	BBYPASS	0	Buffer Bypass; Buffer mode = 0; Unbuffered mode = 1
	VSUPPLY[1:0]	10	When VSUPPLY[1:0] = 10, the LCD must be externally powered via $V_{LCD}$ (see <a href="#">Table 11-20</a> ). For 3-V glass, the nominal value of $V_{LCD}$ should be 1.5-V.
LCDC1 0-----00	LCDIEN	0	LCD frame interrupts disabled
	LCDWAI	0	LCD is "on" in WAIT mode
	LCDSTP	0	LCD is "on" in STOP mode
LCDC0 00101111	LCDEN	0	Initialization is done before initializing the LCD module
	SOURCE	0	Selects the external clock reference as the LCD clock input (OSCOU)
	LCLK[2:0]	101	For 1/8 duty cycle, select closest value to the desired 30 Hz LCD frame frequency (see <a href="#">Table 11-13</a> )
	DUTY[2:0]	111	For 128 segments (8x16), select 1/8 duty cycle
LCDBCTL 0XX-XXXX	BLINK	0	No blinking
	ALT	X	Alternate bit is configured during LCD operation
	BLANK	X	Blank bit is configured during LCD operation
	BMODE	X	N/A; Blink Blank = 0; Blink Alternate = 1
	BRATE[2:0]	XXX	N/A
LCDPEN[7:0]	LCDPEN0 LCDPEN1 LCDPEN2 LCDPEN3	11111111 11111111 11111111 00000000	Only 24 LCD pins need to be enabled.  <b>Note:</b> Any of the 63 LCD pins can be used, this allows flexibility in the hardware design.
LCDBPEN[7:0]	LCDBPEN0 LCDBPEN1 LCDBPEN2 LCDBPEN3	11111111 00000000 00000000 00000000	Eight backplane pins needed.  <b>Note:</b> Any enabled LCD pin can be enabled to operate as a backplane.
LCDWF[63:0]	LCDWF0 LCDWF1 LCDWF2 LCDWF3 LCDWF4 LCDWF5 LCDWF6 LCDWF7	00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000	Configure which phase the eight backplane pins will be active in. This configuration sets LCD[0] to be active in Phase A, LCD[1] to be active in Phase B...etc  <b>Note:</b> Any backplane pin can be active in any phase.

## 11.5.2.2 Initialization Example 2

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

Example	Operating Voltage, $V_{DD}$	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in STOP3 and WAIT modes	LCD Power Input
2	3.6 V	Internal 39.063 kHz	3 V	100	50 Hz	Alternate 0.5 Hz	WAIT: on STOP: off	Power via $V_{DD}$

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

**Table 11-27. Initialization Register Values for Example 2**

Register	bit or bit field	Binary Value	Comment
LCDSUPPLY 1X00-001	CPSEL	1	Enable charge pump
	HREFSEL	X	Don't Care since power is from internal $V_{DD}$ ; Doubler mode = 0; Tripler mode = 1
	LADJ[1:0]	00	Configure LCD charge pump clock source
	BBYPASS	0	Buffer Bypass; Buffer mode = 0; Unbuffered mode = 1
	VSUPPLY[1:0]	01	Generate $V_{LL3}$ from $V_{DD}$ (See <a href="#">Table 11-20</a> )
LCDC1 0-----01	LCDIEN	0	LCD Frame Interrupts disabled
	LCDWAI	0	LCD is "on" in WAIT mode
	LCDSTP	1	LCD is "off" in STOP mode
LCDC0 01010011	LCDEN	0	Initialization is done before initializing the LCD module
	SOURCE	1	Selects the alternate-clock reference as the LCD clock input (ALTCLK) This clock source is configured by the ICS TRIM bits to be 39.063Khz.
	LCLK[2:0]	010	For 1/4 duty cycle, select closest value to the desired 50 Hz LCD frame frequency ( <a href="#">Table 11-14</a> )
	DUTY[2:0]	011	For 100 segments (4x25), select 1/4 duty cycle
LCDBCTL 1XX-1100	BLINK	1	Blinking is turned on or off during LCD operation
	ALT	X	Alternate bit is configured during LCD operation
	BLANK	X	Blank bit is configured during LCD operation
	BMODE	1	Blink Alternate = 1
	BRATE[2:0]	100	Select 5 Hz blink frequency using <a href="#">Table 11-18</a>
LCDPEN[7:0]	LCDPEN0 LCDPEN1 LCDPEN2 LCDPEN3	11111111 11111111 11111111 00011111	29 LCD pins need to be enabled.

Table 11-27. Initialization Register Values for Example 2 (continued)

Register	bit or bit field	Binary Value	Comment
LCDBPEN[7:0]	LCDBPEN0	00001111	Four backplane pins needed.  <b>Note:</b> Any enabled LCD pin can be enabled to operate as a backplane.
	LCDBPEN1	00000000	
	LCDBPEN2	00000000	
	LCDBPEN3	00000000	
LCDWF[63:0]	LCDWF0	00000001	Configure which phase the four backplane pins will be active in. This configuration sets LCD[0] to be active in Phase A, LCD[1] to be active in Phase B etc.  <b>Note:</b> Any backplane pin can be active in any of the phases.
	LCDWF1	00000010	
	LCDWF2	00000100	
	LCDWF3	00001000	

### 11.5.2.3 Initialization Example 3

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

Example	Operating Voltage, $V_{DD}$	LCD Clock Source	LCD Glass Operating Voltage	Required LCD segments	LCD Frame Rate	Blinking Mode/Rate	Behavior in STOP3 and WAIT modes	LCD Power Input
3	3.0 V	External 32.768 kHz	5 V	168	30 Hz	Blank 2.0 Hz	WAIT: off STOP: off	Power via $V_{IREG}$

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

Table 11-28. Initialization Register Values for Example 3

Register	bit or bit field	Binary Value	Comment
LCDRVC 1---XXXX	RVEN	1	Enable $V_{IREG}$ so that it can be used to supply the LCD
	RVTRIM	XXXX	Trim value is determined by characterization
LCDSUPPLY 1100-000	CPSEL	1	Enable charge pump
	HREFSEL	1	For 5V glass must enable 1.67V
	LADJ[1:0]	00	Configure LCD charge pump clock source
	BBYPASS	0	Buffer Bypass; Buffer mode = 0; Unbuffered mode = 1
	VSUPPLY[1:0]	11	When VSUPPLY[1:0] = 11, the LCD can be powered via $V_{IREG}$ (see <a href="#">Table 11-20</a> ). For 5-V glass, the nominal value of $V_{IREG}$ should be 1.67 V
LCDC1 0-----11	LCDIEN	0	LCD Frame Interrupts disabled
	LCDWAI	1	LCD is "off" in WAIT mode
	LCDSTP	1	LCD is "off" in STOP mode

Table 11-28. Initialization Register Values for Example 3 (continued)

Register	bit or bit field	Binary Value	Comment
LCDC0 00101111	LDEN	0	Initialization is done before initializing the LCD module
	SOURCE	0	Selects OSCOUT as the LCD clock source (32.768 KHz crystal)
	LCLK[2:0]	101	For 1/8 duty cycle, select closest value to the desired 30 Hz LCD frame frequency (see <a href="#">Table 11-13</a> )
	DUTY[2:0]	111	For 168 segments (8x21), select 1/8 duty cycle
LCDBCTL XXX-0010	BLINK	X	Blinking is turned on or off during LCD operation
	ALT	X	Alternate bit is configured during LCD operation
	BLANK	X	Blank bit is configured during LCD operation
	BMODE	0	Blink to a blank mode
	BRATE[2:0]	010	Select 2 Hz blink frequency using <a href="#">Table 11-18</a>
LCDPEN[7:0]	LCDPEN0 LCDPEN1 LCDPEN2 LCDPEN3	11111111 11111111 11111111 00011111	29 LCD pins need to be enabled.
LCDBPEN[7:0]	LCDBPEN0 LCDBPEN1 LCDBPEN2 LCDBPEN3	11111111 00000000 00000000 00000000	Eight backplane pins needed. <b>Note:</b> Any enabled LCD pin can be enabled to operate as a backplane
LCDWF[63:0]	LCDWF0 LCDWF1 LCDWF2 LCDWF3 LCDWF4 LCDWF5 LCDWF6 LCDWF7	00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000	Configure which phase the eight backplane pins will be active in. This configuration sets LCD[0] to be active in Phase A, LCD[1] to be active in Phase B. . . etc.  This configuration sets LCD pins 0-7 to represent backplane 1-8.  <b>Note:</b> Any backplane pin can be active in any phase

## 11.6 Application Information

[Figure 11-17](#) is a programmer's model of the LCD module. The programmer's model groups the LCD module register bit and bit field into functional groups. The model is a high-level illustration of the LCD module showing the module's functional hierarchy including initialization and runtime control.



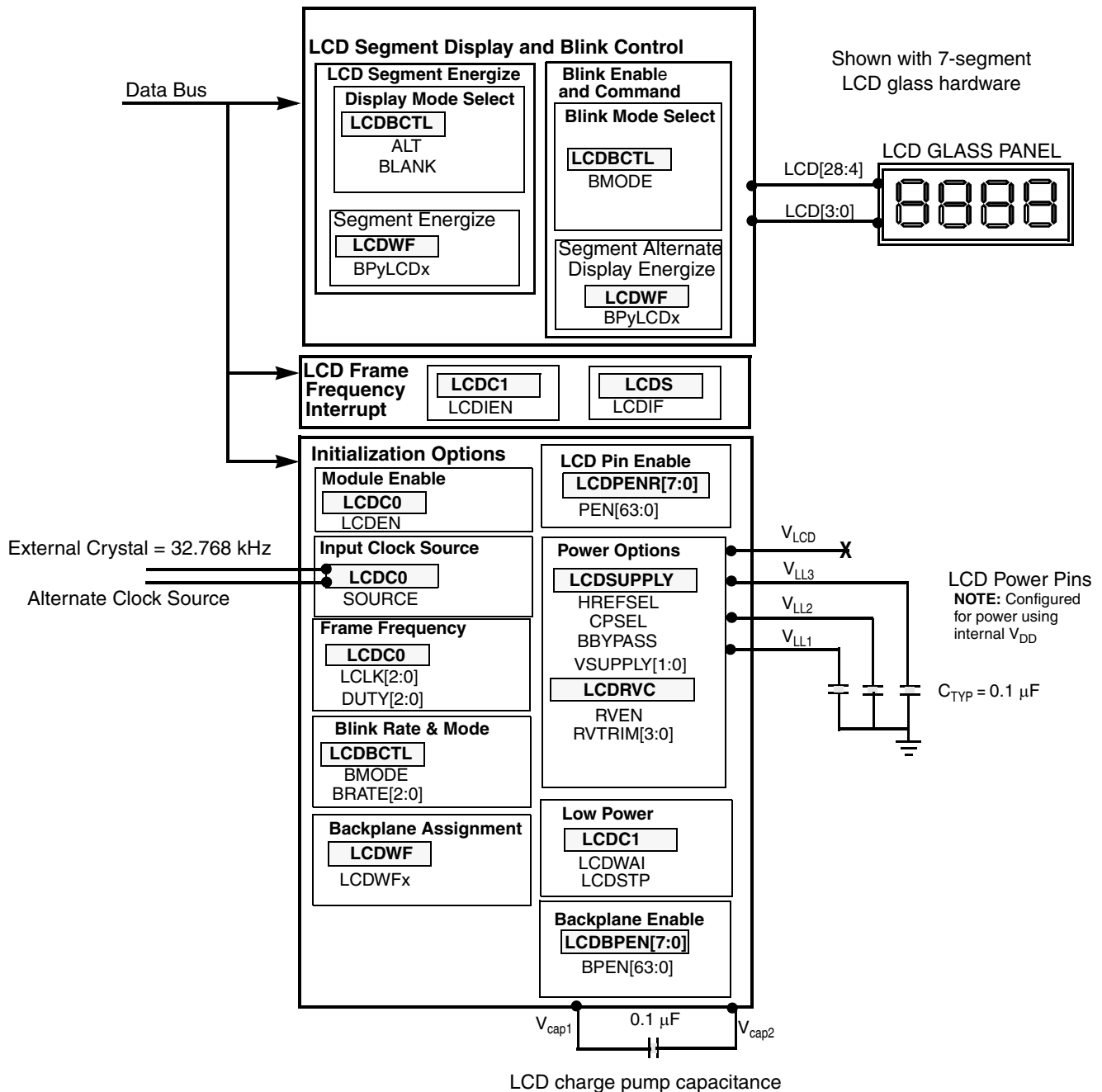
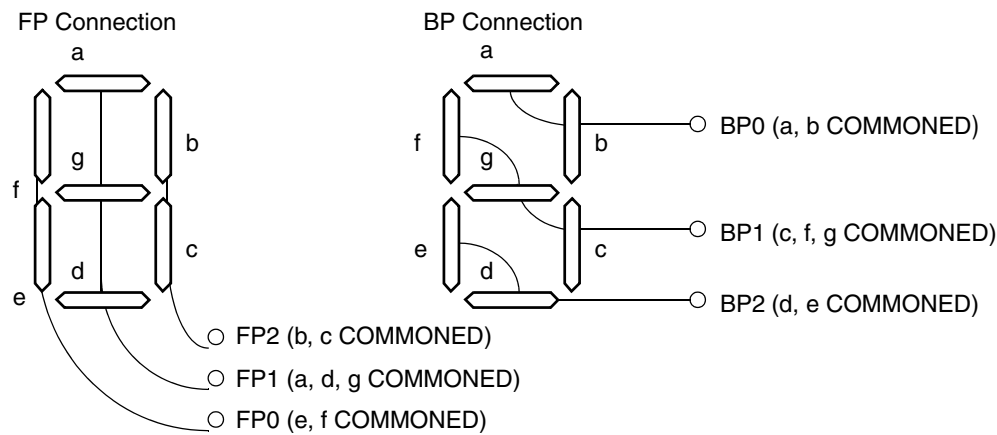


Figure 11-17. LCD Programmer's Model Diagram

### 11.6.1 LCD Seven Segment Example Description

A description of the connection between the LCD module and a seven segment LCD character is illustrated below to provide a basic example for a 1/3 duty cycle LCD implementation. The example uses three backplane pins (LCD[3], LCD[4] and LCD[5] and 3 frontplane pins (LCD[0], LCD[1], and LCD[2]).

LCDWF contents and output waveforms are also shown. Output waveforms are illustrated in [Figure 11-18](#) and [Figure 11-19](#).



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

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

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

With the above conditions the segment assignment is shown below:

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

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

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

X = don't care

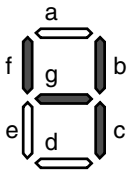


Figure 11-18. Waveform Output from LCDWF Registers

11.6.1.1 LCD Module Waveforms

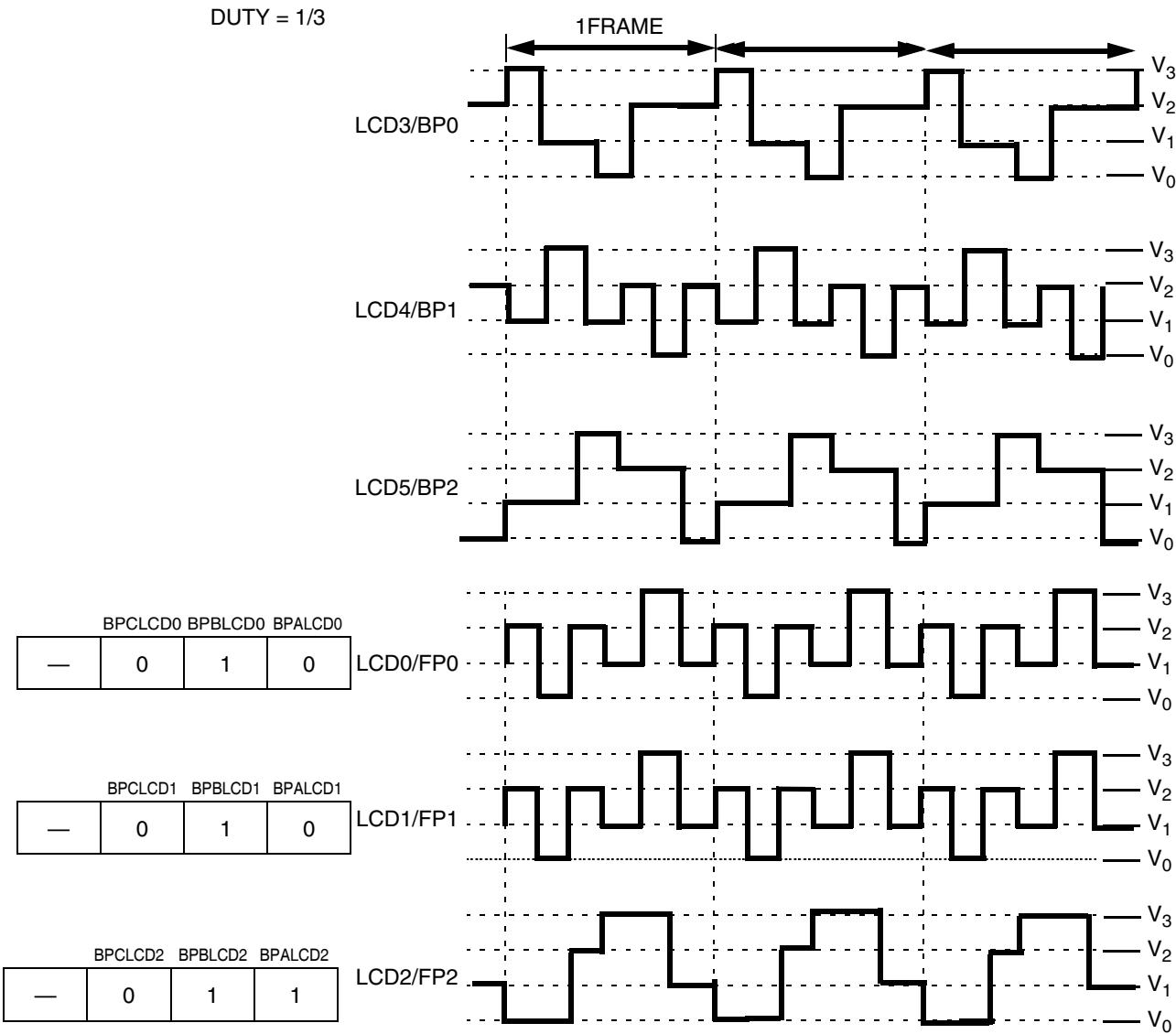


Figure 11-19. LCD Waveforms

### 11.6.1.2 Segment On Driving Waveform

The voltage waveform across the “f” segment of the LCD (between LCD[4]/BP1 and LCD[0]/FP0) is illustrated in Figure 11-20. As shown in the waveform, the voltage level reaches the value  $V_3$  therefore the segment will be on.

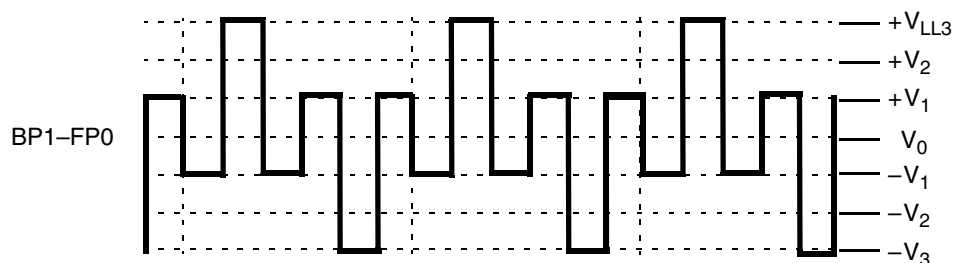


Figure 11-20. “f” Segment Voltage Waveform

### 11.6.1.3 Segment Off Driving Waveform

The voltage waveform across the “e” segment of the LCD (between LCD[5]/BP2 and LCD[0]/FP0) is illustrated in Figure 11-21. As shown in the waveform, the voltage does not reach the voltage  $V_3$  threshold therefore the segment will be off.

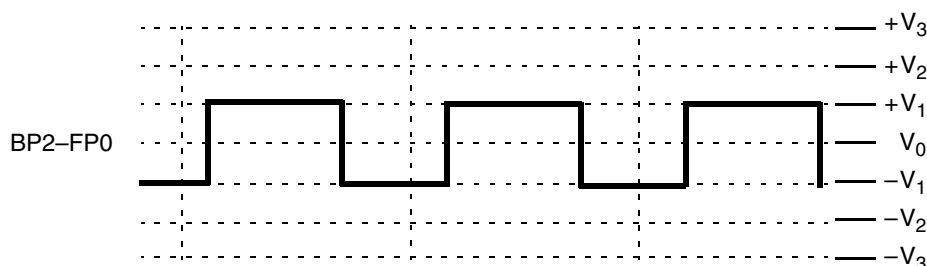


Figure 11-21. “e” Segment Voltage Waveform

## 11.6.2 LCD Contrast Control

Contrast control for the LCD module is achieved when the LCD power supply is adjusted above and below the LCD threshold voltage. The LCD threshold voltage is the nominal voltage required to energize the LCD segments. For 3-V LCD glass, the LCD threshold voltage is 3 V; while for 5-V LCD glass it is 5 V. By increasing the value of the LCD voltage, the energized segments on the LCD glass will become more opaque. Decreasing the value of the LCD voltage makes the energized segments on the LCD glass become more transparent. The LCD power supply can be adjusted to facilitate contrast control by using external components like a variable resistor. Figure 11-22 shows two circuits that can be used to implement contrast control.

Additionally, if the internally regulated voltage source ( $V_{IREG}$ ) is used to power the LCD glass, contrast control can be achieved by software alone. Using the RVTRIM[3:0] bits, the  $V_{IREG}$  can be increased or decreased by steps of 1.5% for each count in the trim register. Just as with an external circuit, increasing the value of the LCD voltage will cause the energized segments on the LCD glass to become more opaque.

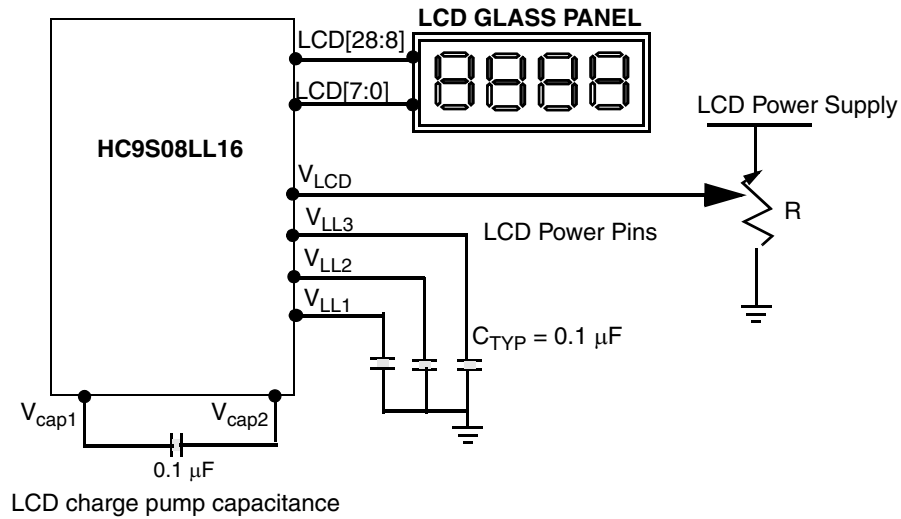
Decreasing the value of the LCD voltage makes the energized segments on the LCD glass more transparent.

**NOTE:**

Contrast control configuration when LCD is powered using external  $V_{LCD}$

This is the recommended configuration for contrast control.

$V_{LCD}$  specified between 1.4 and 1.8 V.

**NOTE:**

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

$V_{DD}$  is specified between 1.8 and 3.6 V.

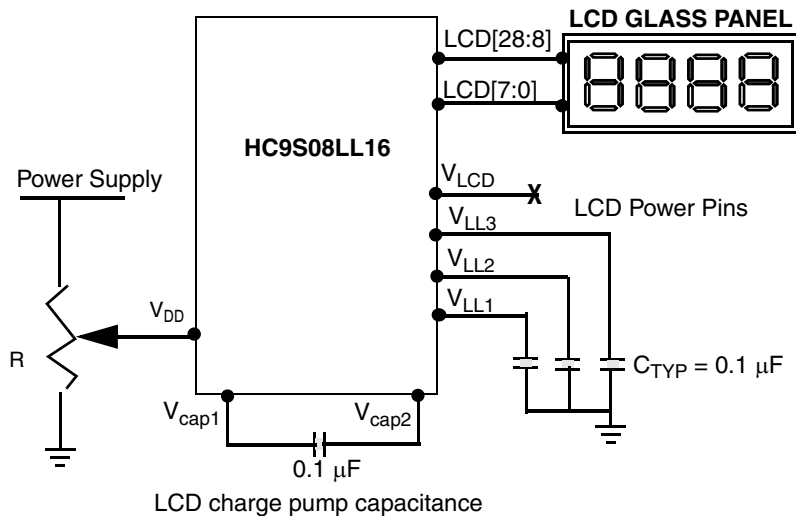


Figure 11-22. Power Connections for Contrast Control

### 11.6.3 Stop Mode Recovery

When the MCU recovers from stop2 mode a reset sequence is initiated. All Control Registers should be re-written before the stop2 recovery acknowledge bit is set. The Registers LCDBPEN, LCDPEN and the LCDFWF retain their values upon stop2 recovery and do not need to be re-written. For more information on how to perform stop recovery please refer to AN2493.



## Chapter 12

# Inter-Integrated Circuit (S08IICV5)

### 12.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface can operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, 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. Support System Management Bus Specification (SMBus), version 2.

#### NOTE

The SDA and SCL can be driven to 5 V which is above  $V_{DD}$  on the PTB4 and PTB5, but they can not be driven above  $V_{DD}$  on the PTA0 and PTA1.

#### 12.1.1 IIC Clock Gating

The bus clock to the IIC can be gated on and off using the IIC bit in SCGC1. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the IIC bit can be cleared to disable the clock to this module when. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### NOTE

- DMAEN bit in IICC1 is not supported in this version.
- HDRS, SBRC bits in IICC2 are not supported in this version.
- FLT width in IICFLT register is [3:0] in this version.
- IPBUS clock is used for filtering logic rather than half IPBUS.

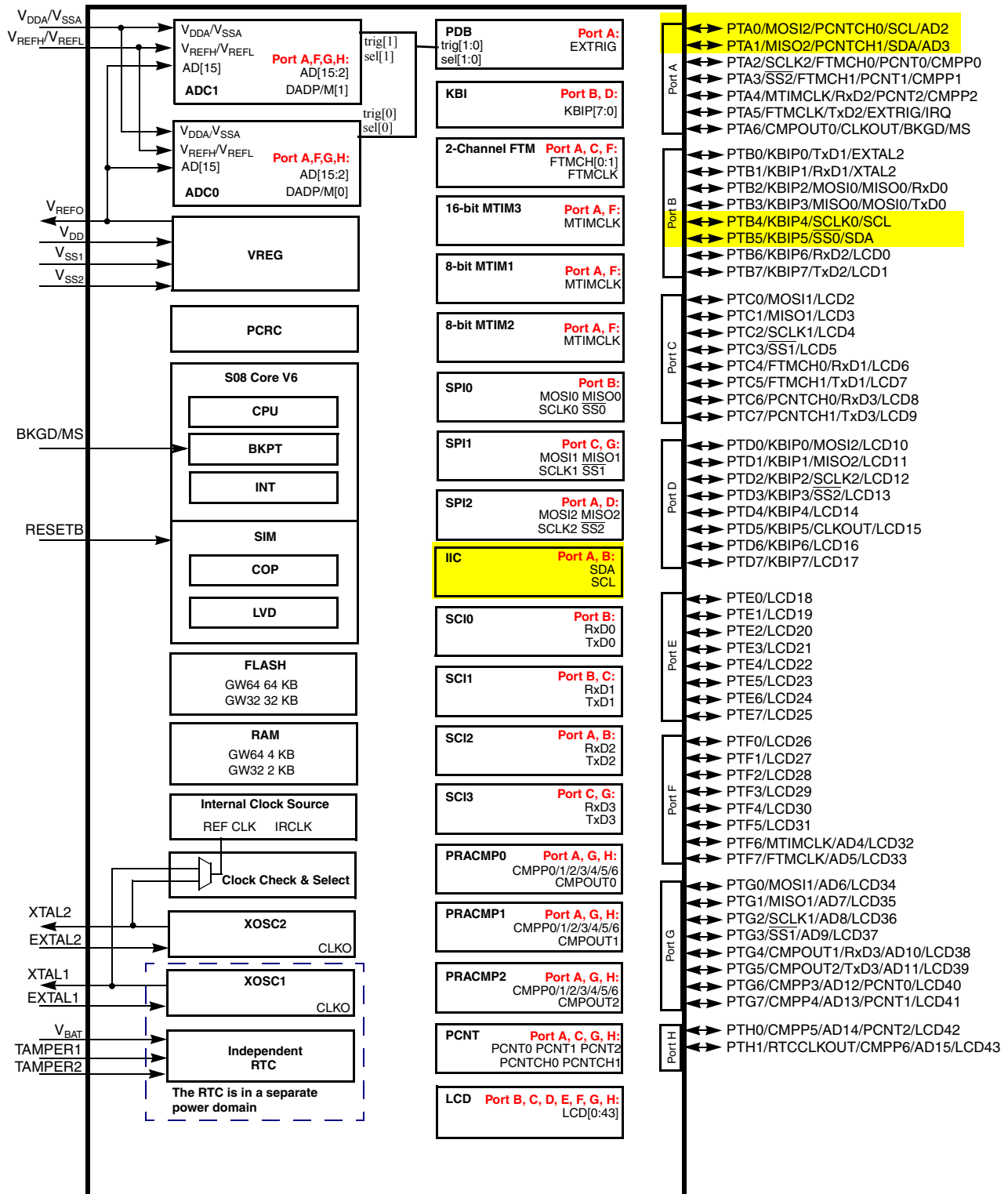


Figure 12-1. MC9S08GW64 Series Block Diagram Highlighting the IIC Block and Pins



## 12.1.2 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Repeated START signal generation/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
- Address matching causes wakeup when MCU is in stop3 mode.

## 12.1.3 Modes of Operation

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

- Run mode — This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode — The module continues to operate when the MCU is in wait mode and can provide a wakeup interrupt.
- Stop mode — The IIC is inactive in stop3 mode for reduced power consumption except address matching is enabled in stop3 mode. The STOP instruction does not affect IIC register states. In stop2 mode the register contents are reset.

## 12.1.4 Block Diagram

Figure 12-2 provides a block diagram of the IIC module.

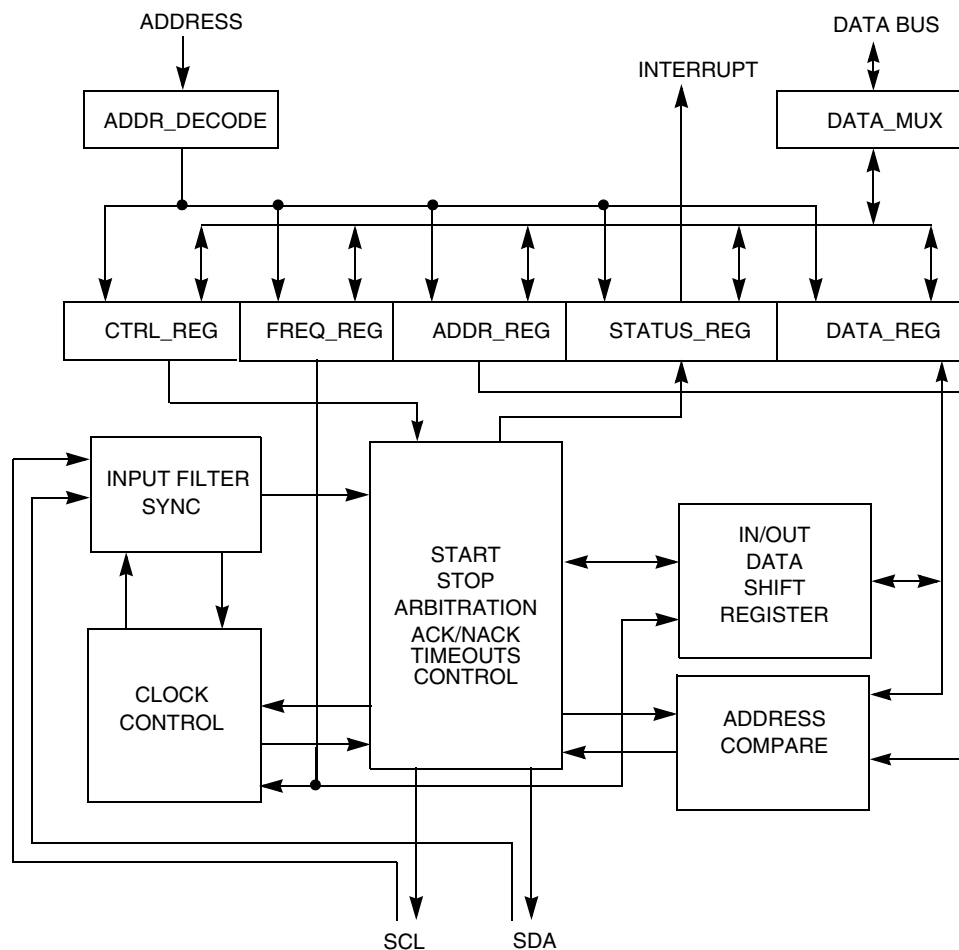


Figure 12-2. IIC Functional Block Diagram

## 12.2 External Signal Description

This section describes each user-accessible pin signal.

### 12.2.1 SCL — Serial Clock Line

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

### 12.2.2 SDA — Serial Data Line

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

## 12.3 Register Definition

### 12.3.1 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 12-1](#) summarizes the IIC module's address space. The following section describes the bit-level arrangement and functionality of each register.

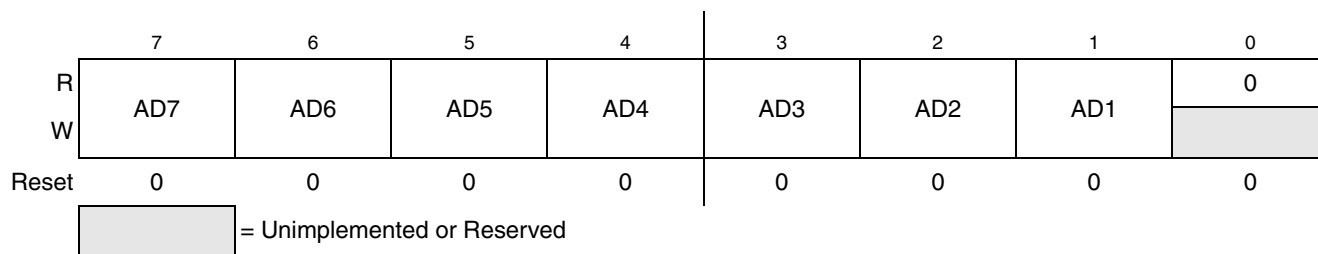
**Table 12-1. Module Memory Map**

Address	Use	Access
Base + 0x0000	IIC Address Register 1 (IICA1)	read/write
Base + 0x0001	IIC Frequency Divider Register (IICF)	read/write
Base + 0x0002	IIC Control Register 1 (IICC1)	read/write
Base + 0x0003	IIC Status Register (IICS)	read
Base + 0x0004	IIC Data IO Register (IICD)	read/write
Base + 0x0005	IIC Control Register 2 (IICC2)	read/write
Base + 0x0006	IIC input programmable filter (IICFLT)	read/write
Base + 0x0007	SMBUS IIC Control and Status Register (IICSMB)	read/write
Base + 0x0008	IIC Address Register 2 (IICA2)	read/write
Base + 0x0009	IIC SCL Low Time Out Register High (IICSLTH)	read/write
Base + 0x000A	IIC SCL Low Time Out Register Low (IICSLTL)	read/write

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

Refer to the direct-page register summary in the [Chapter 4](#), “Memory,” 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.

### 12.3.2 IIC Address Register 1 (IICA1)



**Figure 12-3. IIC Address Register 1 (IICA1)**

Table 12-2. 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.

### 12.3.3 IIC Frequency Divider Register (IICF)

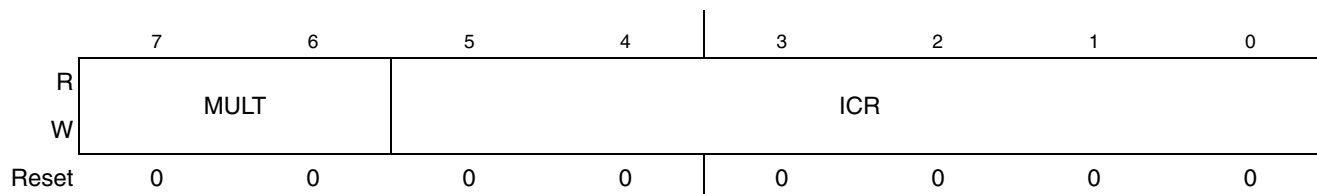


Figure 12-4. IIC Frequency Divider Register (IICF)

Table 12-3. 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	<p><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 12-4</a> provides the SCL divider and hold values for corresponding values of the ICR.</p> <p>The SCL divider multiplied by multiplier factor mul is used to generate IIC baud rate.</p> <p style="text-align: center;"><b>IIC baud rate = bus speed (Hz)/(mul × SCL divider)</b> <span style="float: right;"><b>Eqn. 12-1</b></span></p> <p>SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data).</p> <p style="text-align: center;"><b>SDA hold time = bus period (s) × mul × SDA hold value</b> <span style="float: right;"><b>Eqn. 12-2</b></span></p> <p>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).</p> <p style="text-align: center;"><b>SCL Start hold time = bus period (s) × mul × SCL Start hold value</b> <span style="float: right;"><b>Eqn. 12-3</b></span></p> <p>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).</p> <p style="text-align: center;"><b>SCL Stop hold time = bus period (s) × mul × SCL Stop hold value</b> <span style="float: right;"><b>Eqn. 12-4</b></span></p>

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

MULT	ICR	Hold times ( $\mu$ 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 12-4. IIC Divider and Hold Values

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

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

## 12.3.4 IIC Control Register (IICC1)

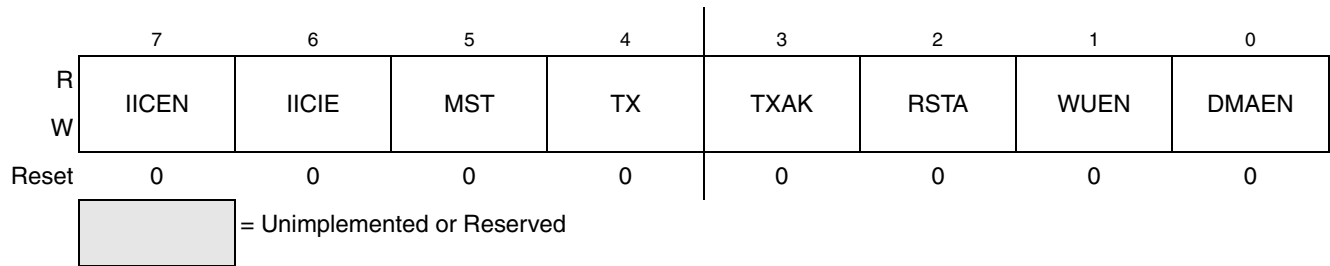


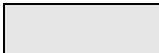
Figure 12-5. IIC Control Register (IICC1)

Table 12-5. 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 0 to 1, a START signal is generated on the bus and master mode is selected. When this bit changes from 1 to 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 must be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. When addressed as a slave this bit must 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 that effect NACK/ACK generation. If FACK (fast NACK/ACK) is cleared, 0 An acknowledge signal is sent 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 FACK bit is set, no ACK or NACK is sent after receiving one data byte until this TXAK bit is written 0 An acknowledge signal is 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 <b>Note:</b> SCL is held to low until TXAK is written.
2 RSTA (Write Only read always 0)	<b>Repeat START</b> — Writing 1 to this bit generates a repeated START condition provided it is the current master. Attempting a repeat at the wrong time results in loss of arbitration. 0 No repeat start detected in bus operation. 1 Repeat start generated.
1 WUEN	<b>Wakeup Enable</b> — IIC can wake the MCU from stop3 mode when slave address matching occurs. 0 Normal operation. No interrupt generated when address matching in stop3 mode. 1 Enables the wakeup function in stop3 mode.

## 12.3.5 IIC Status Register (IICS)

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

 = Unimplemented or Reserved

**Figure 12-6. IIC Status Register (IICS)**

**Table 12-6. 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: <ul style="list-style-type: none"> <li>When the calling address matches the programmed slave address</li> <li>If the GCAEN bit is set and a general call is received</li> <li>If SIICAEN bit is set, when the calling address matches the 2nd programmed slave address</li> <li>If ALERTEN bit is set and SMBus alert response address is received</li> </ul> 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 or by writing 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 $R/\bar{W}$ 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 12-6. 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 or 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 excluding ACK/NCAK bit completes if FACK = 1. an ACK or NACK is sent on the bus by writing 0 or 1 to TXAK after this bit is set as receive mode.</li> <li>Match of slave addresses to calling address including primary slave address, general call address, alert response address, and second slave address.</li> <li>Arbitration lost</li> <li>Timeouts in SMBus mode except both SCL and SDA 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.

### 12.3.6 IIC Data I/O Register (IICD)

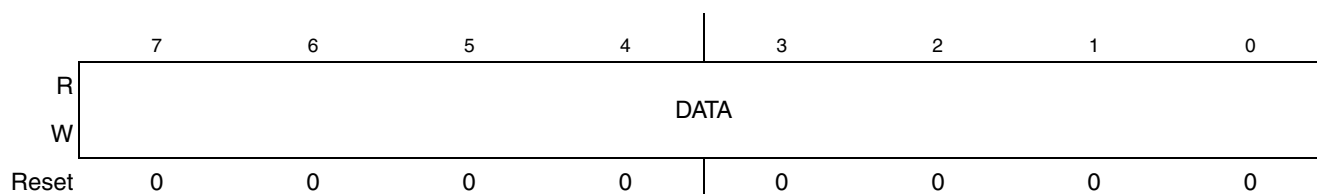


Figure 12-7. IIC Data I/O Register (IICD)

Table 12-7. 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 must 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.

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 does not initiate the receive.

Reading the IICD returns 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 must comprise of the calling address (in bit 7 to bit 1) concatenated with the required  $\overline{R/W}$  bit (in position bit 0).

12.3.7 IIC Control Register 2 (IICC2)

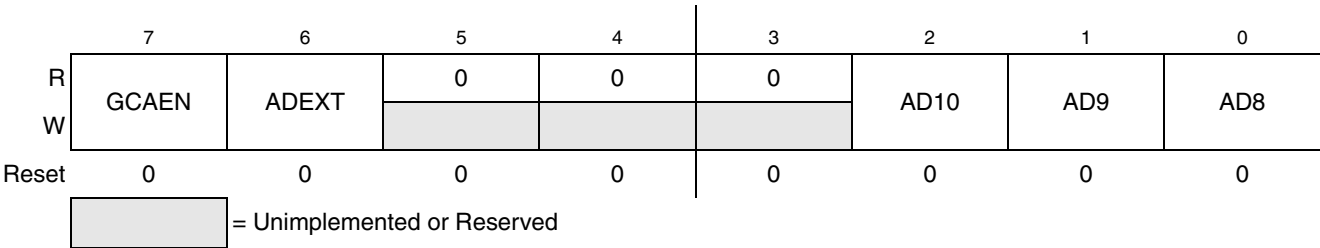


Figure 12-8. IIC Control Register 2 (IICC2)

Table 12-8. 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.

12.3.8 IIC Programmable Input Glitch Filter (IICFLT)

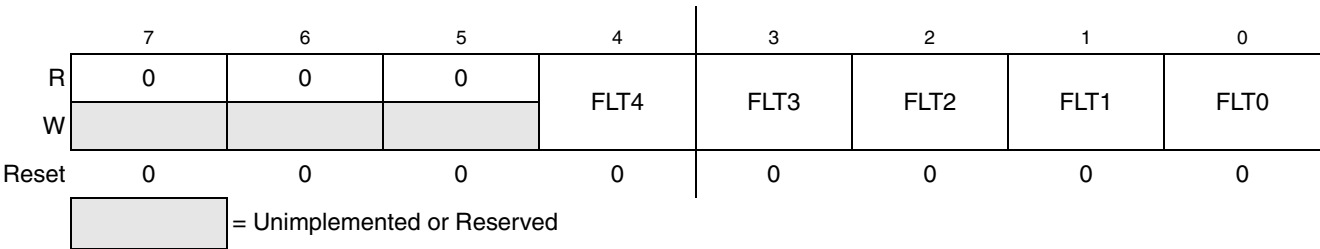


Figure 12-9. IIC Programmable Input Glitch Filter Register (IICFLT)

Table 12-9. IICFLT Field Descriptions

Field	Description
4:0 FLT	<p>IIC Programmable Filter Factor contains the programming controls for the width of glitch (in terms of bus clock cycles) the filter must absorb; in other words, the filter does not let glitches less than or equal to this width setting pass. For instance: FLT[3:0]</p> <p>0000 No Filter / Bypass  0001 Filter glitches up to width of 1 (half) IPBUS clock cycle  0010 Filter glitches up to width of 2 (half) IPBUS clock cycles  0011 Filter glitches up to width of 3 (half) IPBUS clock cycles  0100 Filter glitches up to width of 4 (half) IPBUS clock cycles  0101 Filter glitches up to width of 5 (half) IPBUS clock cycles  0110 Filter glitches up to width of 6 (half) IPBUS clock cycles  0111 Filter glitches up to width of 7 (half) IPBUS clock cycles  1000 Filter glitches up to width of 8 (half) IPBUS clock cycles  1001 Filter glitches up to width of 9 (half) IPBUS clock cycles  1010 Filter glitches up to width of 10 (half) IPBUS clock cycles  1011 Filter glitches up to width of 11 (half) IPBUS clock cycles  1100 Filter glitches up to width of 12 (half) IPBUS clock cycles  1101 Filter glitches up to width of 13 (half) IPBUS clock cycles  1110 Filter glitches up to width of 14 (half) IPBUS clock cycles  1111 Filter glitches up to width of 15 (half) IPBUS clock cycles</p> <p><b>Note:</b> The width of the FLT is an integration option which can be changed in different SoCs. Also the clock source used is an integration configurative option - It could be the 2X IPBus clock or the IPbus clock - which one needs to be identified at architectural definition. For the 4-bit definitions above, hard descriptions of "half" IPBUS clock cycles is not the case when the IPBUS clock is used for filtering logic.</p>

## 12.3.9 IIC SMBus Control and Status Register (IICSMB)

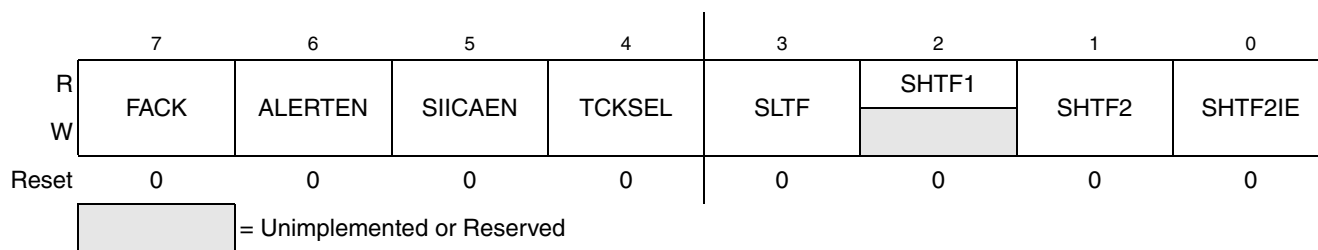


Figure 12-10. IIC SMBus Control and Status Register (IICSMB)

Table 12-10. IICSMB Field Descriptions


Field	Description
7 FACK	<b>Fast NACK/ACK enable</b> — For SMBus packet error checking, CPU must be able to issue an ACK or NACK according to the result of receiving data byte. 0 ACK or NACK is sent out on the following receiving data byte. 1 Writing 0 to TXAK after receiving data byte generates an ACK; writing 1 to TXAK after receiving data byte generates a NACK.
6 ALERTEN	<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	<b>Time Out Counter Clock Select</b> — 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	<b>SCL Low Timeout Flag</b> — This 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 LOW TIME OUT occurs. <b>Note:</b> LOW TIME OUT function is disabled when IIC SCL low timer out register is set to zero.
2 SHTF1	<b>SCL High Timeout Flag 1</b> — 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 SCL high and SDA high TIMEOUT occurs. 1 SCL high and SDA high TIMEOUT occurs.
1 SHTF2	<b>SCL High Timeout Flag 2</b> — This bit is set to logic 1 when SCL is held high and SDA is held low more than clock × LoValue/512. This bit is cleared by software, by writing a logic 1 to it 0 No SCL high and SDA low TIMEOUT occurs. 1 SCL high and SDA low TIMEOUT occurs.
0 SHTF2IE	<b>SHTF2 Interrupt enable</b> — This bit is Interrupt enable for SCL high and SDA low timeout. 0 SHTF2 interrupt is disabled. 1 SHTF2 interrupt is enabled.

**NOTE**

- A master assumes that the bus is free when detecting the clock and data signals being high for greater than high time out period. However, the SHTF1 rises in bus transmission process with idle bus state.
- When TCKSEL=1, there is no meaning to monitor SHTF1 since the bus speed is too high to match the protocol of SMBus.

**12.3.10 IIC Address Register 2 (IICA2)**

	7	6	5	4	3	2	1	0
R	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0
W								
Reset	1	1	0	0	0	0	1	0


 = Unimplemented or Reserved

**Table i**

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 addresses.

**12.3.11 IIC SCL Low Time Out Register High (IICSLTH)**

	7	6	5	4	3	2	1	0
R	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
W								
Reset	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 12-11. IIC SCL Low Time Out Register High (IICSLTH)****Table 12-11. IICSLTH Field Descriptions**

Field	Description
7:0 SSLT[15:8]	<b>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.</b>

12.3.12 IIC SCL Low Time Out register Low (IICSLTL)

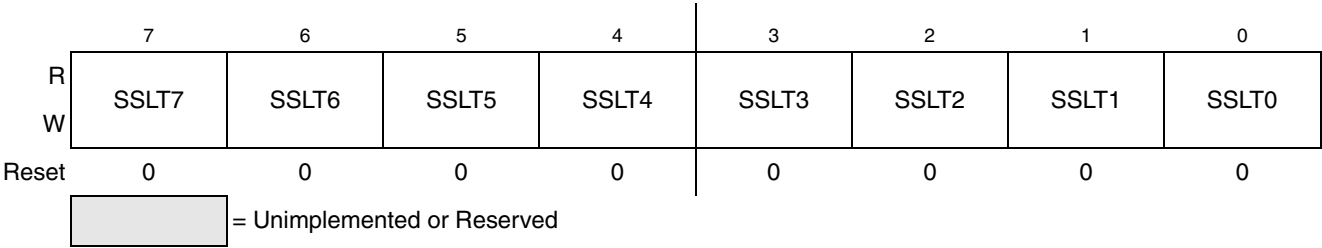


Figure 12-12. IIC SCL Low Time Out register Low (IICSLTL)

Table 12-12. IICSLTL Field Descriptions

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 timeout period of SCL low.

## 12.4 Functional Description

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

### 12.4.1 IIC Protocol

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

Normally, a standard communication is composed of four parts:

- START signal
- Slave address transmission
- Data transfer
- STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The IIC bus system communication is described briefly in the following sections and is illustrated in [Figure 12-13](#).

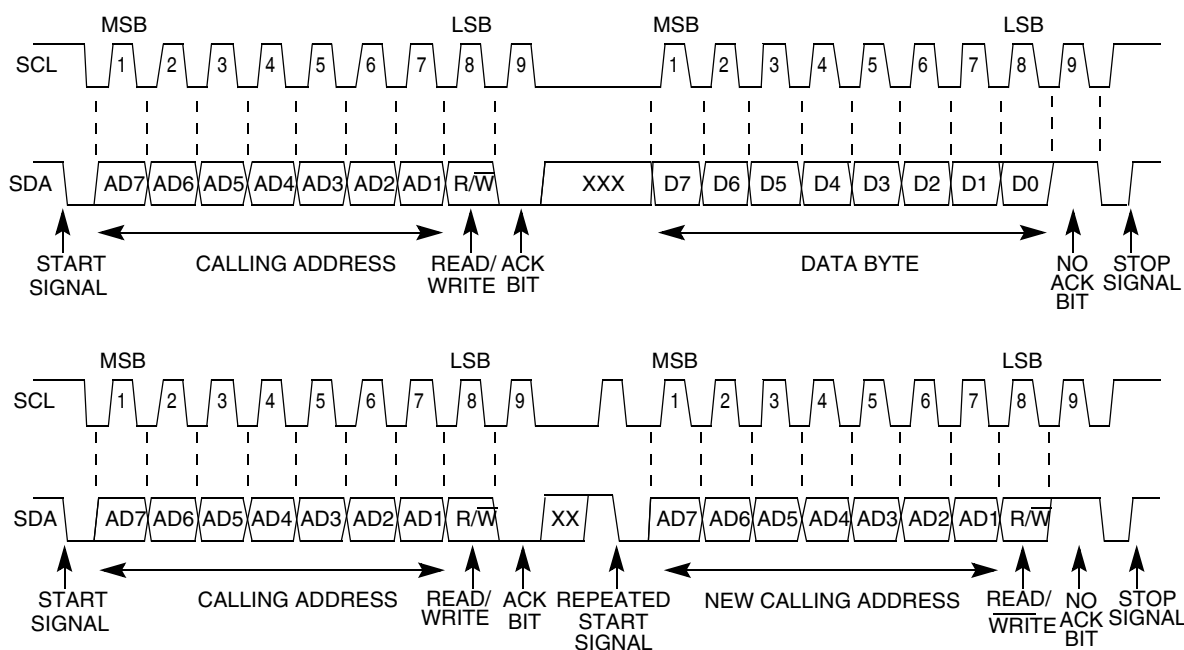


Figure 12-13. IIC Bus Transmission Signals

#### 12.4.1.1 START Signal

When the bus is free that is, 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 12-13](#), 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.

#### 12.4.1.2 Slave Address Transmission

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

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

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

Only the slave with a calling address that matches the one transmitted by the master responds by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see [Figure 12-13](#)).

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 reverts to slave mode and operates correctly even if it is being addressed by another master.

#### 12.4.1.3 Data Transfer

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

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

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in [Figure 12-13](#). 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 that 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.

#### 12.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see [Figure 12-13](#)).



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

#### 12.4.1.5 Repeated START Signal

As shown in [Figure 12-13](#), 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.

#### 12.4.1.6 Arbitration Procedure

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

#### 12.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see [Figure 12-14](#)). 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.

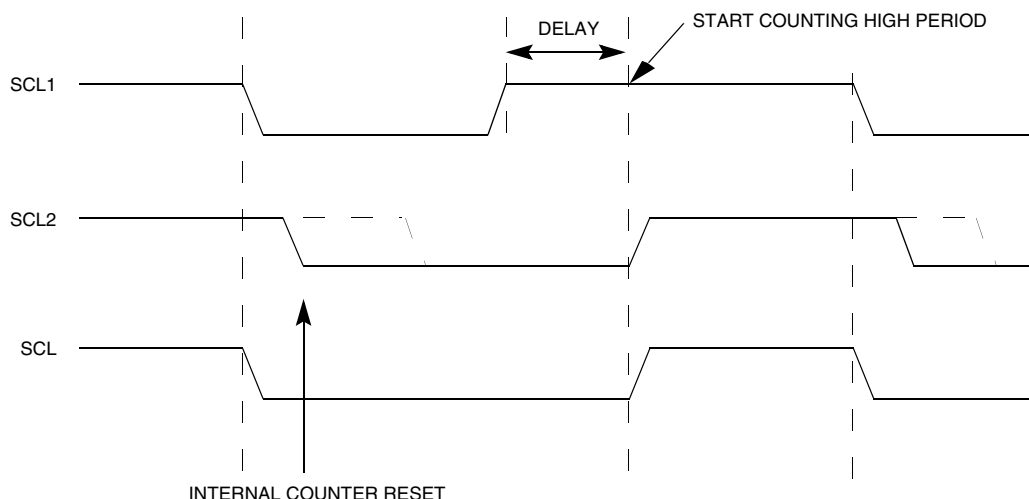


Figure 12-14. IIC Clock Synchronization

### 12.4.1.8 Handshaking

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

### 12.4.1.9 Clock Stretching

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

## 12.4.2 10-bit Address

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

### 12.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 12-13). When a 10-bit address follows a START condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit ( $R/\overline{W}$  direction bit) is 0. It is possible that more than one device finds a match and generates an acknowledge (A1). Each slave that finds a match compares the eight bits of the second byte of the slave address with its own address, but only one slave finds a match and generate an acknowledge (A2). The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

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

**Table 12-13. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address**

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

### 12.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second  $R/\overline{W}$  bit (see Table 12-14). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and tests whether the eighth ( $R/\overline{W}$ ) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

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

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

**Table 12-14. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address**

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

### 12.4.3 Address Matching

All received addresses can be requested in 7-bit or 10-bit address. IIC address register 1 that contains IIC primary slave address, always participates the address matching process. If the GCAEN bit is set, general call participates the address matching process. If the ALERTEN bit is set, alert response participates the address matching process. If SIICAEN bit is set, the IIC address register 2 participates the address matching process.

When the IIC responds to one of the 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 that the address is matched.

### 12.4.4 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.

#### 12.4.4.1 Timeouts

The  $T_{\text{TIMEOUT,MIN}}$  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  $T_{\text{TIMEOUT,MIN}}$ . Devices that have detected this condition must reset their communication and be able to receive a new START condition in no later than  $T_{\text{TIMEOUT,MAX}}$ .

SMBus defines a clock low time out,  $T_{\text{TIMEOUT}}$  of 35 ms and specifies  $T_{\text{LOW:SEXT}}$  as the cumulative clock low extend time for a slave device and specifies  $T_{\text{LOW:MEXT}}$  as the cumulative clock low extend time for a master device.

##### 12.4.4.1.1 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  $T_{\text{TIMEOUT,MIN}}$  it must generate a stop condition within or after the current data byte in the transfer process. When slave, upon detection of the  $T_{\text{TIMEOUT,MIN}}$  condition, the IIC shall reset its communication and be able to receive a new START condition.

### 12.4.4.1.2 SCL High 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  $T_{\text{HIGH:MAX}}$ . 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 SHTF1 rises. HIGH timeout occurred in scenario 2 if it ever detects that both the following is true: BUSY bit is high and SHTF1 is high.

When SMBDAT signal is low and SMBCLK signal is high for a period of time, the other kind of time out occurs. The time period needs to be defined in software. SHTF2 is used as the flag when limited time reaches. This flag is also an interrupt resource, therefore it also triggers IICIF.

### 12.4.4.1.3 CSMBCLK TIMEOUT MEXT

Figure 12-15 illustrates the definition of the timeout intervals,  $T_{\text{LOW:SEXT}}$  and  $T_{\text{LOW:MEXT}}$ . When master mode, the IIC must not cumulatively extend its clock cycles for a period greater than  $T_{\text{LOW:MEXT}}$  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 rises and also triggers the SLTF.

### 12.4.4.1.4 CSMBCLK TIMEOUT SEXT

A master is allowed to abort the transaction in progress to any slave that violates the  $T_{\text{LOW:SEXT}}$  or  $T_{\text{TIMEOUT,MIN}}$  specifications. This can be accomplished by the master issuing a STOP condition at the conclusion of the byte transfer in progress. When slave, the IIC must not cumulatively extend its clock cycles for a period greater than  $T_{\text{LOW:SEXT}}$  during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs, SEXT rises and also triggers SLTF.

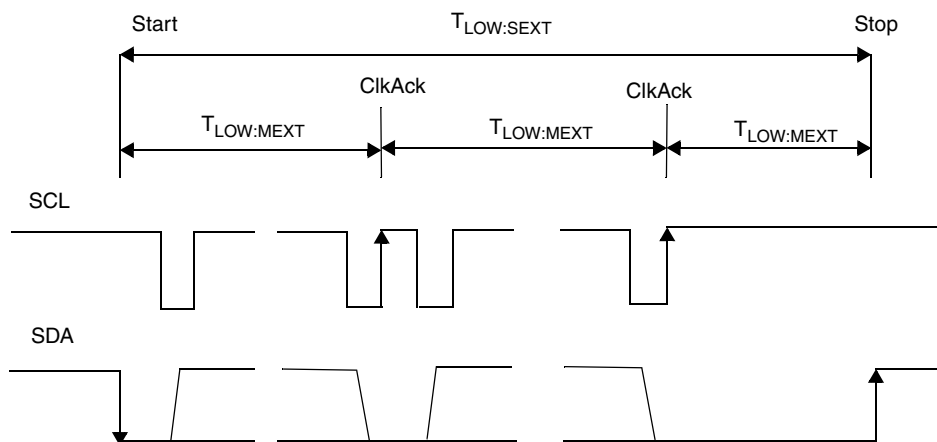


Figure 12-15. Timeout measurement intervals

#### NOTE

CSMBCLK TIMEOUT SEXT and MEXT are optional functions that are implemented in second step.

### 12.4.4.2 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 is 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 must send NACK to bus, so FACK must be switched off before the last byte transmit.

## 12.5 Resets

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

## 12.6 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in [Table 12-15](#) 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 1 to it in the interrupt routine. You 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 1 to it in the interrupt routine. You can determine the interrupt type by reading the status register.

#### NOTE

In master receive mode, the FACK must be set to zero before the last byte transfer.

**Table 12-15. Interrupt Summary**

Interrupt Source	Status	Flag	Local Enable
Complete 1-byte transfer	TCF	IICIF	IICIE

**Table 12-15. Interrupt Summary**

Match of received calling address	IAAS	IICIF	IICIE
Arbitration lost	ARBL	IICIF	IICIE
SMBus SCL low timeout interrupt flag	SLTF	IICIF	IICIE
SMBus SCL high SDA low timeout interrupt flag	SHTF2	IICIF	IICIE&SHTF2IE
Wakeup from stop3 interrupt	IAAS	IICIF	IICIE&WUEN

### 12.6.1 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 and acknowledge transfer.

When FACK is enabled, TCF is then set at the falling edge of 8th clock to indicate the completion of byte.

### 12.6.2 Address Detect Interrupt

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

### 12.6.3 Exit from Low-Power/Stop Modes

The slave receive input detect circuit and address matching feature are still active on low power modes (wait and stop). An asynchronous input matching slave address or general call address brings the CPU out of low power/stop mode if the interrupt is not masked. Therefore, TCF and IAAS both can trigger this interrupt.

### 12.6.4 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.

## 12.6.5 Timeouts Interrupt in SMBus

When IICIE is set, the IIC asserts a timeout interrupt outputs SLTF and SHTF2 upon detection of any of the mentioned timeout conditions, with one exception. The SCL high and SDA high TIMEOUT mechanism must not be used to influence the timeout interrupt output, because this timeout indicates an idle condition on the bus. SHTF1 rises when it matches the SCL high and SDA high TIMEOUT and fall automatically to just indicate the bus status. The SHTF2's timeout period is same as SHTF1 that is short as compared to SLTF, so another control bit SHTF2IE is added to enable or disable it.

## 12.6.6 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 IIC 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.

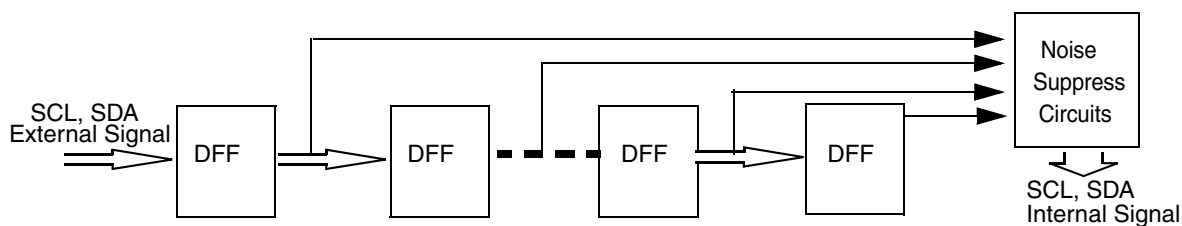


Figure 12-16. Programmable input glitch filter diagram

## 12.6.7 Address Matching Wakeup

When address matching happens as IIC works in slave receive mode, the MCU wakes from stop3 mode. After the address matching IAAS bit is set, an interrupt is sent out at the end of address matching to wake up the MCU. The IAAS bit must be cleared after the clock recovery.

### NOTE

After the system was recovered to run mode IIC must restart if it is needed to work. The SCL line will not be hold low until the IIC resets after address matching.



## 12.7 Initialization/Application Information

### Module Initialization (Slave)

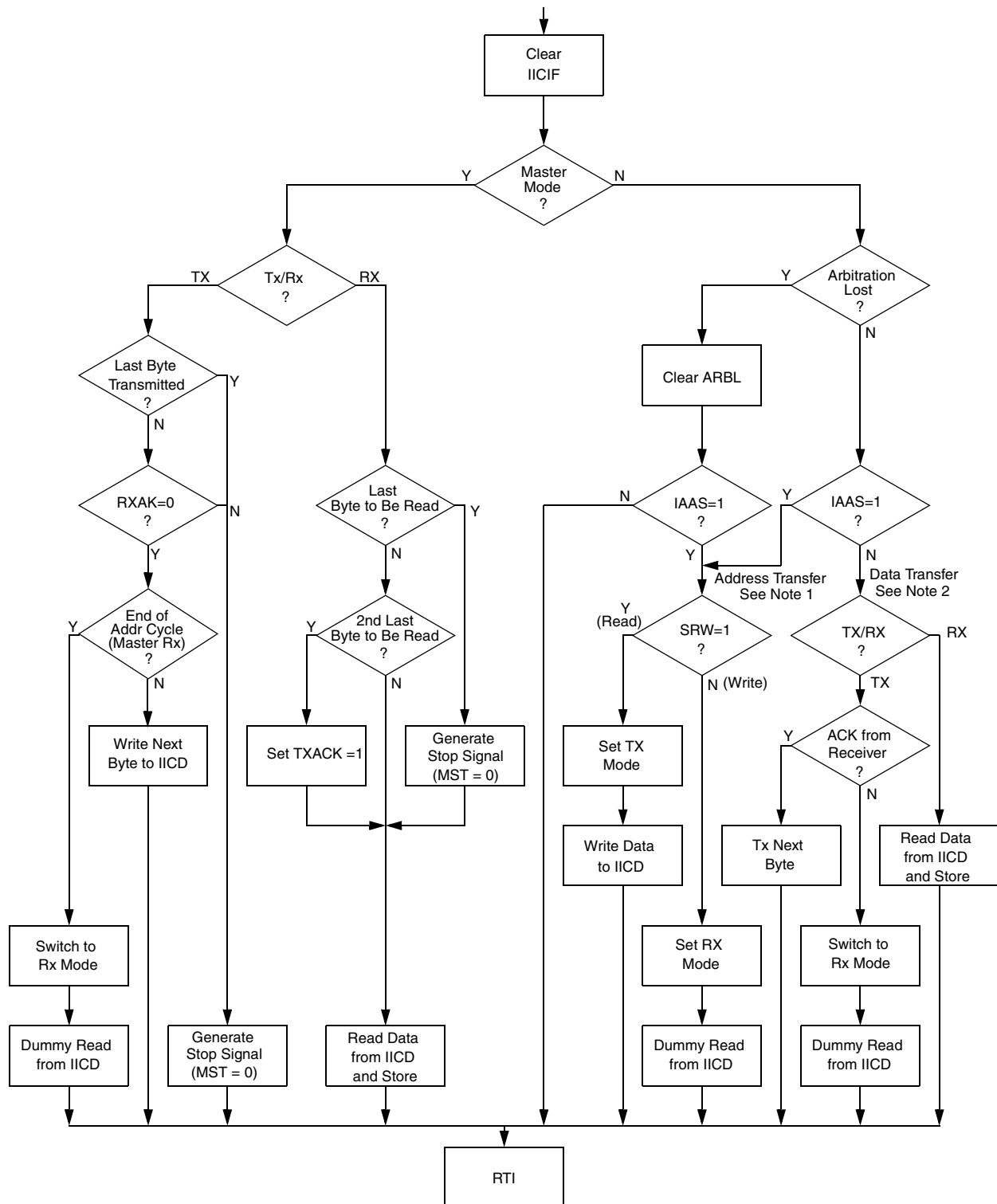
1. Write: IICC2
  - to enable or disable general call
  - to select 10-bit or 7-bit addressing mode
2. Write: IICA1
  - to set the slave address
3. Write: IICC1
  - to enable IIC and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in [Figure 12-17](#)

### Module Initialization (Master)

1. Write: IICF
  - to set the IIC baud rate (example provided in this chapter)
2. Write: IICC1
  - to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in [Figure 12-17](#)
5. Write: IICC1
  - to enable TX
6. Write: IICC1

## Register Model

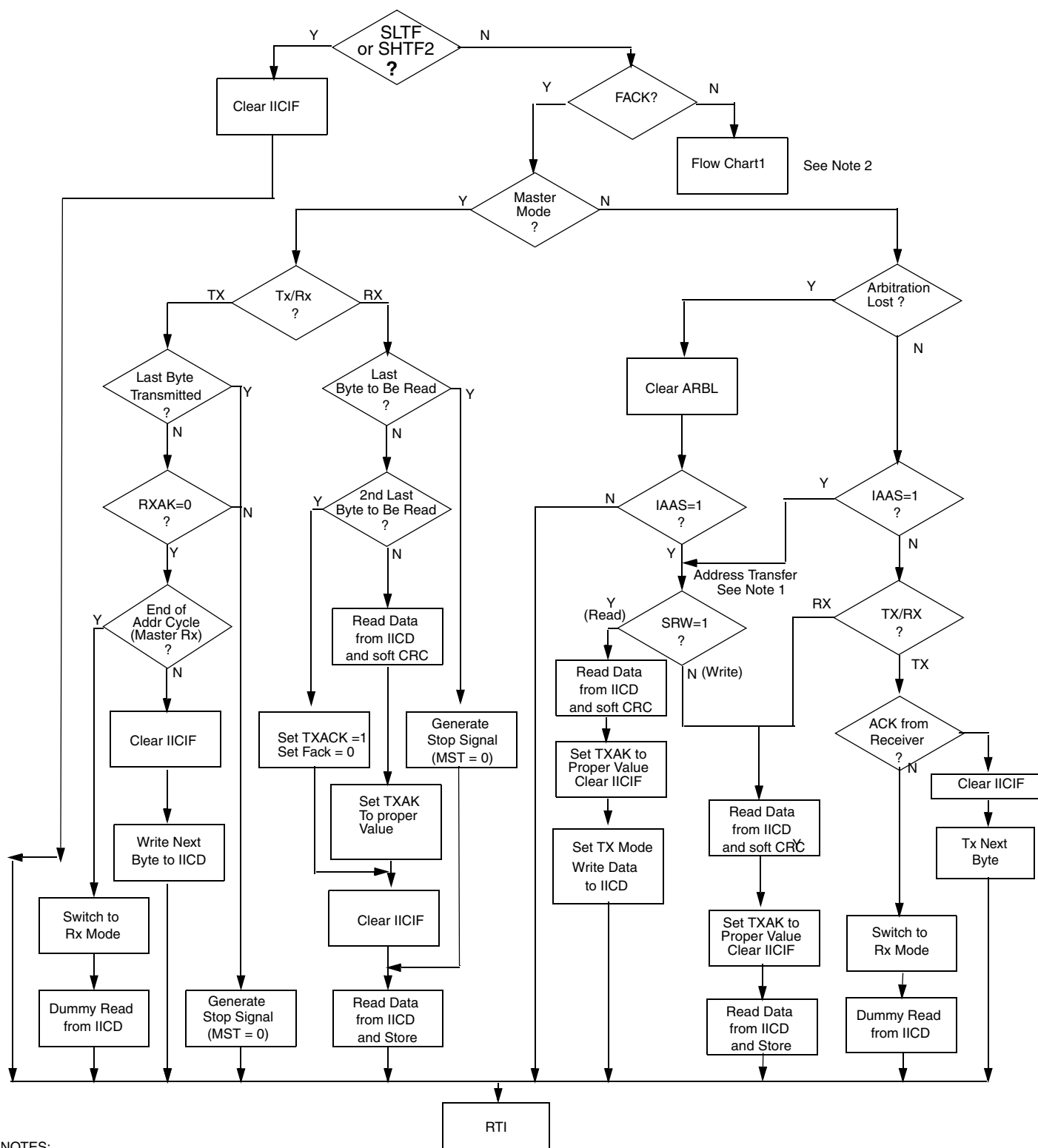
IICA1	AD[7:1]							0
Address to which the module responds when addressed as a slave (in slave mode)								
IICF	MULT		ICR					
Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER))								
IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	WUEN	0
Module configuration								
IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
Module 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								
IICFLT	0	0	0	0	FLT3	FLT2	FLT1	FLT0
IIC Programmable Input Glitch Filter								
IICSMB	FAACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
IIC SMBus Control and Status Register								
IICA2	SAD[7:1]							0
IIC Address Register 2								
IICSLTH	SSLT[15:8]							
IIC SCL Low Time Out Register High								
IICSLTL	SSLT[7:0]							
IIC SCL Low Time Out Register Low								



## NOTES:

1. If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.
2. When 10-bit addressing is used to address a slave, the slave sees an interrupt following the first byte of the extended address.

Figure 12-17. Typical IIC Interrupt Routine



## NOTES:

1. If general call or SIICAEN is enabled, a check must be done to determine whether the received address was a general call address (0x00) or SMBus device default address. If the received address was one of them, then it must be handled by user software.
2. Flow chart1 means [Figure 12-17](#). Typical IIC Interrupt Routine.

Figure 12-18. Typical IIC SMBus Interrupt Routine

## 12.8 SMBALERT#

Another optional signal is an interrupt line for devices that want to trade their ability to master for a pin. SMBALERT# is a wired-AND signal just as the SMBCLK and SMBDAT signals are. SMBALERT# is used in conjunction with the SMBus general call address. Messages invoked with the SMBus are 2 bytes long. (Now there is no ALERT# port in current block)

A slave-only device can signal the host through SMBALERT# that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (ARA). Only the devices that pulled SMBALERT# low acknowledge the alert response address.

The host performs a modified receive byte operation. The 7-bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBALERT# low, the highest priority (lowest address) device win communication rights via standard arbitration during the slave address transfer.

After acknowledging the slave address, the device must disengage its SMBALERT# pulldown. If the host still sees SMBALERT# low when the message transfer is completed, it knows to read the ARA again. A host that does not implement the SMBALERT# signal may periodically access the ARA.

s	Alert Response Address	Rd	A	Device Address	A	P
---	------------------------	----	---	----------------	---	---

**Table 12-16. A 7-bit-Addressable Device Responds to an ARA**

### NOTE

You must put device address on bus by software after response to the alert response address in current block.



## Chapter 13

# Internal Clock Source (S08ICSV4)

### 13.1 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 selected clock source is passed through a reduced bus divider which allows a lower output clock frequency to be derived. The ICS also controls a crystal oscillator (XOSC), which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

The ICS on the MC9S08GW64 series is configured to support only the low range DCO. Clock out fails when the DRS and DRST bits in ICSSC are configured as middle range or high range DCO. The FLL multiplies the reference clock only by 512 or 608 depending on the state of the DMX32 bit.

[Figure 13-1](#) shows the MC9S08GW64 series block diagram with the ICS highlighted.

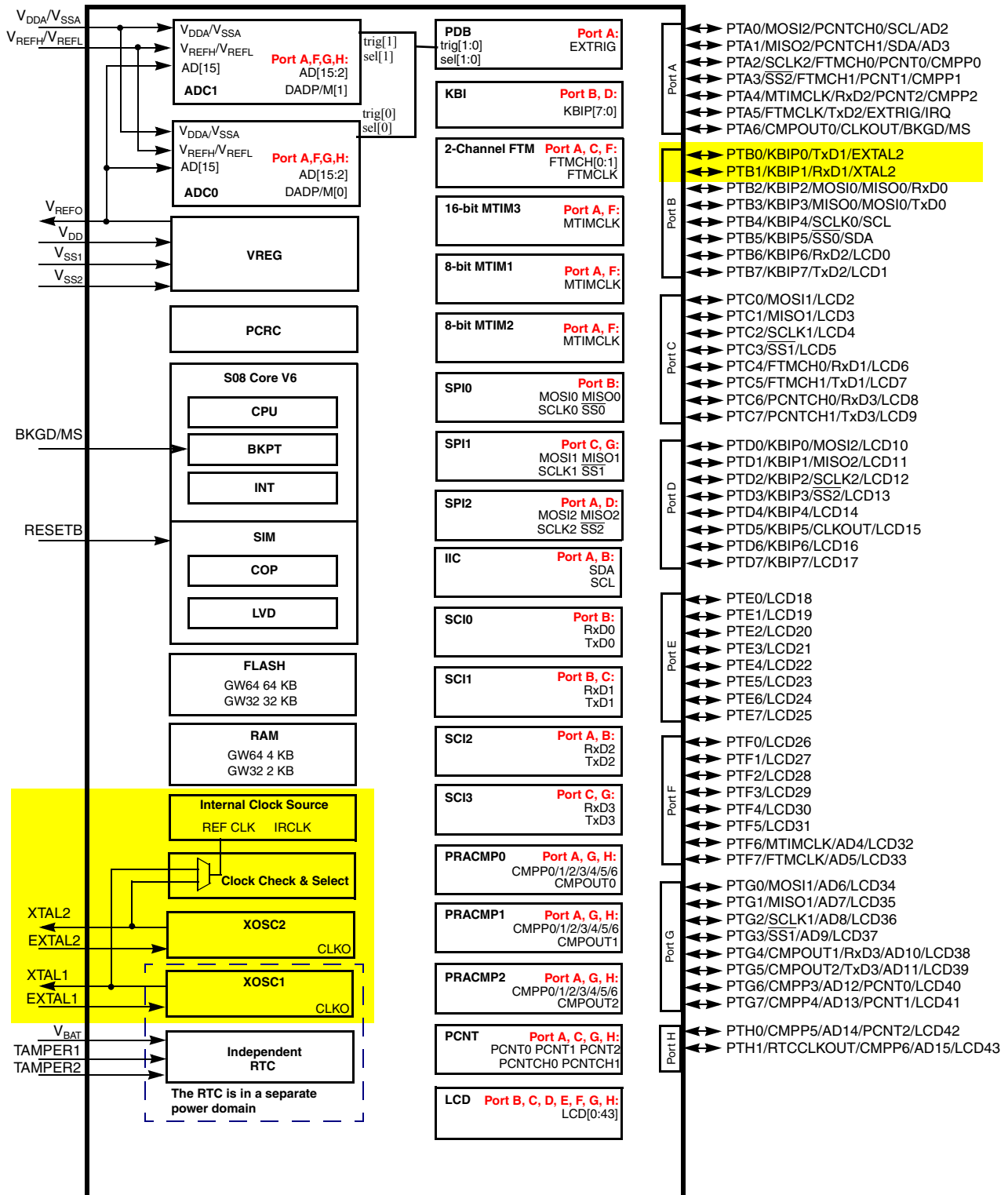


Figure 13-1. MC9S08GW64 Series Block Diagram Highlighting ICS Modules and Pins



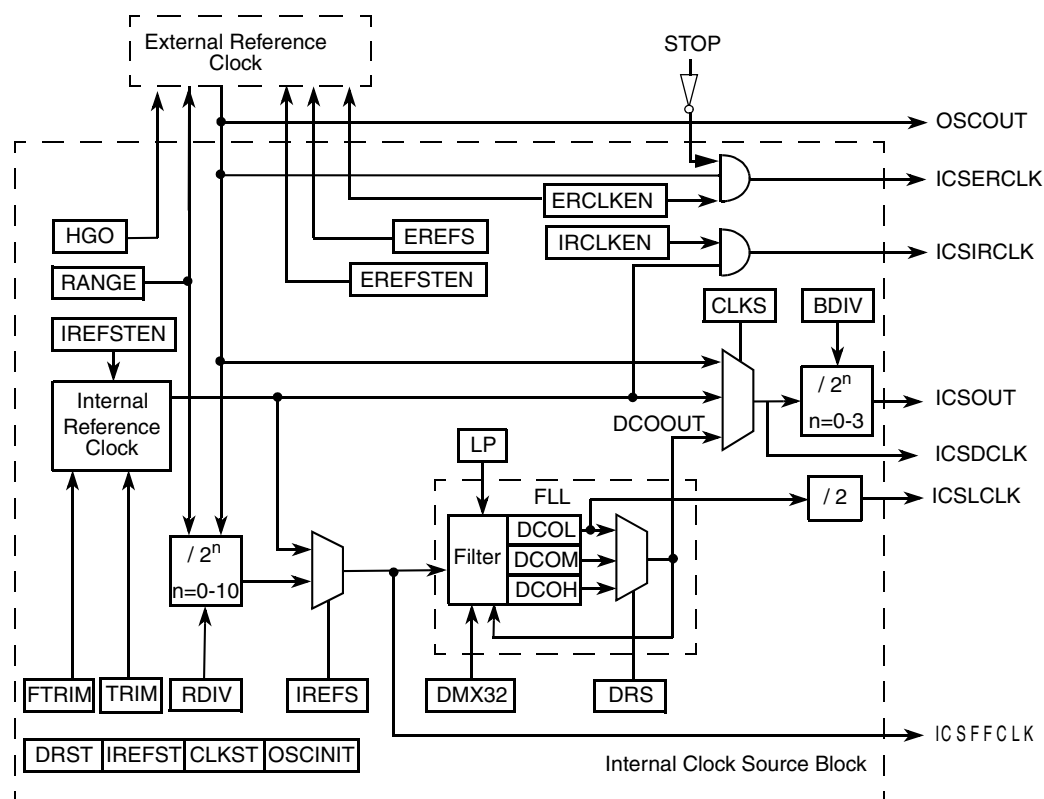
### 13.1.1 Features

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.

### 13.1.2 Block Diagram

Figure 13-2 is the ICS block diagram.



### Figure 13-2. Internal Clock Source (ICS) Block Diagram

### 13.1.3 Modes of Operation

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

### 13.1.3.1 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.

### 13.1.3.2 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.

### 13.1.3.3 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.

### 13.1.3.4 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.

### 13.1.3.5 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.

### 13.1.3.6 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.

### 13.1.3.7 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

Entering STOP mode will not cause the FLL/DCO to reset. The DCO will resume clocking at the same frequency that it was running prior to entering stop mode.

## 13.2 External Signal Description

There are no ICS signals that connect off chip.

## 13.3 Register Definition

Figure 13-1 is a summary of ICS registers.

Table 13-1. 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 13-1. ICS Register Summary (continued)

Name		7	6	5	4	3	2	1	0
ICSSC	R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
	W	DRS							

### 13.3.1 ICS Control Register 1 (ICSC1)

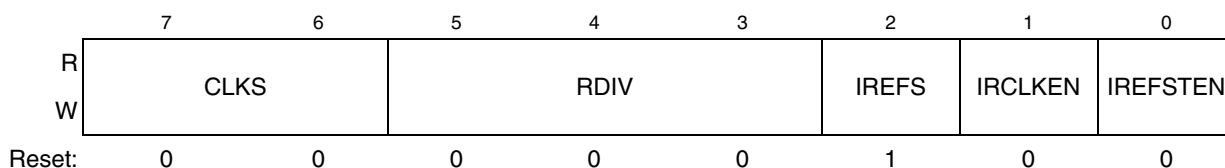


Figure 13-3. ICS Control Register 1 (ICSC1)

Table 13-2. 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 13-3</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 13-3. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
0	1 <sup>1</sup>	32
1	2	64
2	4	128
3	8	256

**Table 13-3. Reference Divide Factor**

<b>RDIV</b>	<b>RANGE=0</b>	<b>RANGE=1</b>
<b>4</b>	16	512
<b>5</b>	32	1024
<b>6</b>	64	Reserved
<b>7</b>	128	Reserved

<sup>1</sup> Reset default

### 13.3.2 ICS Control Register 2 (ICSC2)

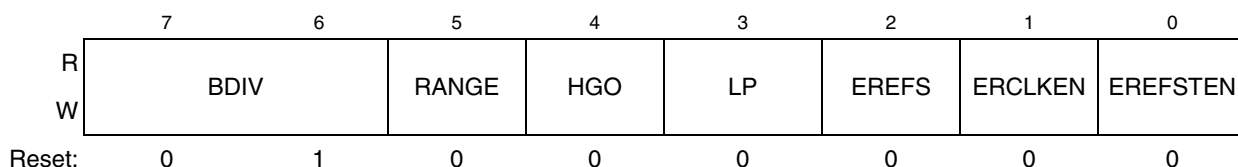
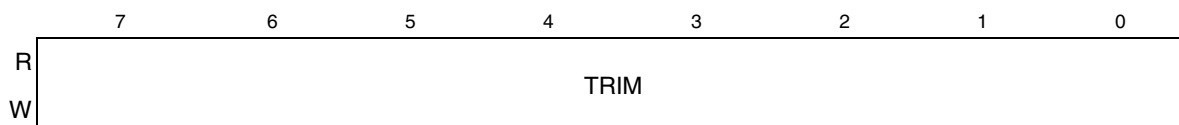


Figure 13-4. ICS Control Register 2 (ICSC2)

Table 13-4. 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 EREFS	<b>External Reference Select</b> — The EREFS bit selects the source for the external reference clock. 1 Oscillator requested. 0 External Clock Source requested.
1 ERCLKEN	<b>External Reference Enable</b> — The ERCLKEN bit enables the external reference clock for use as IC SERCLK. 1 IC SERCLK active. 0 IC SERCLK inactive.
0 EREFSTEN	<b>External Reference Stop Enable</b> — The EREFSTEN bit controls whether or not the external reference clock 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.

### 13.3.3 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 13-5. ICS Trim Register (ICSTRM)

Table 13-5. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	<p><b>ICS Trim Setting</b> — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (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.</p> <p>An additional fine trim bit is available in ICSSC as the FTRIM bit.</p>

### 13.3.4 ICS Status and Control (ICSSC)

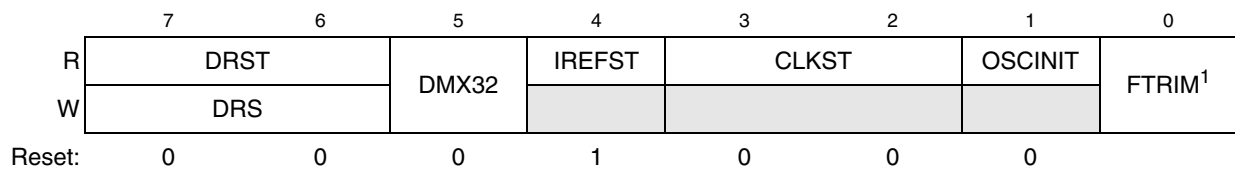


Figure 13-6. ICS Status and Control Register (ICSSC)

- <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 13-6. ICS Status and Control Register Field Descriptions

Field	Description
7-6 DRST DRS	<p><b>DCO Range Status</b> — The DRST read field indicates the current frequency range for the FLL output, DCOOUT. See <a href="#">Table 13-7</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.</p> <p><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.</p> <p>00 Low range. 01 Mid range. 10 High range. 11 Reserved.</p>
5 DMX32	<p><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 13-7</a>.</p> <p>0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.</p>
4 IREFST	<p><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.</p> <p>0 Source of reference clock is external clock. 1 Source of reference clock is internal clock.</p>

Table 13-6. 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 13-7. DCO frequency range<sup>1</sup>

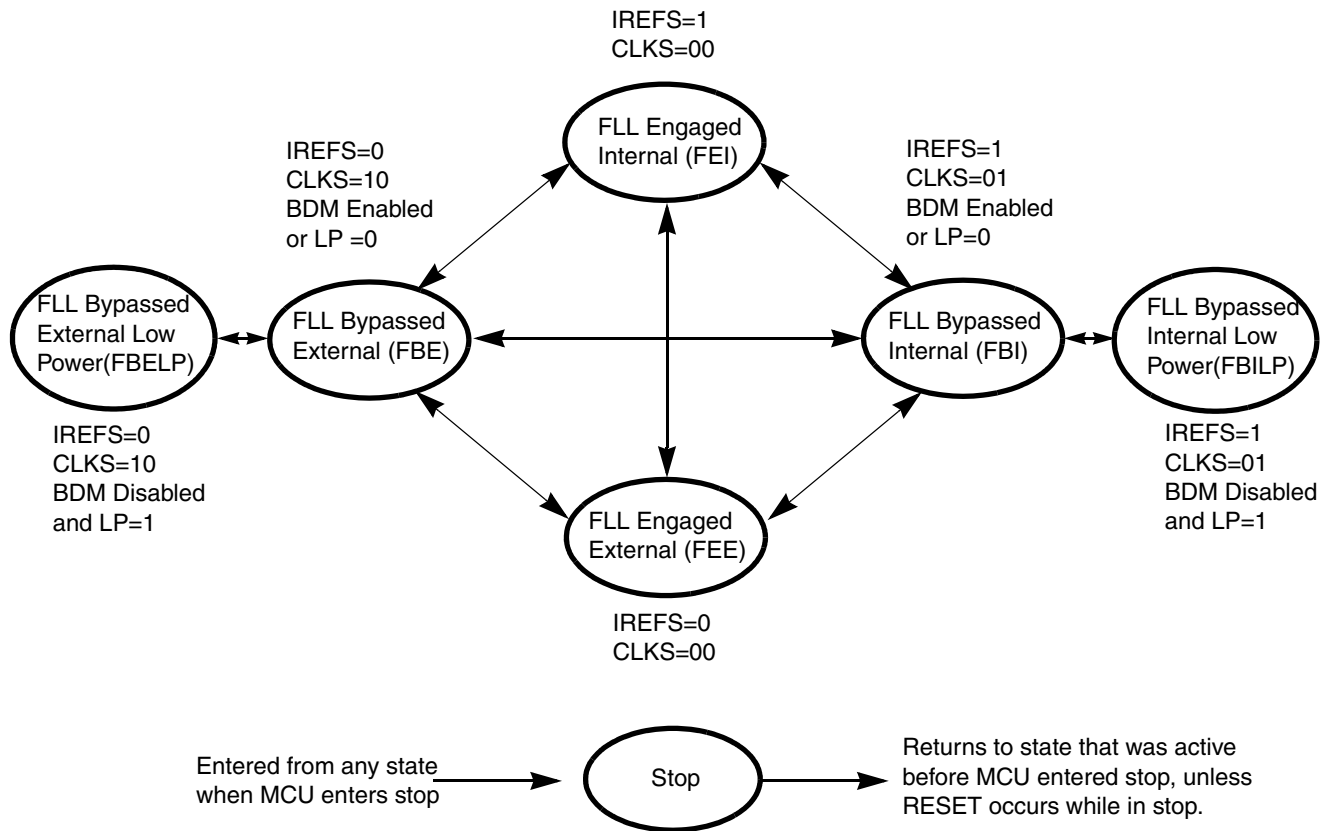
DRS	DMX32	Reference range	FLL factor	DCO range
00	0	31.25 - 39.0625 kHz	512	16 - 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	0	31.25 - 39.0625 kHz	1024	32 - 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	0	31.25 - 39.0625 kHz	1536	48 - 60 MHz
	1	32.768 kHz	1824	59.77 MHz
11	Reserved			

<sup>1</sup> The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.



## 13.4 Functional Description

### 13.4.1 Operational Modes



**Figure 13-7. 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.

#### 13.4.1.1 FLL Engaged Internal (FEI)

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

- CLKS bits are written to 00.
- IREFS bit is written to 1.

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.

### 13.4.1.2 FLL Engaged External (FEE)

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

- CLKS bits are written to 00.
- IREFS bit is written to 0.
- RDIV bits are written to divide 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.

### 13.4.1.3 FLL Bypassed Internal (FBI)

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

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

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop 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.

### 13.4.1.4 FLL Bypassed Internal Low Power (FBILP)

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

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

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

### 13.4.1.5 FLL Bypassed External (FBE)

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

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- 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.

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

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

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

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

#### 13.4.1.7 Stop

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

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

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

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

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

### 13.4.2 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.

### 13.4.3 Bus Frequency Divider

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

### 13.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. 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.

### 13.4.5 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.

### 13.4.6 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.

### 13.4.7 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 ICSECLK, 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 ICSECLK. 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.

### 13.4.8 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. When ICSFFCLK is valid, ICS output signal (ICSFFE) gets asserted high {ics\_ffe.asm}. Because of this requirement, in bypass modes the ICSFFCLK is valid only in bypass external modes (FBE and FBELP) for the following combinations of BDIV, RDIV and RANGE values:

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

### 13.4.9 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 14

# 8-Bit Modulo Timer (S08MTIMV1)

### 14.1 Introduction

The MTIM1 and MTIM2 are simple 8-bit timers with several software selectable clock sources and a programmable interrupt.

The MTIM1 and MTIM2 clock source seletable as bus clock, fixed system clock or an TCLK which is fed from an external clock pin MTIMCLK.

#### 14.1.1 MTIM Clock Gating

The bus clock to the MTIM1, MTIM2 modules can be gated on and off using the MTIM1, MTIM2 bits in SCGC4 register. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the MTIM1, MTIM2 bits can be cleared to disable the clock to this module. See [Section 5.7](#), “Peripheral Clock Gating,” for details.

In this chapter, both MTIM1 and MTIM2 are referred as MTIM. Refer to [Table 4-3](#) for the addresses of MTIM1 and MTIM2 registers.

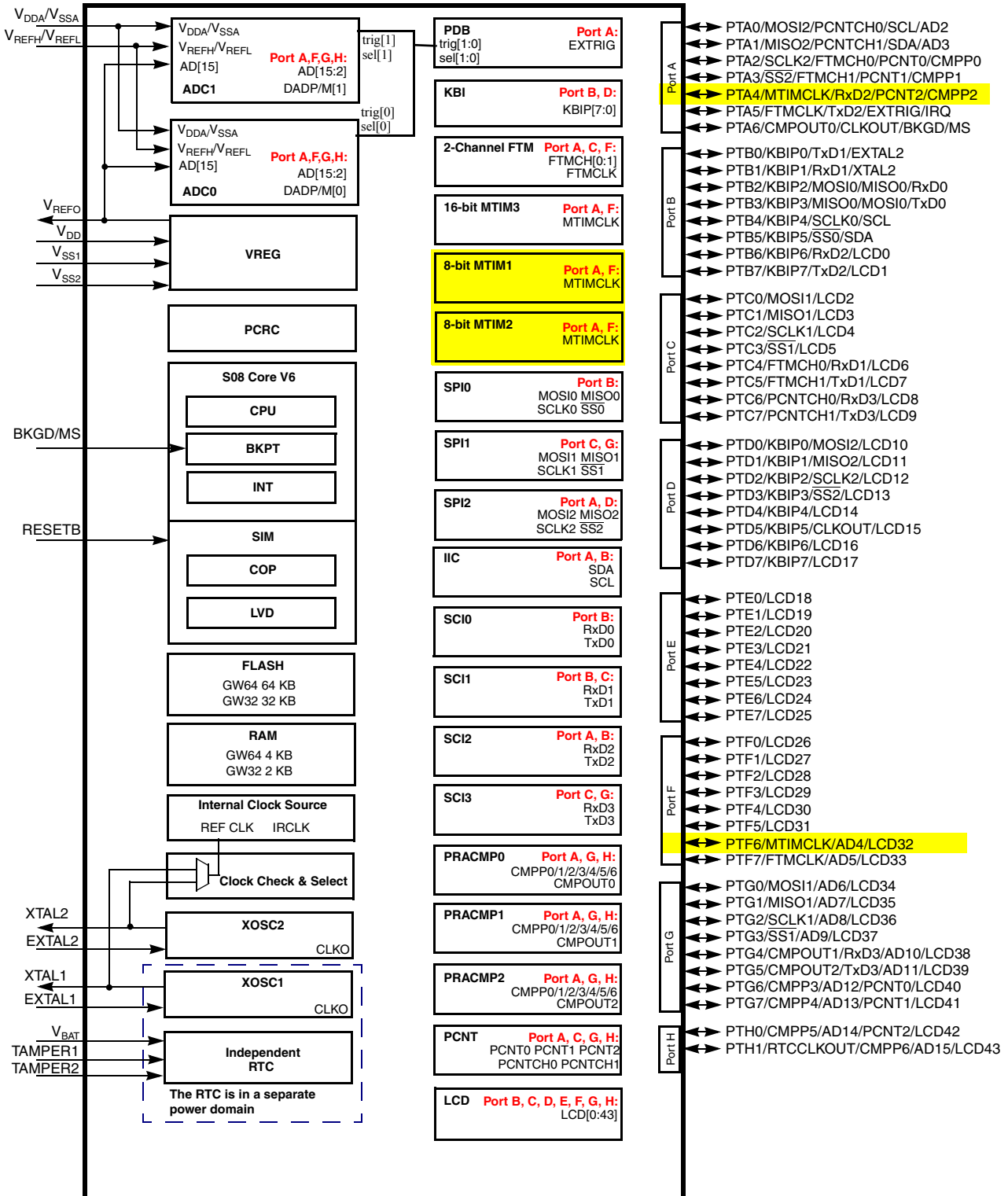


Figure 14-1. MC9S08GW64 Series Block Diagram Highlighting 8-Bit MTIM Modules and Pins



## 14.1.2 Features

Timer system features include:

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

## 14.1.3 Modes of Operation

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

### 14.1.3.1 MTIM in Wait Mode

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

### 14.1.3.2 MTIM in Stop Modes

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

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

### 14.1.3.3 MTIM in Active Background Mode

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

14.1.4 Block Diagram

The block diagram for the modulo timer module is shown [Figure 14-2](#).

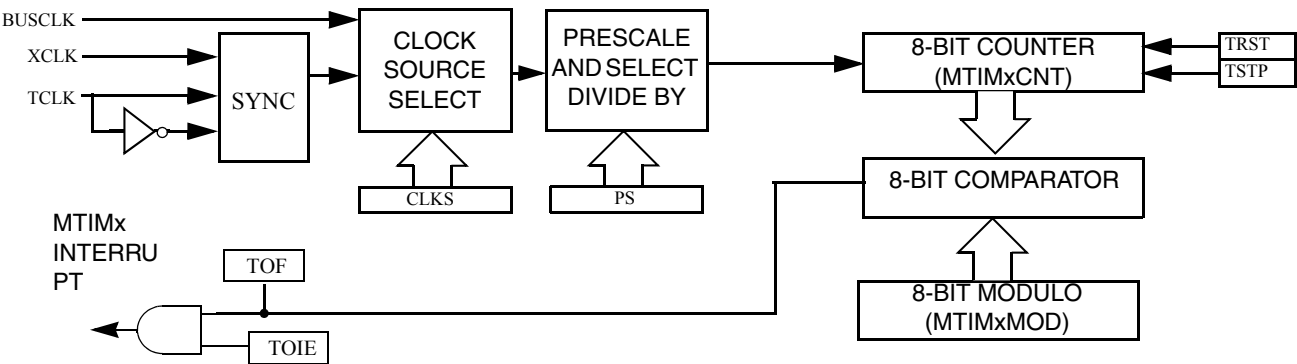


Figure 14-2. Modulo Timer (MTIM) Block Diagram

14.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in [Table 14-1](#).

Table 14-1.

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

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. Therefore, the TCLK signal must be limited to one-fourth of the bus frequency. The TCLK pin can be muxed with a general-purpose port pin. See the [Pins and Connections](#) chapter for the pin location and priority of this function.

14.3 Register Definition

[Figure 14-3](#) is a summary of MTIM registers.

**Figure 14-3. MTIMx Register Summary**

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

Each MTIM includes four registers:

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

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

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

### 14.3.1 MTIMx Status and Control Register (MTIMxSC)

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

	7	6	5	4	3	2	1	0
R			0		0	0	0	0
W	TOF	TOIE	TRST	TSTP				
Reset:	0	0	0	1	0	0	0	0

Figure 14-4. MTIMx Status and Control Register

Table 14-2. MTIMxSC Field Descriptions

Field	Description
7 TOF	<b>MTIM Overflow Flag</b> — This bit is set when the MTIM counter register overflows to \$00 after reaching the value in the MTIM modulo register. Clear TOF by reading the MTIMxSC register while TOF is set, then writing a 0 to TOF. TOF is also cleared when TRST is written to a 1 or when any value is written to the MTIMxMOD register. 0 MTIM counter has not reached the overflow value in the MTIM modulo register. 1 MTIM counter has reached the overflow value in the MTIM modulo register.
6 TOIE	<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.

### 14.3.2 MTIMx Clock Configuration Register (MTIMxCLK)

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

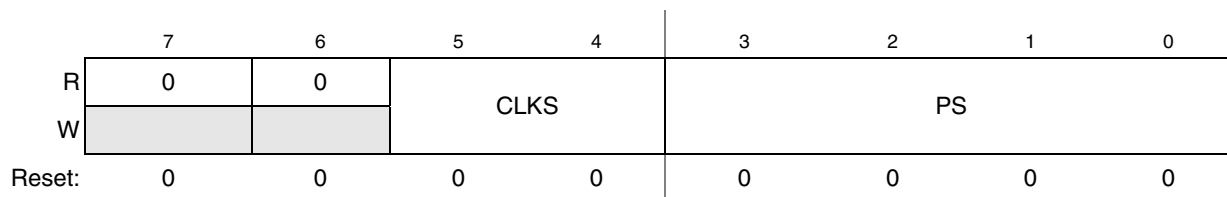


Figure 14-5. MTIMx Clock Configuration Register

Table 14-3. MTIMxCLK Field Descriptions

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 ÷ 1 0001 Encoding 1. MTIM clock source ÷ 2 0010 Encoding 2. MTIM clock source ÷ 4 0011 Encoding 3. MTIM clock source ÷ 8 0100 Encoding 4. MTIM clock source ÷ 16 0101 Encoding 5. MTIM clock source ÷ 32 0110 Encoding 6. MTIM clock source ÷ 64 0111 Encoding 7. MTIM clock source ÷ 128 1000 Encoding 8. MTIM clock source ÷ 256 All other encodings default to MTIM clock source ÷ 256.

14.3.3 MTIMx Counter Register (MTIMxCNT)

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

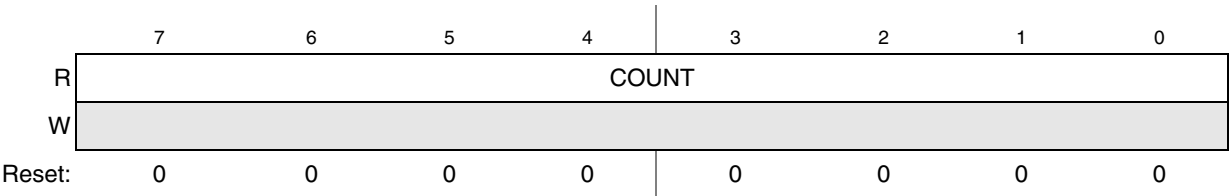


Figure 14-6. MTIMx Counter Register

Table 14-4. MTIMxCNT Field Descriptions

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

14.3.4 MTIMx Modulo Register (MTIMxMOD)

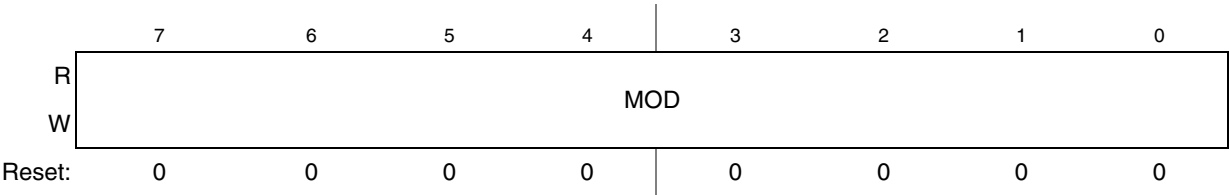


Figure 14-7. MTIMx Modulo Register

Table 14-5. MTIMxMOD 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 MTIMxMOD resets the COUNT to \$00 and clears TOF. Reset sets the modulo to \$00.

## 14.4 Functional Description

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

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

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

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

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

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

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

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

### 14.4.1 MTIM Operation Example

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

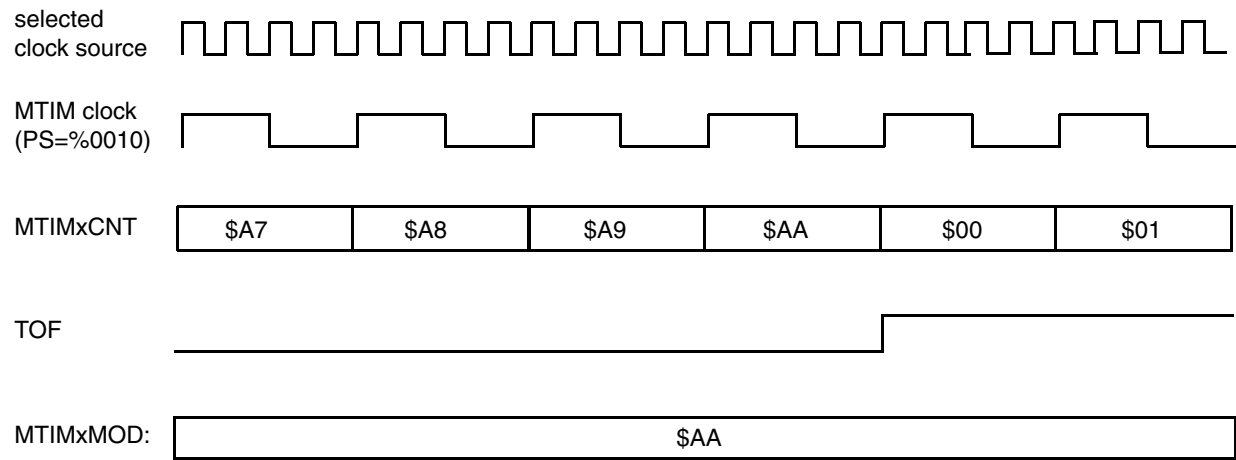


Figure 14-8. MTIM counter overflow example

In the example of [Figure 14-8](#), the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMxMOD register is set to \$AA. When the counter, MTIMxCNT, reaches the modulo value of \$AA, the counter overflows to \$00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from \$AA to \$00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.



## Chapter 15

# 16-Bit Modulo Timer (S08MTIM16V1)

### 15.1 Introduction

The MTIM3 is a 16-bit timer with several software selectable clock sources and a programmable interrupt.

The MTIM3 clock source is selectable from bus clock, fixed system clock or TCLK which is an external clock source and fed from MTIMCLK pin.

#### 15.1.1 MTIM3 Clock Gating

The bus clock to the MTIM3 module can be gated on and off using the MTIM3 bit in SCGC4. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the MTIM3 bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

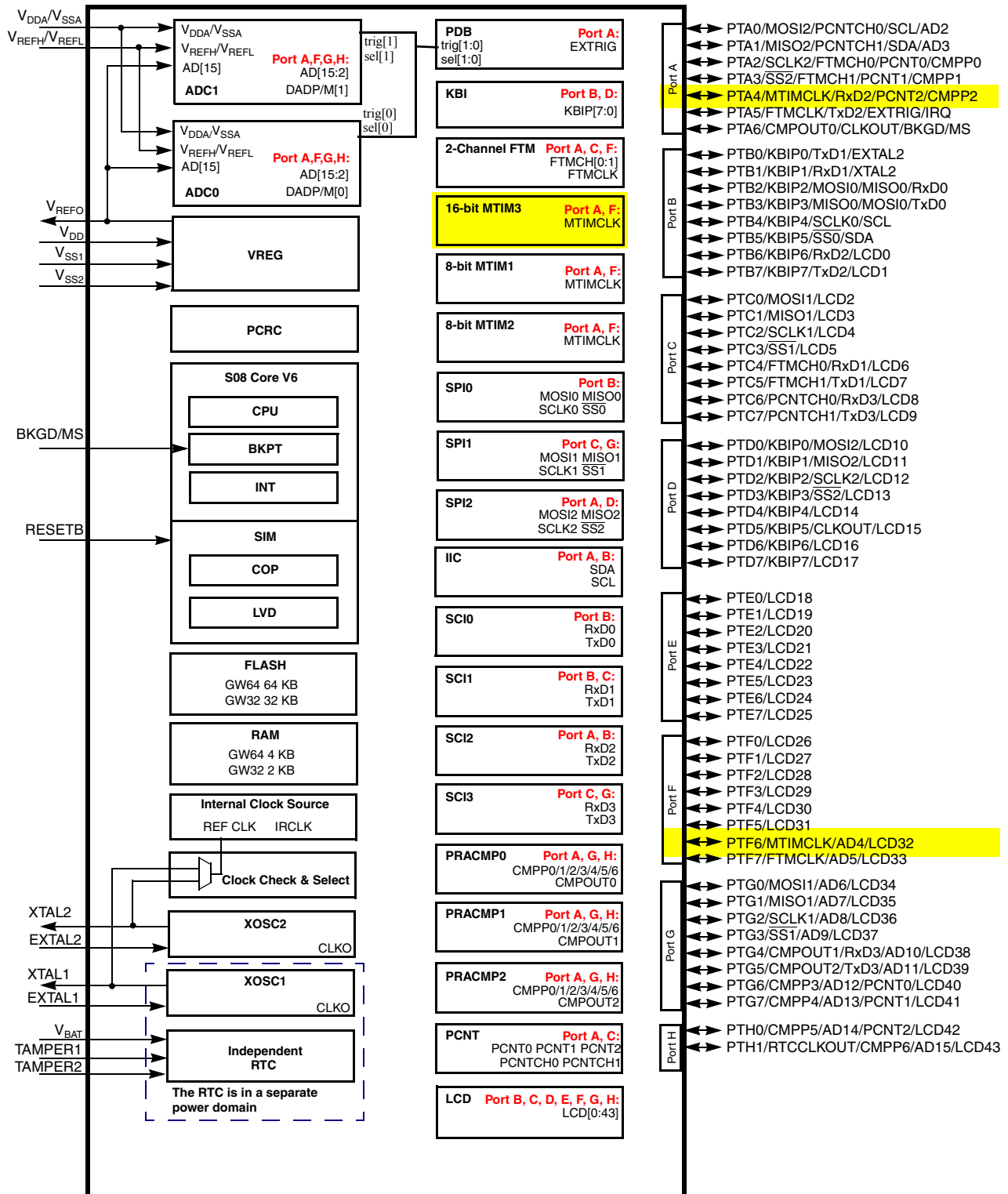


Figure 15-1. MC9S08GW64 Series Block Diagram Highlighting 16-Bit MTIM Module and Pins

## 15.2 Features

Timer system features include:

- 16-bit up-counter
  - Free-running or 16-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

### 15.2.1 Block Diagram

The block diagram for the modulo timer module is shown [Figure 15-2](#).

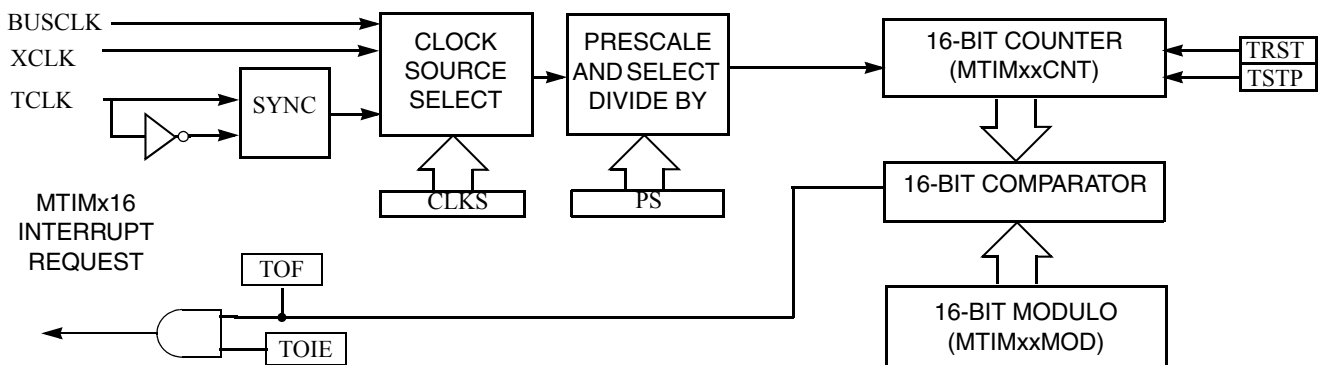


Figure 15-2. Modulo Timer (S08MTIM16) Block Diagram

### 15.2.2 Modes of Operation

This section defines MTIM16 operation in stop, wait, and background debug modes.

#### 15.2.2.1 MTIM16 in Wait Mode

The MTIM16 continues to run in wait mode if enabled prior to the execution of the WAIT instruction. The timer overflow interrupt brings the MCU out of wait mode if it is enabled. For lowest possible current

consumption, the MTIM16 should be stopped by software if it is not needed as an interrupt source during wait mode.

### 15.2.2.2 MTIM16 in Stop Modes

The MTIM16 is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM16 cannot be used as a wake up source from stop mode.

Upon waking from stop2 mode, the MTIM16 will enter its reset state. If stop3 is exited with a reset, the MTIM16 will enter its reset state. If stop3 is exited with an interrupt, the MTIM16 continues from the state it was in stop3. If the counter was active upon entering stop3, the count will resume from the current value.

### 15.2.2.3 MTIM16 in Active Background Mode

The MTIM16 stops all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM16 reset did not occur (TRST written to a 1).

## 15.3 External Signal Description

### 15.3.1 TCLK — External Clock Source Input into MTIM16

The MTIM16 includes one external signal, TCLK, used to input an external clock when selected as the MTIM16 clock source. The signal properties of TCLK are shown in [Table 15-1](#).

**Table 15-1. TCLK Properties**

Signal	Function	I/O
TCLK	External clock source input into MTIM16	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. As a result, 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 [Chapter 2, “Pins and Connections”](#) for the pin location and priority of this function.

## 15.4 Register Definition

Each MTIM16 includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- A 16-bit counter register

A 16-bit modulo register. Refer to the direct-page register summary in the Memory chapter for the absolute address assignments for all MTIM16 registers. This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM16, so register names include placeholder characters to identify the correct MTIM16.

### 15.4.1 MTIMx16 Status and Control Register (MTIMxSC)

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

	7	6	5	4	3	2	1	0
R			0		0	0	0	0
W	TOF	TOIE	TRST	TSTP				
Reset:	0	0	0	1	0	0	0	0

Figure 15-3. MTIM16 Status and Control Register (MTIMxSC)

Table 15-2. MTIMxSC Field Descriptions

Field	Description
7 TOF	<b>MTIM16 Overflow Flag</b> — This bit is set when the MTIM16 counter register overflows to 0x0000 after reaching the value in the MTIM16 modulo register. Clear TOF by reading the MTIMxSC register while TOF is set, then writing a 0 to TOF. Writing a 1 has no effect. TOF is also cleared when TRST is written to a 1. 0 MTIM16 counter has not reached the overflow value in the MTIM16 modulo register. 1 MTIM16 counter has reached the overflow value in the MTIM16 modulo register.
6 TOIE	<b>MTIM16 Overflow Interrupt Enable</b> — This read/write bit enables MTIM16 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>MTIM16 Counter Reset</b> — When an 1 is written to this write-only bit, the MTIM16 counter register resets to 0x0000 and TOF is cleared. Writing an 1 to this bit also makes the modulo value to take effect at once. Reading this bit always returns 0. 0 No effect. MTIM16 counter remains in its current state. 1 MTIM16 counter is reset to 0x0000.
4 TSTP	<b>MTIM16 Counter Stop</b> — When set, this read/write bit stops the MTIM16 counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM16 from counting. 0 MTIM16 counter is active. 1 MTIM16 counter is stopped.
3:0	Unused register bits, always read 0.

### 15.4.2 MTIM16 Clock Configuration Register (MTIMxCLK)

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

	7	6	5	4	3	2	1	0
R	0	0						
W				CLKS			PS	
Reset:	0	0	0	0	0	0	0	0

Figure 15-4. MTIM16 Clock Configuration Register (MTIMxCLK)

Table 15-3. MTIMxCLK Field Descriptions

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 MTIM16 prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 00. 00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge
3:0 PS	<b>Clock Source Prescaler</b> — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM16 clock source ÷ 1 0001 Encoding 1. MTIM 16clock source ÷ 2 0010 Encoding 2. MTIM16 clock source ÷ 4 0011 Encoding 3. MTIM16 clock source ÷ 8 0100 Encoding 4. MTIM16 clock source ÷ 16 0101 Encoding 5. MTIM16 clock source ÷ 32 0110 Encoding 6. MTIM16 clock source ÷ 64 0111 Encoding 7. MTIM16 clock source ÷ 128 1000 Encoding 8. MTIM16 clock source ÷ 256 All other encodings default to MTIM16 clock source ÷ 256.

### 15.4.3 MTIM16 Counter Register High/Low (MTIMxCNTH:L)

MTIMxCNTH is the read-only value of the high byte of current MTIM16 16-bit counter.

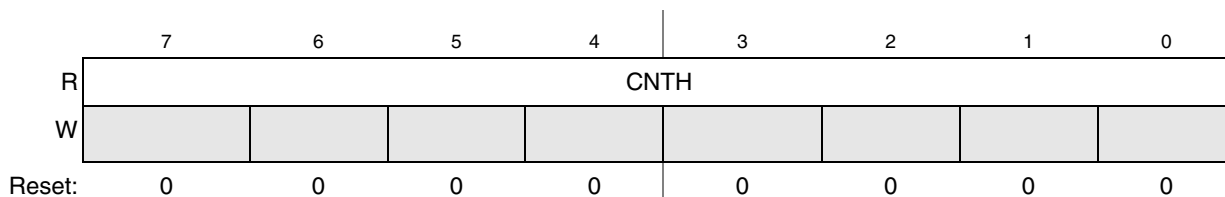


Figure 15-5. MTIMx16 Counter Register High (MTIMxCNTH)

Table 15-4. MTIMxCNTH Field Descriptions

Field	Description
7:0 CNTH	<b>MTIM16 Count (High Byte)</b> — These eight read-only bits contain the current high byte value of the 16-bit counter. Writing has no effect to this register. Reset clears the register to 0x00.

MTIMxCNTL is the read-only value of the low byte of current MTIM16 16-bit counter.

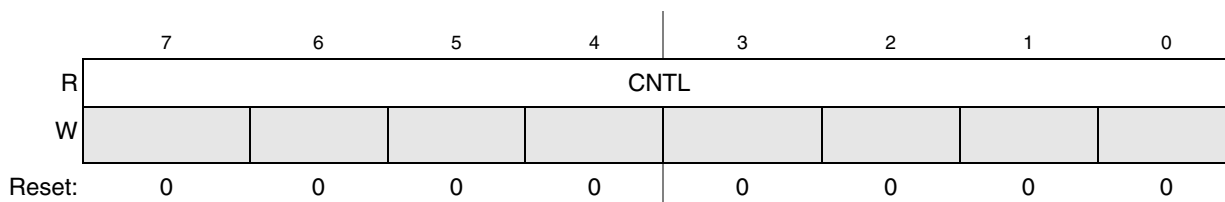


Figure 15-6. MTIM16 Counter Register Low (MTIMxCNTL)

Table 15-5. MTIMxCNTL Field Descriptions

Field	Description
7:0 CNTL	<b>MTIM16 Count (Low Byte)</b> — These eight read-only bits contain the current low byte value of the 16-bit counter. Writing has no effect to this register. Reset clears the register to 0x00.

When either MTIMxCNTH or MTIMxCNTL is read, the content of the two registers is latched into a buffer where they remain latched until the other register is read. This allows the coherent 16-bit to be read in both big-endian and little-endian compile environments and ensures the 16-bit counter is unaffected by the read operation. The coherency mechanism is automatically restarted by an MCU reset or setting of TRST bit of MTIMxSC register (whether BDM mode is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the counter 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, the appropriate value from the other half of the 16-bit value will be read after returning to normal execution. The value read from the MTIMxCNTH and MTIMxCNTL registers in BDM mode is the value of these registers and not the value of their read buffer.

#### 15.4.4 MTIM16 Modulo Register High/Low (MTIMxMODH/MTIMxMODL)

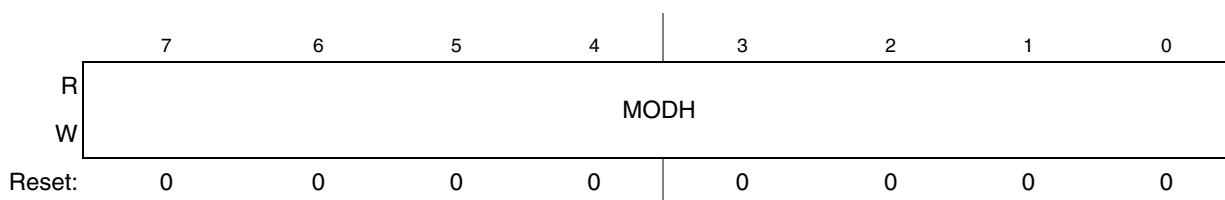


Figure 15-7. MTIM16 Modulo Register High (MTIMxMODH)

Table 15-6. MTIMxMODH Field Descriptions

Field	Description
7:0 MODH	<b>MTIM16 Modulo (High Byte)</b> — These eight read/write bits contain the modulo high byte value used to reset the counter and set TOF. Reset sets the register to 0x00.

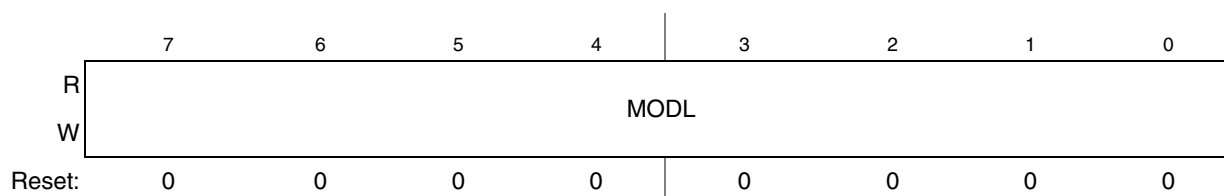


Figure 15-8. MTIM16 Modulo Register Low (MTIMxMODL)

Table 15-7. MTIMxMODL Field Descriptions

Field	Description
7:0 MODL	<b>MTIM16 Modulo (Low Byte)</b> — These eight read/write bits contain the modulo low byte value used to reset the counter and set TOF. Reset sets the register to 0x00.

A value of 0x0000 in MTIMxMODH:L puts the MTIM16 in free-running mode. Writing to either MTIMxMODH or MTIMxMODL latches the value into a buffer and the registers are updated with the value of their write buffer after the second byte writing, the updated MTIMxMODH:L will take effect in the next MTIM16 counter cycle except for the first writing of modulo after a chip reset or in BDM mode. But after a software reset, the MTIMxMODH:L takes effect at once even if it didn't take effect before the reset. On the first writing of MTIMxMODH:L after chip reset, the counter is reset and the modulo takes effect immediately. The latching mechanism may be manually reset by setting the TRST bit of MTIMxSC register (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any writing to the modulo registers bypasses the buffer latches and writes directly to the modulo register while BDM is active, and also the counter is cleared at the same time. The reading of MTIMxMODH:L returns the modulo value which is taking effect whenever in normal run mode or in BDM mode.

## 15.5 Functional Description

The MTIM16 is composed of a main 16-bit up-counter with 16-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM16 counter (MTIMxCNTH:L) has three modes of operation: stopped, free-running, and modulo. The counter is stopped out of reset. If the counter starts without writing a new value to the modulo registers, it will be in free-running mode. The counter is in modulo mode when a value other than 0x0000 is in the modulo registers.

After an MCU reset, the counter stops and resets to 0x0000, and the modulo is also reset to 0x0000. The bus clock functions as the default clock source and the prescale value is divided by 1. To start the MTIM16 in free-running mode, write to the MTIM16 status and control register (MTIMxSC) and clear the MTIM16 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



MTIM16 clock select bits (CLKS1:CLKS0) in MTIMxCLK are used to select the desired clock source. If the counter is active (TSTP = 0) when a new clock source is selected, the counter continues 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 MTIMxCLK select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter continues counting from the previous value using the new prescaler value.

The MTIM16 modulo register (MTIMxMODH:L) allows the overflow compare value to be set to any value from 0x0001 to 0xFFFF. Reset clears the modulo value to 0x0000, which results in a free running counter.

When the counter is active (TSTP = 0), it increases at the selected rate until the count matches the modulo value. When these values match, the counter overflows to 0x0000 and continues counting. The MTIM16 overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to 0x0000.

Clearing TOF is a two-step process. The first step is to read the MTIMxSC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF stays 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.

The MTIM16 allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM16 overflow interrupt, set the MTIM16 overflow interrupt enable bit (TOIE) in MTIMxSC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

### 15.5.1 MTIM16 Operation Example

This section shows an example of the MTIM16 operation as the counter reaches a matching value from the modulo register.

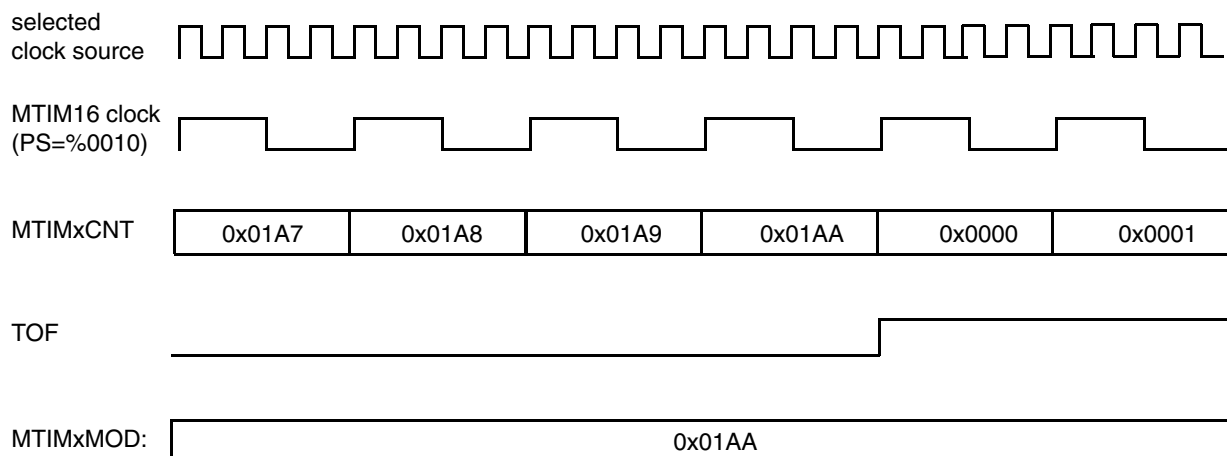


Figure 15-9. MTIM16 Counter Overflow Example

In the example of [Figure 15-9](#), the selected clock source could be any of the four possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMxMODH:L register is set to 0x01AA. When the counter, MTIMxCNTH:L, reaches the modulo value of 0x01AA, the counter overflows to 0x0000 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from 0x01AA to 0x0000. An MTIM16 overflow interrupt is generated when TOF is set, if TOIE = 1.

# Chapter 16

## Independent Real Time Clock (IRTCV2)

### 16.1 Introduction

The independent real time clock provides the functionality of a basic RTC like time keeping and calendaring and additionally provides protection against tampering, spurious memory/register updates and battery operation. It can additionally compensate the 1 Hz clock against variations in 32 kHz clock in oscillator due to crystal or temperature. A standby RAM is provided if the CPU wants to store any data that has to be retained in battery operation mode.

#### NOTE

The IRTC configuration data register (IRTC\_CFG\_DATA) bits [7:1] are reserved in MC9S08GW64 series, and must be written as 0's.

MC9S08GW64 series have two tamper pins TAMPER1 and TAMPER2. TAMPER0 pin is dedicatedly used for Battery Removal Tamper and not exposed on any pins. Further reference to a different number of tamper pins must be disregarded.

The IRTC features a protection mechanism to protect against spurious writes into the IRTC registers by any run-away code. On power-on reset a 15 s window is allowed for the CPU to configure the IRTC after which the registers are locked.

The IRTC tamper interrupt is enabled after reset (IRTC\_IER register, bit TMPR = 1). A POR generates a tamper interrupt request by setting the bit TMPR in the IRTC\_ISR register.

Year & Month Alarm (IRTC\_ALM\_YRMON), Days Alarm (IRTC\_ALM\_DAYS), Hours and Minutes Alarm (IRTC\_ALM\_HM), Seconds Alarm (IRTC\_ALM\_SEC), Status (IRTC\_STATUS), Interrupt Status (IRTC\_ISR) bits [15:1], Interrupt Enable (IRTC\_IER) bits [15:1] and Configuration Data Registers (IRTC\_CFG\_DATA) are reset by IRTC soft reset or IRTC power-on reset. All other registers are reset by IRTC power-on reset. IRTC power-on reset occurs when  $V_{DD}$  or  $V_{bat}$  applies a voltage above the POR rearm voltage.

#### 16.1.1 IRTC Power Supply Source

The IRTC power supply source depends on the MCU operation mode and the LVD configuration.

Refer to the following for the IRTC power supply with the MCU in run, wait modes<sup>1</sup>:

- If ( $V_{DD} > V_{LVDH}$ ), the IRTC is powered by  $V_{DD}$  pin.
- If ( $V_{DD} < V_{LVDH}$ ) and ( $V_{DD} > V_{LVDL}$ ), the IRTC is powered by  $V_{DD}$  and  $V_{BAT}$  pins.
- If ( $V_{DD} < V_{LVDL}$ ), the IRTC is powered by the  $V_{BAT}$  pin.

Refer to the following for the IRTC power supply with the MCU in LPRun, LPWait, stop3 or stop2<sup>2</sup>:

- The IRTC always operates from  $V_{BAT}$ .

#### NOTE

When IRTC is powered by  $V_{BAT}$ , it can not be configured or read.

When powered by  $V_{BAT}$ , the 512 Hz clock is not available for use by other modules (e.g. PCounter) as the internal prescaler is gated to save on battery power.

### 16.1.2 IRTC Clock Gating

The bus clock to the IRTC module can be gated on and off using the IRTC bit in SCGC5. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the IRTC bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

<sup>1</sup>. The IRTC power supply in run and wait modes is independent of the LVD configuration.

<sup>2</sup>. In LPRun, LPWait, stop3 and stop2, the LVD is disabled.

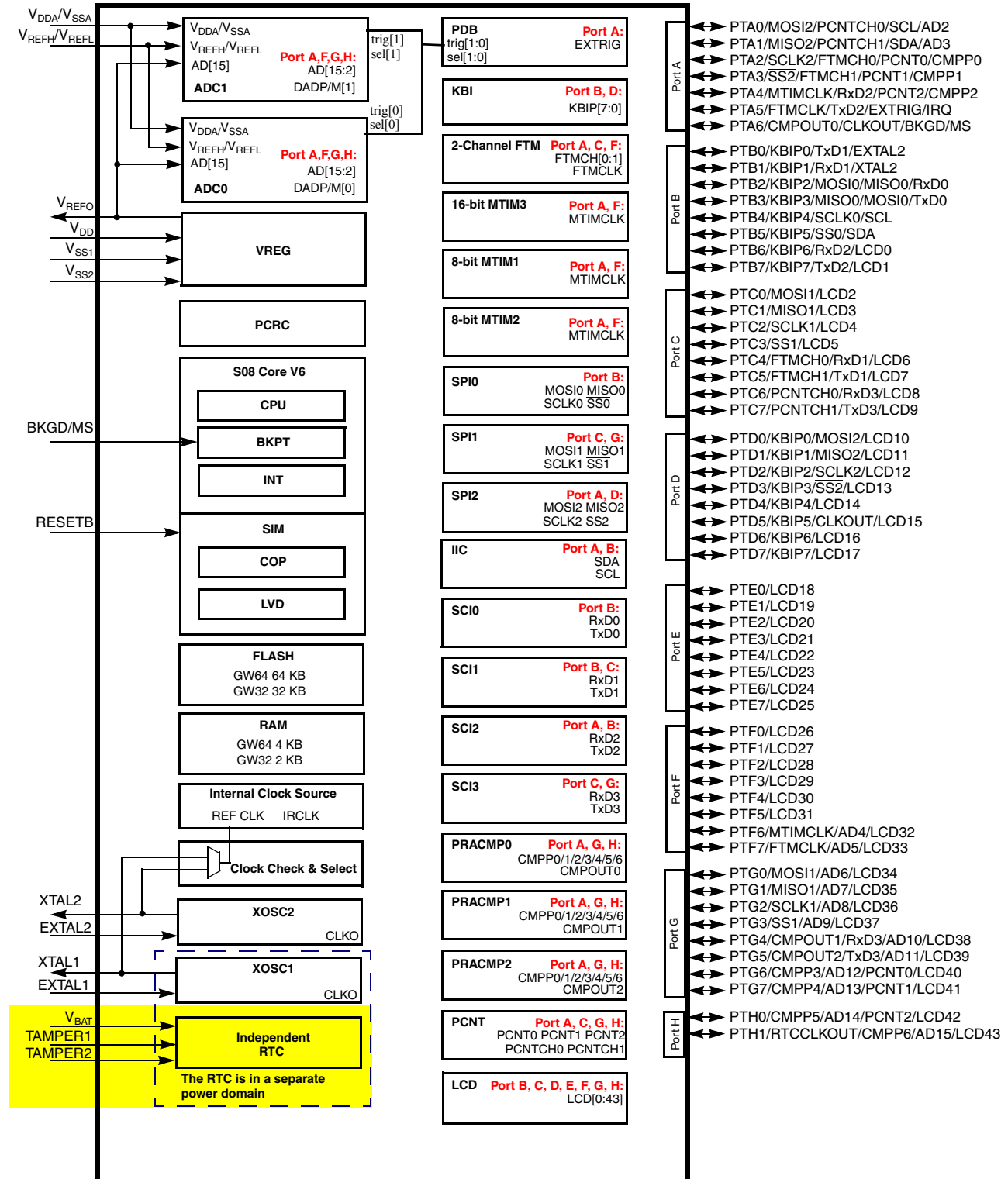


Figure 16-1. MC9S08GW64 Series Block Diagram Highlighting IRTC Module and Pins

## 16.2 Features

The Independent Real Time Clock (V2.0) or IRTC is a low power module that provides time keeping functions, calendaring functions and additionally provides protection against tampering, spurious memory/register updates and battery operation. It can additionally compensate the 1 Hz clock against variations in 32 kHz clock in oscillator due to crystal or temperature. A Standby RAM is provided if the CPU wants to store any data that has to be retained when in battery operation mode.

The IRTC supports the following features:

- Designed for low power
  - Time & date counters are rippled with respect to each other to prevent simultaneous toggling
- Basic clock functions
  - Days, hours, minutes and seconds
- Calendaring – month, year and day of the week
- Auto adjustment for day light saving with user defined parameters
- Automatic month and leap year adjustment
- IRTC utilizes ‘local time’ which implicitly contains the time zone offset
- Programmable alarm with interrupt.
- Seven periodic interrupts
- Minute countdown timer with minute resolution
- Hardware compensation to compensate 1 Hz clock (to the counters) against frequency variations in oscillator clock due to temperature or crystal characteristics. Correction factor calculated by firmware. (Programmable correction factor).
- CPU Register Programming Interface with protection against run-away code.
- Reset to the IRTC block is generated only when both battery supply and CPU power are removed and either is powered up.
- Battery operation (standby mode) ensures seamless IRTC operation when CPU power is removed
- Option to output the buffered 32.768 kHz clock or the compensated 1 Hz clock.
- Standby RAM of 32 Bytes depth
- Enhanced tamper detection
  - Tamper detection to detect illegal access to the system. Two external tamper detect sources supported. One internal tamper event on battery removal is also detected.
  - Tamper sources can be individually enabled/disabled and polarity can be controlled
  - Single time stamp stored on each tamper event. Latest tamper time is stored in case of multiple tampers
  - Tamper input widths can be configured to be from the minimum 64  $\mu$ s to 125 ms separately.
- Standby exit indication to allow the CPU to execute code from RAM rather than fetching the code again.

## 16.3 Modes of Operation

The IRTC block provides basic time keeping functions by second, minute or hour counters and calendaring functions via date, day-of-week, or month and year counters; with automatic adjustment for leap year and day light saving. Reading these counters indicates the current date and time and writing to these registers sets the date and time given by the user.

The alarm is set for specific hour, minute and second. When the time counters match the alarm hour, minute and second setting, the alarm flag is set and an interrupt to CPU is generated if the alarm interrupt is enabled. The alarm can additionally be configured to match days, months and year to generate the alarm interrupt. The alarm signal has been brought out from IRTC for use by MCU to allow certain wake up events.

A countdown timer with minute resolution is also provided for time keeping applications. This counter can be enabled or disabled separately. An interrupt is generated at the expiry of the counter. IRTC module also provides seven sampling timer interrupts apart from normal interrupts for alarm and countdown timeout.

A frequency compensation module is integrated into the IRTC to correct any error in 1 Hz clock due to variations in the 32 kHz clock caused by crystal inaccuracy, board variations or changes in temperature. The compensation value for both crystal and temperature variation is set by software and correction is done in hardware.

The registers in the IRTC are programmable via the Register Programming Interface. A protection mechanism is built-in the IRTC to protect against spurious writes into the IRTC by any run-away code. The protection mechanism requires the CPU to write a specific sequence of codes to the IRTC\_CTRL[1:0] bits in the control register, which if correct, allows write access to the registers. On completing the update of registers the CPU writes any value to IRTC\_CTRL[1:0] bits to enable the write protection again. After unlocking the registers, the CPU has a window of 2 seconds for updating the register space. On power on reset, a window of 15 seconds is allowed for the CPU to configure the IRTC after which the registers are locked. Any further updates would require the CPU to unlock the registers.

The IRTC battery supply maintains normal IRTC functionality when the CPU power is removed. The battery supply allows IRTC to keep functioning in case CPU is completely turned off. Reset to the IRTC block is generated only when both the battery supply and the CPU power are removed and either is powered up. The reset generation and switching of power is done external to the IRTC by an on-chip analog switch/regulator.

The IRTC is also equipped with a 32-byte RAM that is powered by the battery supply in the event of main supply being switched off. CPU can use this RAM to store any data it wants to retain in case the CPU power is switched off. This RAM loses all its contents when both CPU and battery power are removed.

IRTC can detect any intrusion via its tamper detection mechanism that is enabled automatically after reset. IRTC supports up to two external tamper events all individually configurable and any change of state on these pins, (active level of tamper pins or polarity is configurable); indicates a tamper which get stored in the IRTC tamper registers. Removal of battery after calibration can also be configured as a tamper event. The external tamper inputs pass through a low pass filter (digital) to prevent accidental assertion by noise. Any tamper detected causes an interrupt to the CPU. IRTC can also store the time and date stamp of the latest tamper event recorded. Tamper detection and its interrupt are enabled on reset.

For more details on the operation of this block, please refer to [Section 16.6, “Functional Description.”](#)

### 16.3.1 Low Power Modes

#### 16.3.1.1 Run Mode

In run mode, IRTC is fully operational.

#### 16.3.1.2 Wait Mode

In wait mode, IRTC is fully operational. Because IRTC runs off an independent 32.768 kHz clock, its time keeping functions and other functions, that depend on the 32.768 kHz clock and operate independent of CPU and time, are not lost. Only the register block's clock is gated as it runs off the bus clock. No register contents are lost.

#### 16.3.1.3 Stop Mode

In Stop mode, IRTC is fully operational. Because IRTC runs off an independent 32.768 kHz clock, its time keeping functions and other functions, which depend on the 32.768 kHz clock and operate independent of CPU and IRTC contents, are not lost. The register block's clock is gated and no contents are lost. In case power to peripherals is cut in stop mode then IRTC can run off a battery supply and isolate itself from the powered off domain

## 16.4 External Signal Description

[Table 16-1](#) shows the user-accessible signals of IRTC available at MCU's pins.

**Table 16-1. External Signals of IRTC**

S. No.	Signal Name	Type	Direction	Functional Description
1	IRTC_EXTAL	—	In	32.768 kHz crystal input
2	IRTC_XTAL	—	Out	Oscillator output to crystal
3	VBAT or VSTBY	—	In out	Battery supply or standby voltage supply

## 16.5 Register Definitions

The IRTC memory map contains twenty nine 16-bit aligned registers and Standby RAM.

All registers except the write enable bits in the control register are protected from spurious updated by any run-away code. Writing to the reserved bits has no effect and reads returns zeros. Write access to the reserved locations & registers during write protect enable being asserted generates transfer error and read access to reserved locations generates transfer error and the read data bus shows all 1s. Read to valid register locations is allowed at all times.



**NOTE**

IRTC does not check the correctness of the values programmed into its registers; hence programming illogical time & date entries will result in an undefined operation. Normal functionality is not guaranteed in that case.

Below is a detailed description of the registers. Each register is explained using a register diagram showing the bits and their reset value with access type. This is followed by a detailed description of each field.

**16.5.1 IRTC Year & Month Counters Register (IRTC\_YEARMON)**

This register stores the value of the month and year counters. The year bits do not store the year value but calculate the increment in years instead. This is a signed value and the range of values is from  $-128$  to  $127$ . The software programs the offset from that base year into this register. The BASE YEAR is hard coded as the year 2112. For example, if the current year is 2007, then this is represented in this register as  $-105$  or  $0x97$ . The actual year value can be got by adding the BASE YEAR and the offset in the IRTC\_YEARMON[15:8] register. Hence the range of year supported is from 1984 ( $2112 - 128$ ) to 2239 ( $2112 + 127$ ).

Hence for year calculation we have:

Actual Year = Base Year (i.e. 2112) + Offset Year

The month register stores the count value of the months register. Writing to this register loads the months counter with the new value. Writing any other value has no effect on this counter. Both month and year are unaffected on software reset.

MCU first reads the INVAL bit of the IRTC\_STATUS (bit 0) to determine that the counters are stable and they can be changed. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU checks the status of this bit during reads and writes.



**Figure 16-2. IRTC Year & Month Counters (IRTC\_YEARMON) Register**

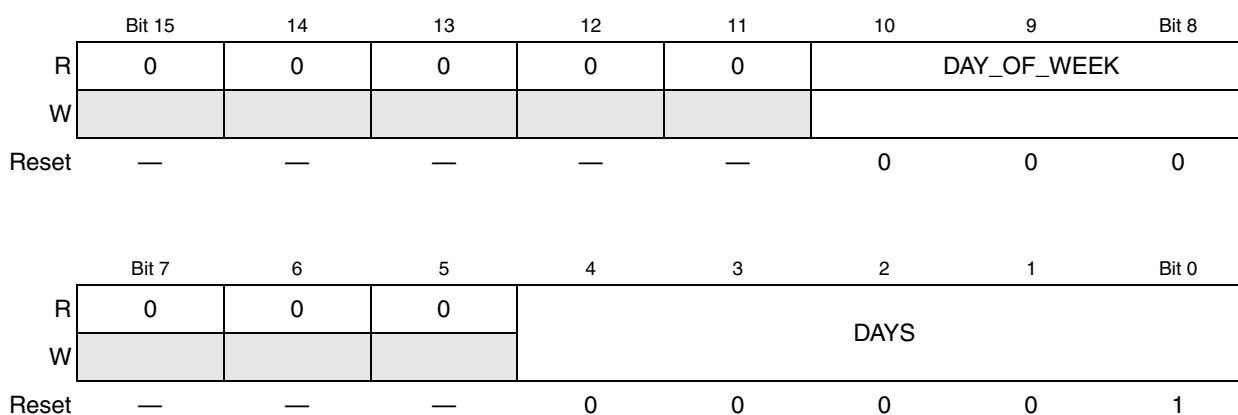
**Table 16-2. IRTC\_YEARMON Register Field Descriptions**

Field	Description
15:8 YEAR	Year Count Value. It indicates the offset in years from the base year (hard coded as 2112) and does not show the actual year value. This is a signed value. Valid values: –128 to 127 (Base year 2112 and if the value of YEAR bits is 0x10, the actual year will be 2112 + 16 = 2128).
3:0 MONTH	Month Count Value. Valid values: 1–12 1 January 2 February 3 March 4 April 5 May 6 June 7 July 8 August 9 September 10 October 11 November 12 December 0 & 13 to 15 – Reserved and are not used.

### 16.5.2 IRTC Day & Day-of-Week Counters Register (IRTC\_DAYS)

This read/write register contains the current value of the day-of-week counter and day counter. This register can be read at any time without affecting the counter count values. Writing to this register loads the value to the day-of-week and day counter and the counters continue to count from this new value. The day-of-week is not calculated automatically and must be written by CPU. It can be calculated automatically by software based on date given by the user. This register unaffected on software reset.

MCU must first read the **INVAL** bit of the **IRTC\_STATUS** (bit 0) to determine that the counters are stable and can be changed. The **INVAL** bit ensures that no operation is done at the boundary of a second when counters change value. MCU checks the status of this bit during reads and writes.

**Figure 16-3. IRTC Day & Day-of-Week Counters Register (IRTC\_DAYS) Register**

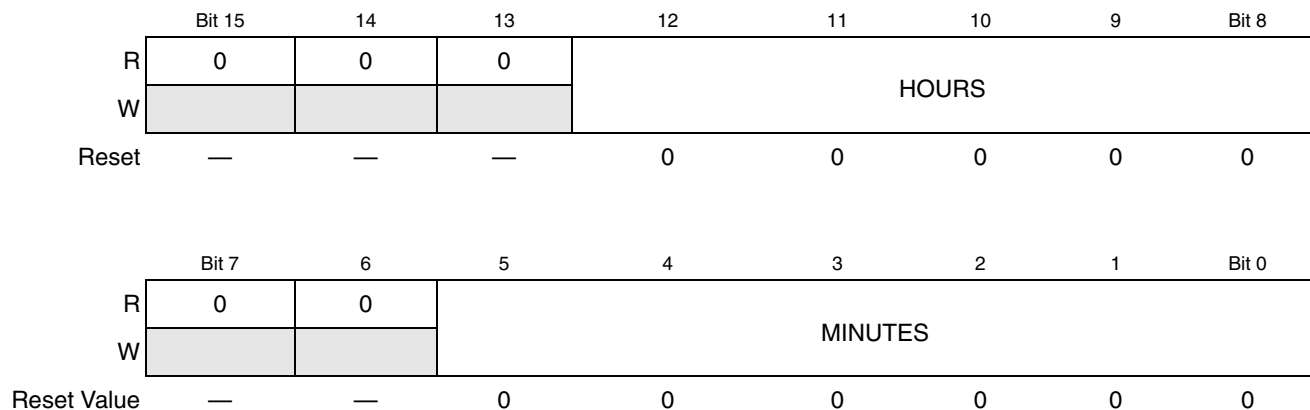
**Table 16-3. IRTC\_DAYS Register Field Descriptions**

Field	Description
10:8 DAY_OF_WEEKS	Day of the Week Count. 0 Sunday 1 Monday 2 Tuesday 3 Wednesday 4 Thursday 5 Friday 6 Saturday 7 Reserved. Must not be used
4:0 DAYS	Days counter value. Valid count: 1–31

### 16.5.3 IRTC Hours and Minutes Counters Register (IRTC\_HOURMIN)

This register is used to program the hours and minutes counter. It can be read anytime to get the current value of the counters. Only power-on reset can reset this register. Hours counter can be set between 0 and 23 both included. Minutes counter can be set between 0 and 59 both included. This register is unaffected by software reset.

MCU must first read the **INVAL** bit of the **IRTC\_STATUS** (bit 0) to determine that the counters are stable and can be changed. The **INVAL** bit ensures that no operation is done at the boundary of a second when counters change value. MCU checks the status of this bit during reads and writes.

**Figure 16-4. IRTC Hours and Minutes Counters (IRTC\_HOURMIN) Register****Table 16-4. IRTC\_HOURMIN Register Field Description**

Field	Description
15:13	Reserved bits. Read returns zeros.
12:8 HOURS	Hour Counter Value. Count: 0–23. Software interprets 0–11 as AM and 12–23 as PM.

Table 16-4. IRTC\_HOURMIN Register Field Description

7:6	Reserved bits. Read returns zeros.
5:0 MINUTES	Minutes Counter Value. Count: 0–59

### 16.5.4 IRTC Seconds Counter Register (IRTC\_SECONDS)

This register is used to program the seconds counter. It can be read anytime to get the current value of the counter. Only power-on reset can reset this register. Seconds counter can be set between 0 and 59 both included. This register is unaffected by software reset.

MCU must first read the INVAL bit of the IRTC\_STATUS (bit 0) to determine that the counters are stable and can be changed. The INVAL bit ensures that no operation is done at the boundary of a second when counters change value. MCU checks the status of this bit during reads and writes.

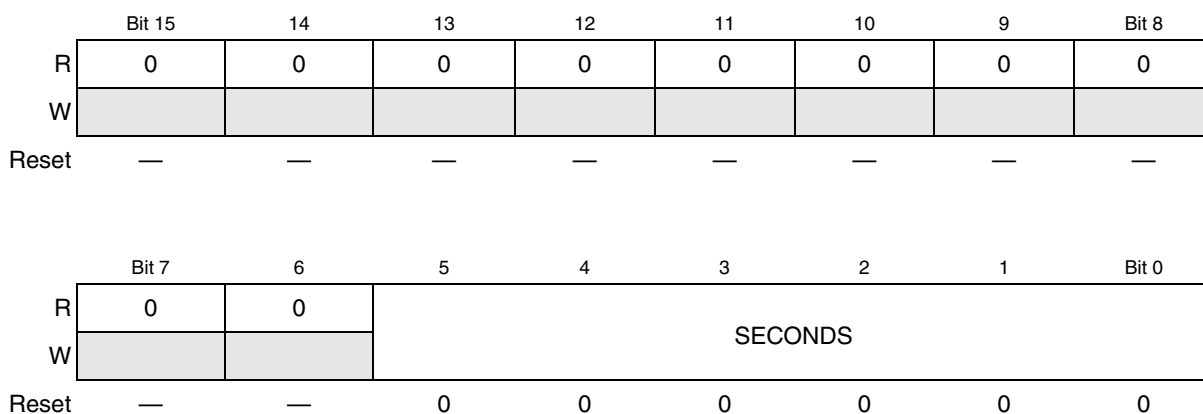


Figure 16-5. IRTC Seconds Counter (IRTC\_SECONDS) Register

Table 16-5. IRTC\_SECONDS Register Field Descriptions

Field	Description
15:8	Reserved bits. Not writeable. Read returns zeros.
7:6	Reserved bits. Read returns zeros.
5:0 SECONDS	Seconds Counter Value. Counter width parameterized. Count: 0–59

### 16.5.5 IRTC Year and Month Alarm Register (IRTC\_ALM\_YRMON)

The month and year alarm register is used to configure the month and year setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of the alarm seconds, minutes, hours, days, month and year match their respective counter values.

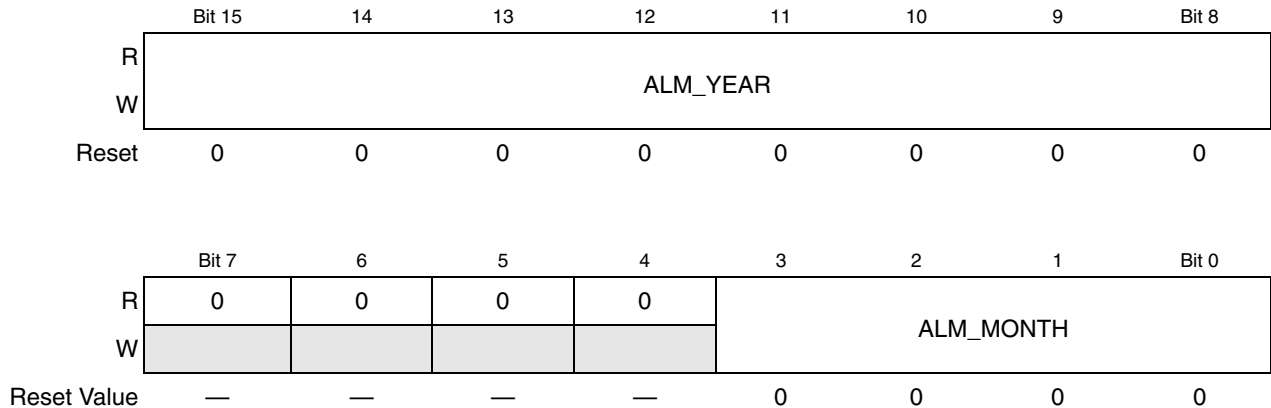


Figure 16-6. IRTC Year and Month Alarm (IRTC\_ALM\_YRMON) Register

Table 16-6. IRTC\_ALM\_YRMON Register Field Descriptions

Field	Description
15:8	Year Count Value. Indicates the offset in years from the base year (hard coded as 2112) and does not show the actual year value. This is a signed value. Valid Values: –128 to +127 (Base Year 2112 and if the value of YEAR bits is 0x10, the actual year will be 2112 + 16 = 2128).
7:4	Reserved bits. Read returns zeros.
3:0	Month Alarm Value. Valid Values: 1 – 12 <div> <div>1 January 2 February 3 March 4 April 5 May</div> <div>6 June 7 July 8 August 9 September 10 October</div> <div>11 November 12 December 13 to 15 – Reserved and are not used. 0 No match done</div> </div>

### 16.5.6 IRTC Days Alarm Register (IRTC\_ALM\_DAYS)

The days alarm register is used to configure the day setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month and year match their respective counter values.

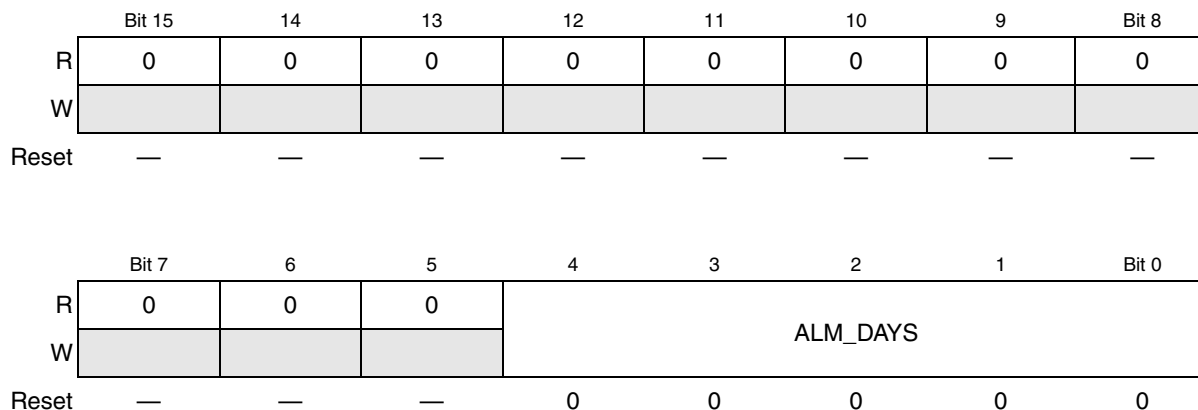


Figure 16-7. IRTC Days Alarm (IRTC\_ALM\_DAYS) Register

Table 16-7. IRTC\_ALM\_DAYS Register Field Descriptions

Field	Description
15:8	Reserved bits. Read returns zeros.
7:5	Reserved bits. Read returns zeros.
4:0 ALM_DAYS	Days Alarm value. Valid values: 1–31

## 16.5.7 IRTC Hours and Minutes Alarm Register (IRTC\_ALM\_HM)

The hours and minutes alarm register is used to configure the hour and minute setting of the alarm. The alarm setting can be read or written at anytime. This register is reset to default state on software reset. In BCD mode, all 8-bits are used to indicate the time.

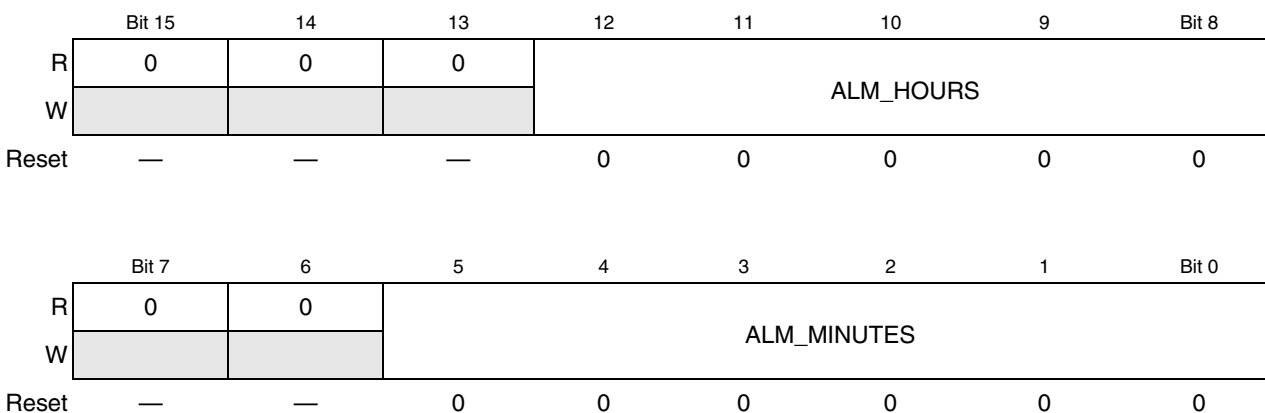


Figure 16-8. IRTC Hours and Minutes Alarm (IRTC\_ALM\_HM) Register

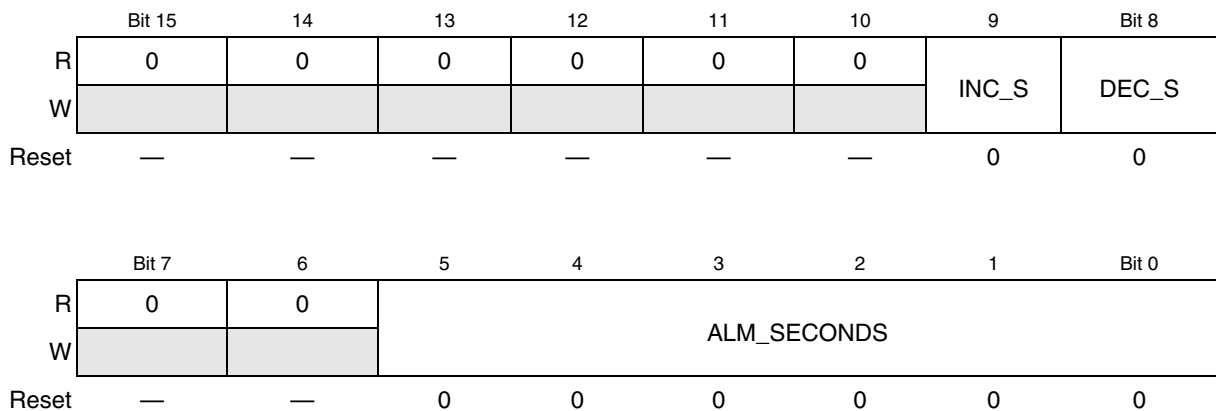
**Table 16-8. IRTC\_ALM\_HM Register Field Descriptions**

Field	Description
15:13	Reserved bits. Read returns zeros.
12:8 ALM_HOURS	Alarm Hour Value. Count: 0 – 23
7:6	Reserved bits. Read returns zeros.
5:0 ALM_MINUTES	Alarm Minutes Value. Count: 0 – 59

### 16.5.8 IRTC Seconds Alarm Register (IRTC\_ALM\_SECS)

The seconds alarm register is used to configure the seconds setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default value on software reset. In BCD mode, all 8-bit are used to indicate the time.

Bits 9 and 8 provide options to MCU to perform correction on seconds counter to compensate for the leap seconds. Writing to these bits adds or subtracts 1 from the seconds counter and read returns zeros. MCU must first read the **INVAL** bit of the **IRTC\_STATUS** (bit 0) to determine that the counters are stable and can be incremented. The **INVAL** bit ensures that no operation is done at the boundary of a second when counters change value. MCU checks the status of this bit both during reads and writes.

**Figure 16-9. IRTC Seconds Alarm (IRTC\_ALM\_SECS) Register****Table 16-9. IRTC\_ALM\_SECS Register Field Descriptions**

Field	Description
15:10	Reserved bits. Not writeable. Read returns zeros.
9 INC_S	This bit controls the increment of seconds counter in case the MCU wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit increments the seconds counter and then the bit gets cleared on next posedge.

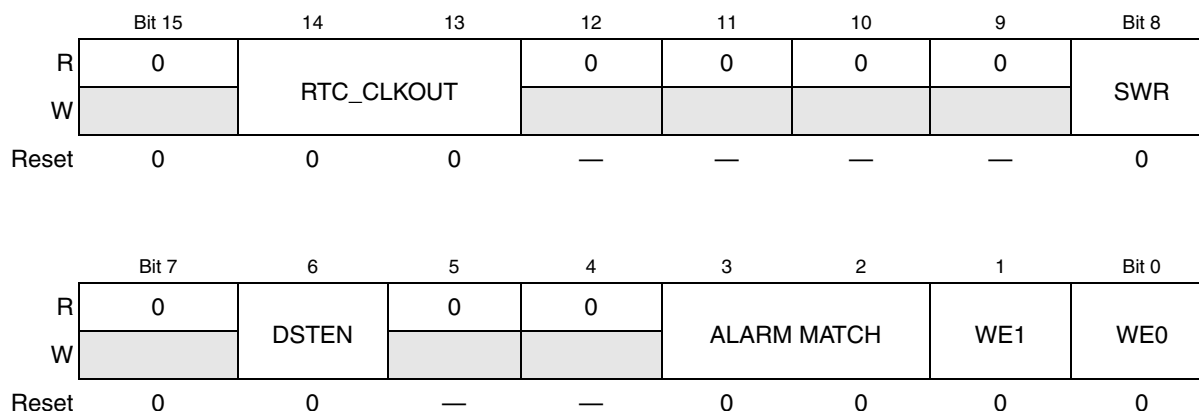
**Table 16-9. IRTC\_ALM\_SECS Register Field Descriptions**

8 DEC_S	This bit controls the decrement of seconds counter in case the MCU wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit has decrements the seconds counter and then the bit gets cleared on next posedge of bus clock.
7:6	Reserved bits. Read returns zeros.
5:0 ALM_SECONDS	Alarm Seconds Value. Count: 0 – 59

### 16.5.9 IRTC Control Register (IRTC\_CTRL)

This is a control register and governs all operations inside the IRTC. This register is used to specify the software reset, daylight controls and write protection control.

All registers including the RAM are protected against spurious updates by the write protect mechanism. To unlock the register bits, specific pattern has to be written in the WE[1:0] bits. Only the write protect bits are freely writeable by the CPU. The write protect bits WE[1:0] are self clearing bits that always return zeros on read. The sequence must be written into these bits to enable or disable write protection.

**Figure 16-10. IRTC Control (IRTC\_CTRL) Register****Table 16-10. IRTC\_CTRL Register Field Descriptions**

Field	Description
15	Reserved bits. Not writeable. Read returns zeros.
14:13 RTC_CLKOUT	RTC Clock Output Enable and Select. These bits control the clock to be output from the IRTC block for external use. 00 No clock is output 01 Compensated 1 Hz clock output (if COMPENSATION feature is enabled). 1 Hz clock output from Prescaler (if COMPENSATON feature is not enabled) 10 Buffered Oscillator clock output 11 Reserved
12:9	Reserved bits. Not writeable. Read returns zeros.



Table 16-10. IRTC\_CTRL Register Field Descriptions (continued)

Field	Description
8 SWR	Software Reset. This is a self clearing bit. It automatically gets cleared on the next clock after write. 1 Software Reset 0 No Software Reset Software reset clears the contents of alarm, interrupt (status & enable except tamper) registers and has no effect on DST, standby RAM, up counter, time, calendaring and tamper detect registers.
7	Reserved bit. Not writeable. Read returns zeros
6 DSTEN	Daylight Saving Enable. Automatic adjustment of time is done when enabled. 1 Enabled. Daylight saving changes are applied. 0 Disabled. Daylight saving changes are NOT applied.
5:4	Reserved bits. Not writeable. Read returns zeros
3:2 ALARM MATCH	Alarm Match bits. These bits define which time and calendar counters will be used for matching and generating an alarm 00 Only Seconds, Minutes and Hours matched 01 Only Seconds, Minutes, Hours and Days matched 10 Only Seconds, Minutes, Hours, Days and Months matched 11 Only Seconds, Minutes, Hours, Days, Months and Year (offset) matched
1:0	Write Enable bits. Controls the entry and exit into/from the Register/Memory write protection mode. Self clearing bits. Reads will return zeros. 10 Enable Write Protection – Registers are locked 00 – 01 – 11 – 10' - Disable Write Protection – Registers are unlocked <b>NOTE:</b> When the registers are unlocked, they remain in this unlocked state for a time of 2 seconds after which the registers are automatically locked. After power-on-reset too, the registers come out as unlocked but they get locked automatically 15 seconds after power on.

### 16.5.10 IRTC Status Register (IRTC\_STATUS)

This register indicates the status of various processes going inside the IRTC. This register also helps the MCU read time or date register when their values are stable and not changing. Software reset resets the register to its default state. Done bit and Bus Error Bits get cleared by writing 1 to them. PORB bit is NOT cleared by software reset.

	Bit 15	14	13	12	11	10	9	Bit 8
R	0	0	0	0	0	0	0	0
W								
Reset	—	—	—	—	—	—	—	—

	Bit 7	6	5	4	3	2	1	Bit 0
R	0	PORB	CLV	WPE	OCAL	BERR	C_DON	INVAL
W								
Reset	—	1	0	0	0	0	0	0

Figure 16-11. IRTC Status (IRTC\_STATUS) Register

Table 16-11. IRTC\_STATUS Register Field Descriptions

Field	Description
15:7	Reserved bits. Not writeable. Read returns zeros.
6 PORB	Boot Source after POR. This read-only bit indicates that chip has come out of POR and booting there after. On entering standby mode, this bit is cleared which indicates that chip is booting after standby exit. This bit is asserted on IRTC POR only, which indicates that entire chip has been reset. 0 Chip booting after exiting standby mode 1 Chip booting after POR (RTC POR)
5 CLV	CPU Low Voltage. This read-only bit is asserted when the MCU/CPU power falls below the read only threshold when all write cycles are terminated normally and no change is done to the registers. Registers are read only. Transfer Error is not asserted. 0 CPU in normal operating voltage 1 CPU voltage is below normal operating voltage. IRTC registers in read-only mode
4 WPE	Write Protect Enable. This read-only bit indicates that registers are in locked mode and write to registers is disabled. Any write access made to the register space when write protection is enabled (i.e. registers in locked mode) will cause the transfer error signal to be asserted. 0 Registers are unlocked and can be accessed 1 Registers are locked and in read-only mode
3 OCAL	Calibration Output Signal or Compensation Interval Bit. This status bit is read-only and is asserted for a time equal to compensation interval seconds. This bit is used by MCU to calculate the interrupts serviced during this interval and perform corrections in case of deviations (i.e. Calibration). This bit toggles on every compensation interval start and is either 0 or 1 during the entire duration. This bit will not toggle if compensation logic has been disabled by MCU.
2 BERR	Bus Error. Indicates that a read or write cycle was started by MCU when the INVAL bit is set. Write access to time/date registers gets nullified (terminate normally) and no register value gets changed. Read during INVAL bit asserted returns 16'hFFFF. No Transfer Error is asserted. This bit gets cleared by writing 1 to it. 0 Read & Write accesses are normal 1 Read or Write accesses occurred when INVAL bit is asserted
1 C_DON	Compensation Done. This bit indicates that current compensation cycle is complete. The bit gets cleared by writing 1 to it. Reserved in Basic & Standard Feature sets. 1 Compensation Completed 0 Compensation busy or not enabled This bit is asserted <u>seven</u> oscillator cycles (i.e. 7 * 30.52us) before actual compensation interval expires so that back to back compensation can be enabled
0 INVAL	Invalid Time bit. This read-only bit indicates the time is invalid or changing and should not be read. This bit is asserted 1 oscillator clock cycle before and after the 1 Hz (seconds') boundary. Write access to time/date registers gets nullified (terminate normally) and no register value gets changed. Read during INVAL bit asserted returns 16'hFFFF. No Transfer Error is asserted. 1 Counter values changing. Time/Date is invalid 0 Time can be read. Time is valid

### 16.5.11 IRTC Interrupt Status Register (IRTC\_ISR)

The real-time clock interrupt status register (IRTC\_ISR) indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs, the bit will be set in this register regardless of its corresponding interrupt enable bit being set. The status bits are cleared by writing a value of 1, which also clears the interrupt. Interrupts may occur while the system clock is idle or in standby mode. When the system enters the active power mode, interrupt will be indicated to the CPU.

The first event of the sampling timer interrupts after power on reset must not be used to qualify any periodic interval. However, the correct periodic interval (i.e. 512 Hz or 256 Hz, etc.) must be determined using two sampling timer interrupts. The time between two interrupts would always be the correct time period.

Tamper interrupt status is set on reset as POR is generated when battery and CPU power are unavailable and either is powered up. Removal of battery is considered as a tamper and hence this bit is set on reset.

The status register is also cleared on software reset except for the tamper status which remains unaffected.

	Bit 15	14	13	12	11	10	9	Bit 8
R								
W								
	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0
Reset	0	0	0	0	0	0	0	0

	Bit 7	6	5	4	3	2	1	Bit 0
R								
W								
	2Hz	1Hz	MIN	HR	DAY	ALM	CDT	TMPR
Reset	0	0	0	0	0	0	0	1

**Figure 16-12. IRTC Interrupt Status (IRTC\_ISR) Register**

**Table 16-12. IRTC\_ISR Register Field Descriptions**

Field	Description
15 SAM7	Sampling Timer Interrupt Flag 7; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 512 Hz
14 SAM6	Sampling Timer Interrupt Flag 6; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 256 Hz
13 SAM5	Sampling Timer Interrupt Flag 5; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 128 Hz
12 SAM4	Sampling Timer Interrupt Flag 4; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 64 Hz
11 SAM3	Sampling Timer Interrupt Flag 3; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 32 Hz
10 SAM2	Sampling Timer Interrupt Flag 2; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 16 Hz
9 SAM1	Sampling Timer Interrupt Flag 1; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 8 Hz
8 SAM0	Sampling Timer Interrupt Flag 0; indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 4 Hz

**Table 16-12. IRTC\_ISR Register Field Descriptions**

7 2Hz	Sampling Timer Interrupt at 2 Hz frequency, indicates that an interrupt has occurred. If enabled, this bit periodically sets at a rate equal or close to 2 Hz
6 1Hz	1 Hz Flag. Indicates that the seconds counter has incremented
5 MIN	Minutes Flag. Indicates that the minutes counter has incremented
4 HR	Hour Flag. Indicates that the hour counter has incremented
3 DAY	Day Flag. Indicates that the day counter has incremented
2 ALM	Alarm Flag. Indicates that the alarm value programmed matches the counter values. This interrupt is generated only when the seconds, minutes, hours and days counter match the alarm register values
1 CDT	Count Down Timer Expiry Flag. Indicates that the programmed count in the count down timer has expired.
0 TMPR	Tamper Detect Interrupt, indicates that a tamper has been done to the system and security is compromised.

### 16.5.12 IRTC Interrupt Enable Register (IRTC\_IER)

The real-time clock interrupt enable register (IRTC\_IER) is used to enable/disable the various real-time clock interrupts. Disabling an interrupt bit has no effect on its corresponding status bit.

	Bit 15	14	13	12	11	10	9	Bit 8
R								
W	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0
Reset	0	0	0	0	0	0	0	0

	Bit 7	6	5	4	3	2	1	Bit 0
R								
W	2Hz	1Hz	MIN	HR	DAY	ALM	CTD	TMPR
Reset	0	0	0	0	0	0	0	1

**Figure 16-13. IRTC Interrupt Enable (IRTC\_IER) Register**

**Table 16-13. IRTC\_IER Register Field Descriptions**

<b>Field</b>	<b>Description</b>
15 SAM7	Sampling Timer Interrupt Flag 7, enable/disable interrupt 1 Enable 0 Disable
14 SAM6	Sampling Timer Interrupt Flag 6, enable/disable interrupt 1 Enable 0 Disable
13 SAM5	Sampling Timer Interrupt Flag 5, enable/disable interrupt 1 Enable 0 Disable
12 SAM4	Sampling Timer Interrupt Flag 4, enable/disable interrupt 1 Enable 0 Disable
11 SAM3	Sampling Timer Interrupt Flag 3, enable/disable interrupt 1 Enable 0 Disable
10 SAM2	Sampling Timer Interrupt Flag 2, enable/disable interrupt 1 Enable 0 Disable
9 SAM1	Sampling Timer Interrupt Flag 1, enable/disable interrupt 1 Enable 0 Disable
8 SAM0	Sampling Timer Interrupt Flag 0, enable/disable interrupt 1 Enable 0 Disable
7 2Hz	Sampling Timer Interrupt at 2 Hz frequency, enable/disable interrupt 1 Enable 0 Disable
6 1Hz	1 Hz Flag, enable disable interrupt 1 Enable 0 Disable
5 MIN	Minutes Flag, enable disable interrupt 1 Enable 0 Disable
4 HR	Hour Flag, enable disable interrupt 1 Enable 0 Disable
3 DAY	Day Flag, enable disable interrupt 1 Enable 0 Disable
2 ALM	Alarm Flag, enable disable interrupt 1 Enable 0 Disable

**Table 16-13. IRTC\_IER Register Field Descriptions (continued)**

Field	Description
1 CTD	Count Down Timer Interrupt, enable disable interrupt 1 Enable 0 Disable
0 TMPR	Tamper Detect Interrupt, enable disable interrupt 1 Enable 0 Disable <b>NOTE:</b> This interrupt gets enabled on POR and is unaffected by software reset.

Alarm interrupt is set as per the following table.

**Table 16-14. Alarm Match Table**

Register Bits (IRTC_CTRL[3:2])	Counters Matched	Alarm Type
00	Seconds, Minutes and Hours matched	Daily
01	Seconds, Minutes, Hours and Days matched	Monthly
10	Seconds, Minutes, Hours, Days and Months matched	Yearly
11	Seconds, Minutes, Hours, Days, Months & Year (offset) matched	One-Time

A single interrupt is output from this block which is an ORed version of all the above interrupt. The CPU reads the status register in the interrupt status routine to determine which interrupt has occurred.

The tamper detect interrupt enable is an exception as it is enabled after POR. All interrupts enables except tamper interrupt enable are reset to default state on software reset.

### 16.5.13 IRTC Minutes Down Counter Register (IRTC\_COUNT\_DN)

This counter is used to generate an interrupt on a minute boundary, for example, to turn off the LCD controller after five minutes of inactivity. The countdown timer is decremented by the minute (MIN) tick output from the real-time clock, so the average tolerance of the count is 0.5 minutes. For better accuracy, enable the countdown timer by waiting for the MIN bit of the IRTC\_ISR register to be set (i.e. waiting for the minute interrupt). Loading the countdown timer counter with 0 has no effect. Software reset brings the countdown timer count to its default state. Interrupt is generated when the timer reaches 0. The value in the register shows the current value of the counter at all times. Note that the actual delay includes the seconds from setting the countdown to the next minute tick.

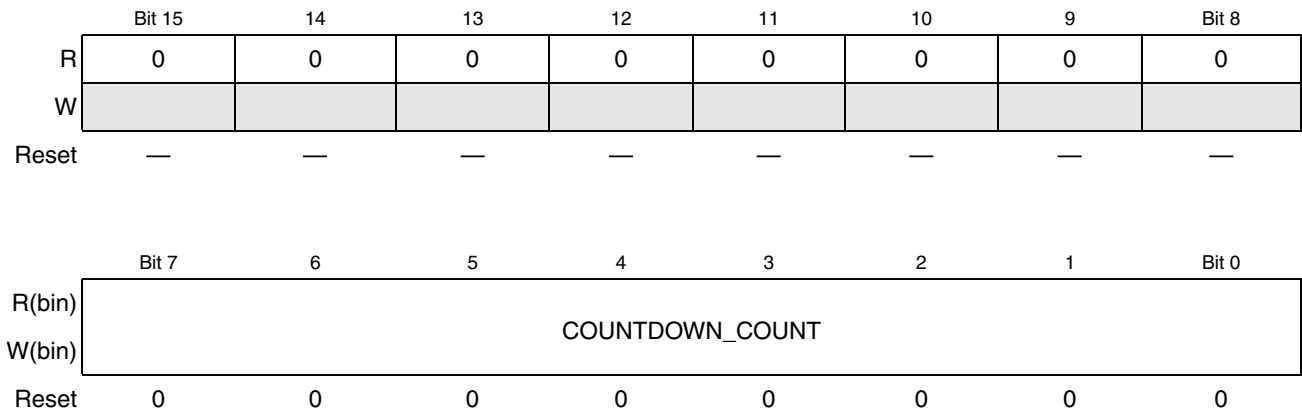


Figure 16-14. IRTC Minutes Down Counter (IRTC\_COUNT\_DN) Register

Table 16-15. IRTC\_COUNT\_DN Register Field Descriptions

Field	Description
15:8	Reserved Bits. Read returns zeros.
7:0 RTC_COUNT_DN	Countdown counter's value. Zero disables the counter.

### 16.5.14 IRTC Daylight Saving Hour Register (IRTC\_DST\_HOUR)

This register stores the time in hours when the daylight saving is applied or reversed. This register is programmable when the DST\_EN bit in IRTC\_CTRL register is 0. When DST\_EN bit is set, the contents of this register cannot be changed. The CPU programs the correct hour value (0 – 23) as per the regional settings. For example, if the daylight saving starts at 2:00 AM on March 25 and ends at 2:00 AM on October 28 in 2007 then the time at which time advances or falls back is actually 1:59 AM. Hence the CPU must program 1 for the hour count value (and not 2) i.e. write 0x0101 in this register. 59 minutes count is automatically checked inside IRTC and hence not required to be programmed. This register has no effect on software reset.

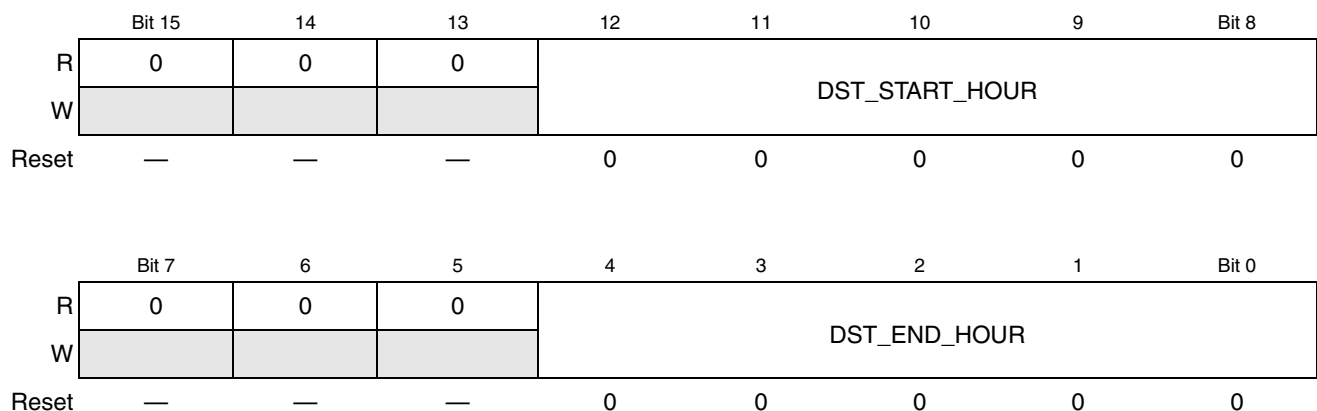


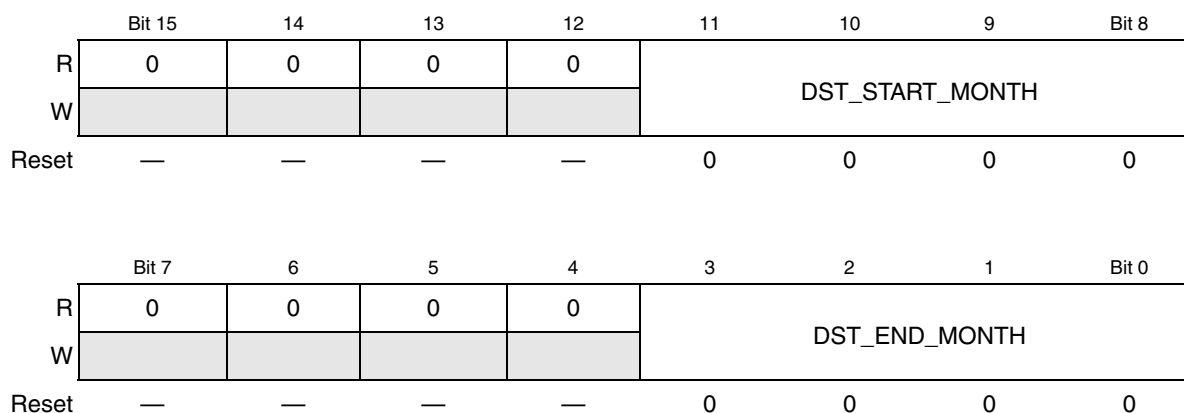
Figure 16-15. IRTC Daylight Saving Hour (IRTC\_DST\_HOUR) Register

**Table 16-16. IRTC\_DST\_HOUR Field Descriptions**

Field	Description
15:13	Reserved bits. Read returns zeros.
12:8 DST_START_HOUR	Hour value when daylight saving starts. Valid values: 0 – 23
7:5	Reserved bits. Read returns zeros.
4:0 DST_END_HOUR	Hour value when daylight saving ends. Valid values: 0 – 23

#### 16.5.14.1 IRTC Daylight Saving Month Register (IRTC\_DST\_MNTH)

This register stores the month information when the daylight saving is applied or reversed. This register is programmable when the DST\_EN bit in IRTC\_CTRL register is 0. When DST\_EN bit is set, the contents of this register cannot be changed. The CPU programs the correct month value (1–12) as per the regional settings. For example, if the daylight saving starts at March 25 and ends at October 28 in 2007. Hence the CPU writes 0x030A to this register. This register has no effect on software reset.

**Figure 16-16. IRTC Daylight Saving Month (IRTC\_DST\_MNTH) Register****Table 16-17. IRTC\_DST\_MNTH Field Descriptions**

Field	Description
15:12	Reserved bits. Read returns zeros.
11:8 DST_START_MONTH	Month value when Daylight saving has to start. Valid values: 1 – 12
7:4	Reserved bits. Read returns zeros.
3:0 DST_END_MONTH	Month value when Daylight saving has to end. Valid values: 1 – 12



### 16.5.15 IRTC Daylight Saving Day Register (IRTC\_DST\_DAY)

This register stores the day information when the daylight saving is applied or reversed. This register is programmable when the DST\_EN bit in IRTC\_CTRL register is 0. When DST\_EN bit is set, the contents of this register cannot be changed. The CPU programs the correct day value (1–31) as per the regional settings. For example, if the daylight saving starts at March 25 and ends at October 28 in 2007. Hence the CPU writes 0x191C to this register. This register is unaffected by software reset.

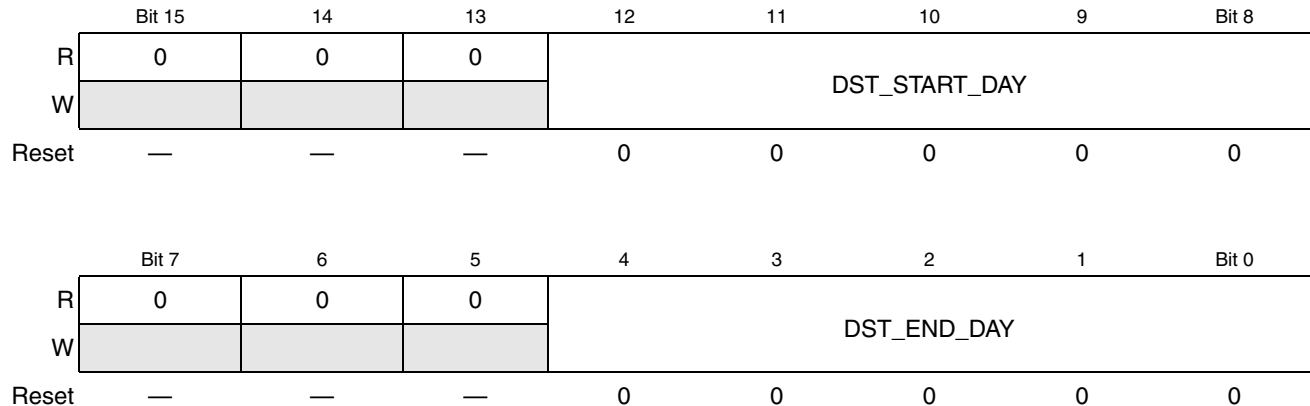


Figure 16-17. IRTC Daylight Saving Day (IRTC\_DST\_DAY) Register

Table 16-18. IRTC\_DST\_DAY Field Descriptions

Field	Description
15:13	Reserved bits. Read returns zeros.
12:8 DST_START_DAY	Day value when Daylight saving has to start. Valid values: 1 – 31
7:5	Reserved bits. Read returns zeros.
4:0 DST_END_DAY	Day value when Daylight saving has to end. Valid values: 1 – 31

### 16.5.16 IRTC Compensation (IRTC\_COMPEN)

The compensation register stores the compensation value that is used by the compensation block to correct the 1 Hz clock. This value provides the number of oscillator clock cycles to be added or removed. The value stored is in the 2's complement format. The range of the value is programmed from –128 to 127. When the compensation cycle completes, the 'Done' bit in the status register is set. IRTC continues to compensate with this value till disabled. If a new value is programmed in between, then that cycle starts after the compensation cycle is over. In the current compensation cycle, new values can be changed many times and the latest value of the current compensation cycle will be used in the next compensation cycle.

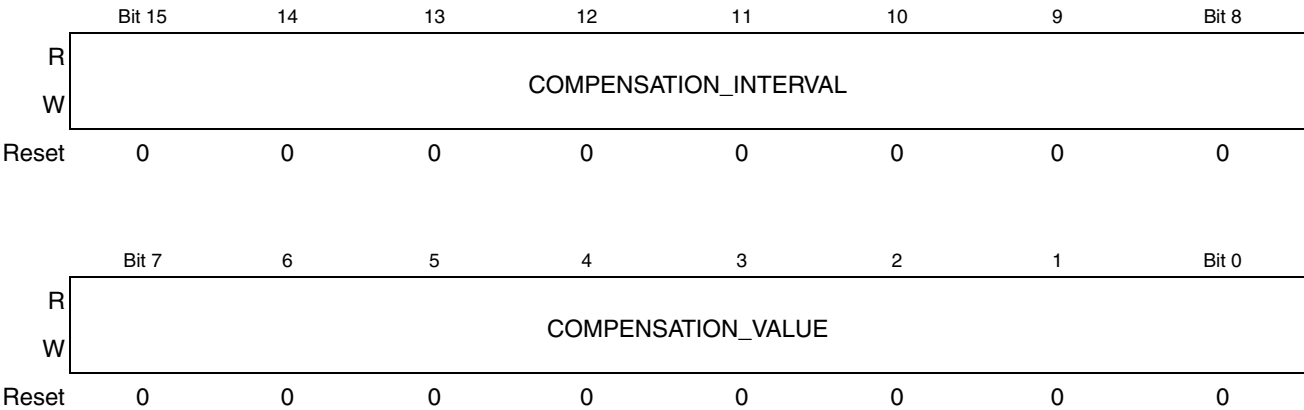


Figure 16-18. IRTC Compensation (IRTC\_COMPEN) Register

Table 16-19. IRTC\_COMPEN Field Descriptions

Field	Description
15:8 COMPENSATIO N_INTERVAL	Compensation Interval. Indicates the window over which the compensation has to be carried out. Minimum interval is 1 second and maximum is 255 seconds. A compensation interval of 0 disables the compensation logic. MCU can simply write zeros to these bits to disable the compensation logic. Non-zero value starts the compensation logic.
7:0 COMPENSATIO N_VALUE	Compensation value. Two's complement number which indicates the number of Oscillator clock cycles the IRTC requires to compensate for the specified compensation interval. Range: -128 to +127. A value of zero indicates no compensation is needed.

16.5.17 IRTC Tamper Time Stamp Year & Month Register (IRTC\_TTSR\_YM)

This register is a used to store the value of months and year counters when a tamper is detected. This register serves as a purpose for CPU to know when a tamper had occurred. This register is has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. Writing to this register has no effect and is a read only register.

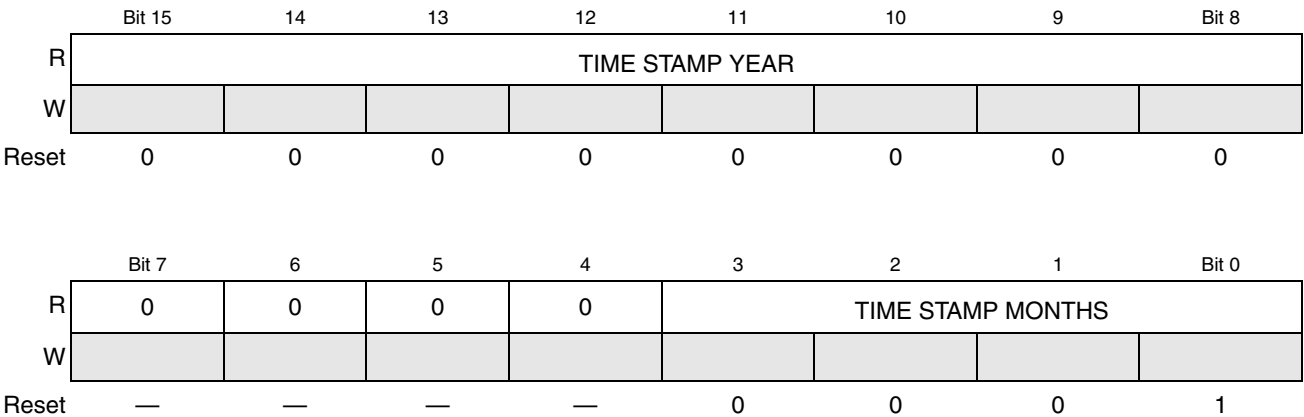


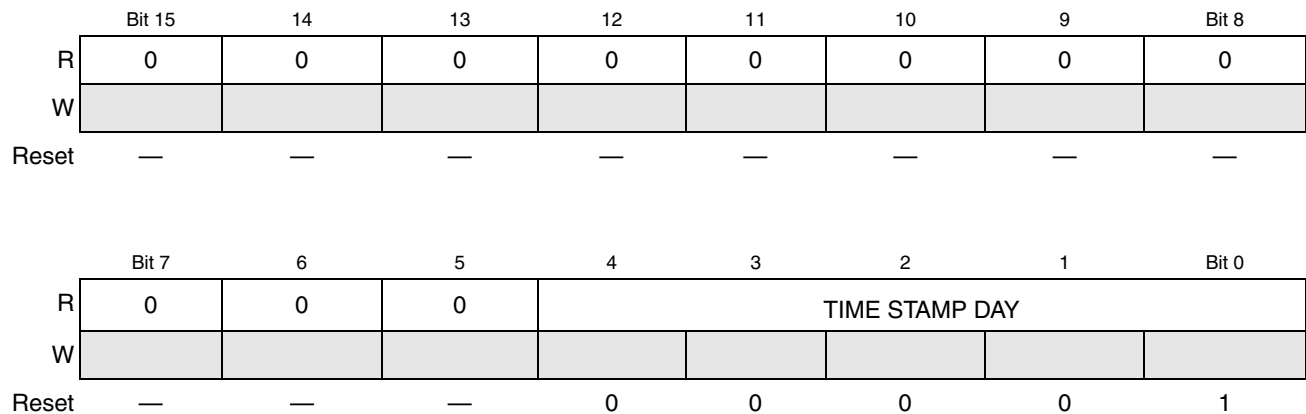
Figure 16-19. IRTC Tamper Time Stamp Year & Month (IRTC\_TTSR\_YM) Register

**Table 16-20. IRTC\_TTSR\_YM Field Descriptions**

Field	Description
15:8 TIME STAMP YEAR	Tamper Detect Year Value. Count: 0 – 255
7:4	Reserved bits. Read returns zeros. Not Writeable
3:0 TIME STAMP MONTHS	Tamper Detect Month Value. Valid Values: 1 – 12 1 January                      6 June                              11 November 2 February                    7 July                                12 December 3 March                        8 August 4 April                         9 September                      0 & 13 to 15 – Reserved and are not used. 5 May                         10 October

### 16.5.18 IRTC Tamper Time Stamp Day Register (IRTC\_TTSR\_DAY)

This read only register is used to store the value of days counter when a tamper is detected. It is for CPU to know when a tamper occurred and has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. Writing to this register has no effect.

**Figure 16-20. IRTC Tamper Time Stamp Day Register (IRTC\_TTSR\_DAY)****Table 16-21. IRTC\_TTSR\_DAY Field Descriptions**

Field	Description
15:8	Reserved bits. Read returns zeros. Not Writeable
7:5	Reserved bits. Read returns zeros.
4:0 TIME STAMP DAY	Tamper Detect Days Value. Count: 1 – 31

### 16.5.19 IRTC Tamper Time Stamp Hours & Minutes Register

This read-only register is used to store the value of hours and minutes counters when a tamper is detected. it is for CPU to know when a tamper occurred and has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected. Writing to this register has no effect.

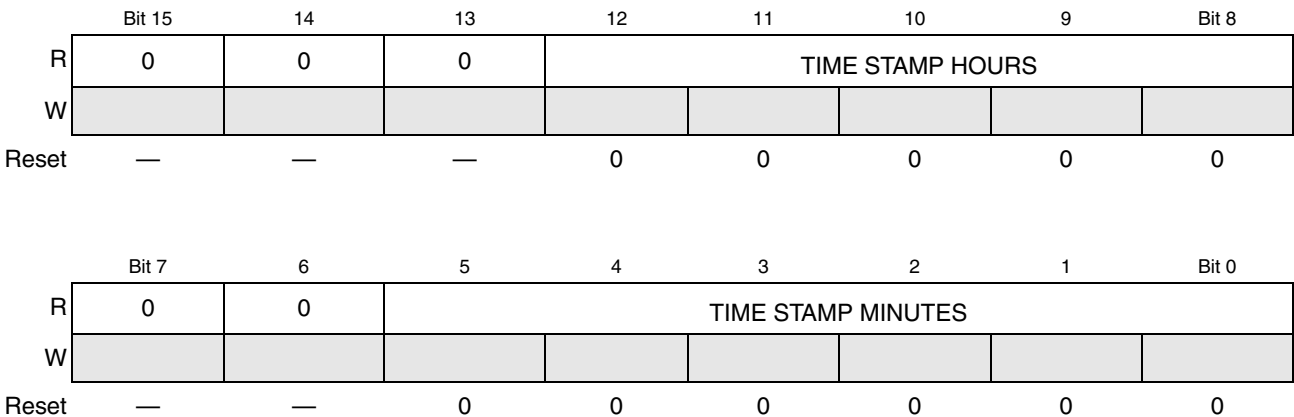


Figure 16-21. IRTC Tamper Time Stamp Hours & Minutes Register

Table 16-22. IRTC Tamper Time Stamp Hours & Minutes Register Field Descriptions

Field	Description
15:13	Reserved bits. Read returns 0.
12:8 TIME STAMP HOURS	Tamper Detect Hours Value. Count: 0–23
7:6	Reserved bits. Read returns 0.
5:0 TIME STAMP MINUTES	Tamper Detect Minutes Value. Count: 0–59

### 16.5.20 IRTC Tamper Time Stamp Seconds Register (IRTC\_TTSR\_SEC)

This register is a used to store the value of seconds counter when a tamper is detected. It is for CPU to know when a tamper occurred. This register has no effect on software reset. The value shown in this register is for recent tamper detection only. Previous intrusions cannot be detected.

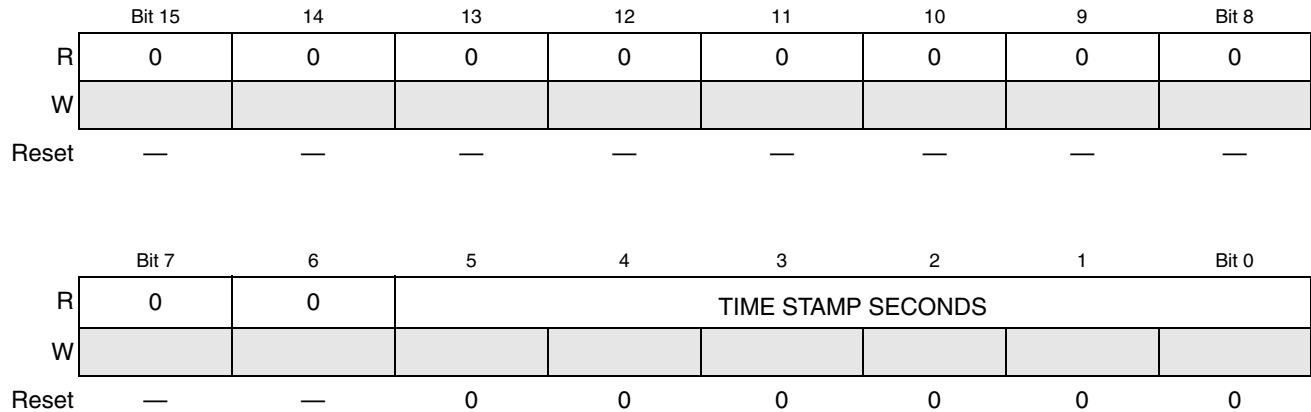


Figure 16-22. IRTC Tamper Time Stamp Seconds (IRTC\_TTSR\_SEC) Register

Table 16-23. IRTC\_TTSR\_SEC Field Descriptions

Field	Description
15:8	Reserved bits. Read returns 0.
7:6	Reserved bits. Read returns 0.
5:0 TIME STAMP SECONDS	Tamper Detect Seconds Value. Count: 0–59. Read only.

### 16.5.21 IRTC Tamper Status & Control Register (IRTC\_TAMPER\_SCR)

This register stores the tamper event and provides control to the user to enable or disable each tamper individually. No tamper is disabled automatically unless corresponding control bit is de-asserted by the software. The tamper statuses are stored as active high after being filtered by the tamper block. The tamper status bits store the tamper event, no matter their corresponding control bit being asserted or not. The control bits provide enabling/disabling of corresponding tamper events (status bits). The control bits gate the assertion of tamper interrupt bit in IRTC\_ISR.

These status and controls bits combined assert the tamper interrupt in the IRTC\_ISR register. The tamper interrupt will be cleared when all above status bits are cleared. Battery tamper is asserted on POR for the case when CPU power is removed and then battery is removed (a kind of tamper). Software should be able to differentiate between a normal battery removal and forceful battery removal. The other two tamper status bits reflect the status of the two TAMPER pins of SoC.

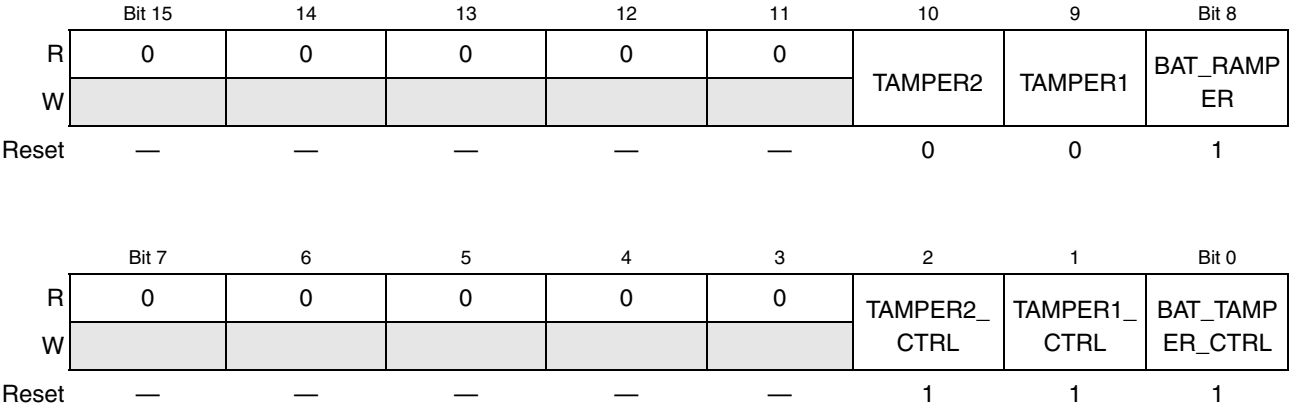


Figure 16-23. IRTC Tamper Status & Control (IRTC\_TAMPER\_SCR) Register

Table 16-24. IRTC\_TAMPER\_SCR Field Descriptions

Field	Description
15:11	Reserved bits. Read returns 0.
10:8 TAMPER_STAT US	Tamper Status Bits. Configured via parameter TAMPER_PINS. Each bit can be used by the MCU to indicate a tamper event. All these bits are write 1 to clear bits. 1 Tamper event has occurred 0 No tamper
7:3	Reserved bits. Read returns 0.
2:0 TAMPER_CTRL	Tamper Control Bits. Configured via parameter TAMPER_PINS. Each control bit enables or disables corresponding tamper status bit. All tampers are enabled on reset. 1 Tamper Status bit is enabled 0 Tamper Status bit is disabled

16.5.22 IRTC Tamper Filter 0 & 1 Control Register (IRTC\_FILTER01\_CTRL)

This register must be configured once during the initial startup of application and must not be changed on the fly to avoid erratic behavior. Changing the register on the fly might lead to missing of tamper events or assertion of spurious tamper events.

Tamper 0 is an internal tamper and hence no filtering required.

	Bit 15	14	13	12	11	10	9	Bit 8
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

	Bit 7	6	5	4	3	2	1	Bit 0
R	POL	CLKSEL	FILTER1 DURATION					
W								
Reset	0	0	0	0	0	0	0	0

Figure 16-24. IRTC Tamper Filter 0 &amp; 1 Control (IRTC\_FILTER01\_CTRL) Register

Table 16-25. IRTC\_FILTER01\_CTRL Field Descriptions

Field	Description
15:8	Reserved bits. Not writeable. Read returns zeros.
7 POL	Tamper Detect Input Bit 1 Polarity Control. This bit controls the polarity of tamper detect input bit 1 ( <i>tamper_detect[1]</i> ). 0 Tamper detect input bit 1 is active high 1 Tamper detect input bit 1 is active low
6 CLKSEL	Tamper Filter 1 Clock Select. This bit is a write once bit and selects the clock source for tamper filter for tamper detect input bit 1. 0 Clock to tamper filter 1 is 32.768 kHz (Oscillator clock) Tamper filter duration is 45.5 $\mu$ s (i.e. 1.5 clock) to 1.95ms (64 clocks) in increments of 30.5 $\mu$ s 1 Clock to tamper filter 1 is 512 Hz Tamper filter duration is 2.85 ms (i.e. 1.5 clock) to 125 ms (64 clocks) in increments of 1.95 ms
5:0 FILTER1 DURATION	Tamper Detect Bit 1 Filter Duration. This bit indicates the number of tamper filter clock cycles for which the <i>tamper_detect[1]</i> signal must remain stable before being detected as a tamper. These bits are used by the tamper filtering operation. 0 Filtering operation disabled 1 to 63 number of tamper filter clock cycles to be counted when tamper is asserted With the tamper duration set to 0, any tamper detected on the tamper pins will directly set the tamper status and interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins causes a tamper interrupt.

### 16.5.23 IRTC Tamper Filter 2 & 3 Control Register (IRTC\_FILTER23\_CTRL)

This register must be configured once during the initial startup of application and must not be changed on the fly to avoid erratic behavior. Changing the register on the fly might lead to missing of tamper events or assertion of spurious tamper events.

Tamper 2 is an internal tamper and hence no filtering is required.

	Bit 15	14	13	12	11	10	9	Bit 8
R	POL	CLKSEL	FILTER2 DURATION					
W								
Reset	0	0	0	0	0	0	0	0

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

Figure 16-25. IRTC Tamper Filter 2 &amp; 3 Control (IRTC\_FILTER23\_CTRL) Register

Table 16-26. IRTC\_FILTER23\_CTRL Field Descriptions

Field	Description
15 POL	Tamper Detect Input Bit 2 Polarity Control. This bit controls the polarity of tamper detect input bit 2 (tamper_detect[2]). 0 Tamper detect input bit 2 is active high 1 Tamper detect input bit 2 is active low
14 CLKSEL	Tamper Filter 2 Clock Select. This bit is a write once bit and selects the clock source for tamper filter for tamper detect input bit 2. 0 Clock to tamper filter 2 is 32.768 kHz (Oscillator clock) Tamper filter duration is 45.5 $\mu$ s (i.e. 1.5 clock) to 1.95 ms (64 clocks) in increments of 30.5 $\mu$ s 1 Clock to tamper filter 2 is 512 Hz Tamper filter duration is 2.85 ms (i.e. 1.5 clock) to 125 ms (64 clocks) in increments of 1.95 ms
13:8 FILTER2 DURATION	Tamper Detect Bit 2 Filter Duration. This bit indicates the number of tamper filter clock cycles for which the tamper_detect[2] signal must remain stable before being detected as a tamper. These bits are used by the tamper filtering operation. 0 Filtering operation disabled 1 to 63 number of tamper filter clock cycles to be counted when tamper is asserted With the tamper duration set to 0, any tamper detected on the tamper pins will directly set the tamper status and interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins causes a tamper interrupt.
7:0	Reserved bits. Not writeable. Read returns zeros.

### 16.5.24 IRTC Standby RAM (IRTC\_STDBY\_RAM)

The Standby RAM is both 8-bits and 16-bits accessible and has 16-bit aligned addresses. The ordering used is Big Endian.



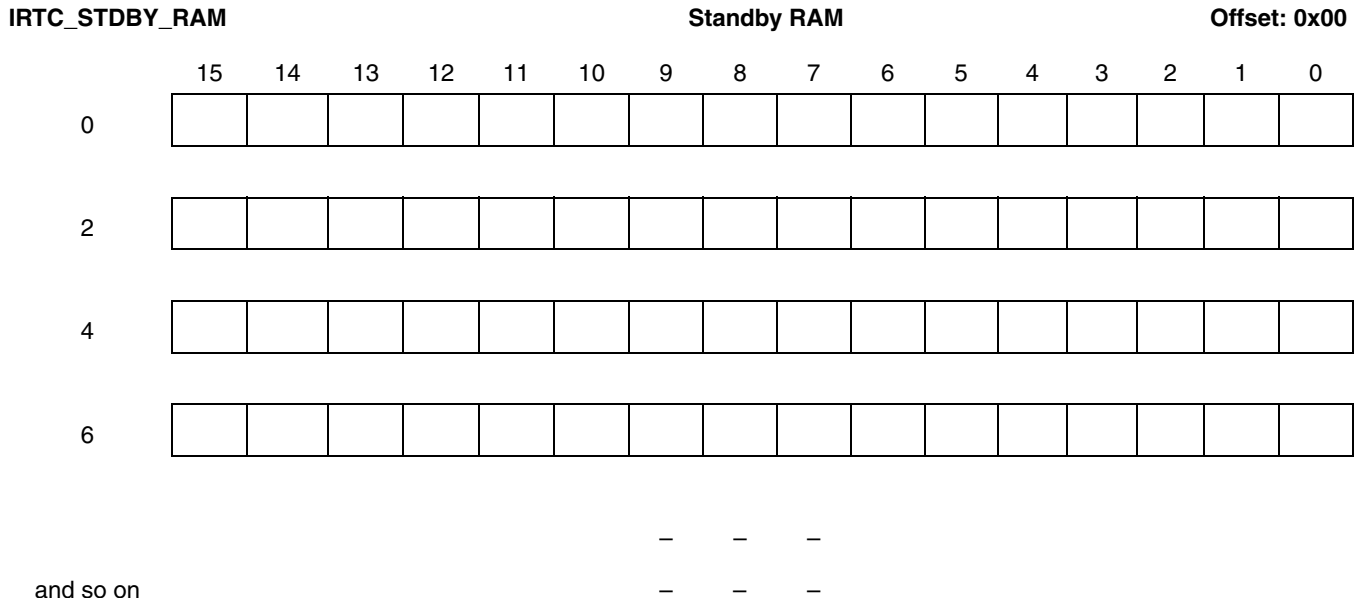


Figure 16-26. IRTC Standby RAM (IRTC\_STDBY\_RAM)

## 16.6 Functional Description

### 16.6.1 Compensation Logic

The compensation logic inside IRTC provides an accurate and wide compensation range, which is suitable for many crystals, and can correct errors as high as 3906 PPM and as low as 0.119 PPM. The same hardware logic supports both temperature compensation and frequency compensation.

To perform temperature compensation, the CPU maintains a look-up table which lists the changes in frequency of crystal for each degree change in temperature. CPU wakes periodically to measure the external temperature via a temperature sensor connected to an A/D converter. CPU uses the look-up table to determine the compensation factor and writes the value to be compensated (in terms of number of IRTC oscillator clock cycles in 2's complement format) in the IRTC compensation register (RTC\_COMPEN). Based on the value written, the hardware add or remove pulses accordingly to adjust the 1 Hz frequency due to variation on temperature.

To perform crystal compensation, the firmware calculates the correction by using crystal characteristics, where the CPU sets the correction factor in the two's complement format in the same IRTC compensation register (IRTC\_COMPEN). Based on the values written in the IRTC\_COMPEN register, the circuit compensates by adding or skipping pulses in the oscillator clock signal.

There are two important components in the compensation algorithm: compensation value and compensation interval. They are defined as:

- **Compensation/Correction Value:** It is a 2's complement value with which the IRTC oscillator clock is modified by adding or removing pulses from it.

- **Compensation Interval:** Compensation interval is the duration in which the correction value is applied. This is the time in which this block adds or removes pulses thereby ensuring that the compensation interval is close to the interval obtained with an ideal 1 Hz clock.

#### Vital statistics:

- Range of compensation interval: 1 second to 255 seconds (0 disables compensation)
- Range of compensation: –128 clocks to 127 clocks (IRTC oscillator clocks)
- Selection criteria: Compensation is done only when enabled by CPU. CPU can disable compensation by programming a zero compensation interval

#### Compensation flow:

The operation starts in an idle state waiting for the firmware to enable compensation. Since the same hardware logic is used for temperature and crystal compensation, the firmware must provide a value that takes into account the correction for both temperature and crystal. When enabled, the compensation cycles are added or removed till the compensation interval expires. On completion of the compensation interval, the done bit is asserted to the CPU and if compensation is still enabled, the next compensation cycle starts. Compensation state machine returns to idle state when compensation is disabled by writing 0 to compensation interval. A newly programmed value is picked only when the current compensation cycle has completed.

Figure 16-27 shows the logical compensation flow of the state machine.

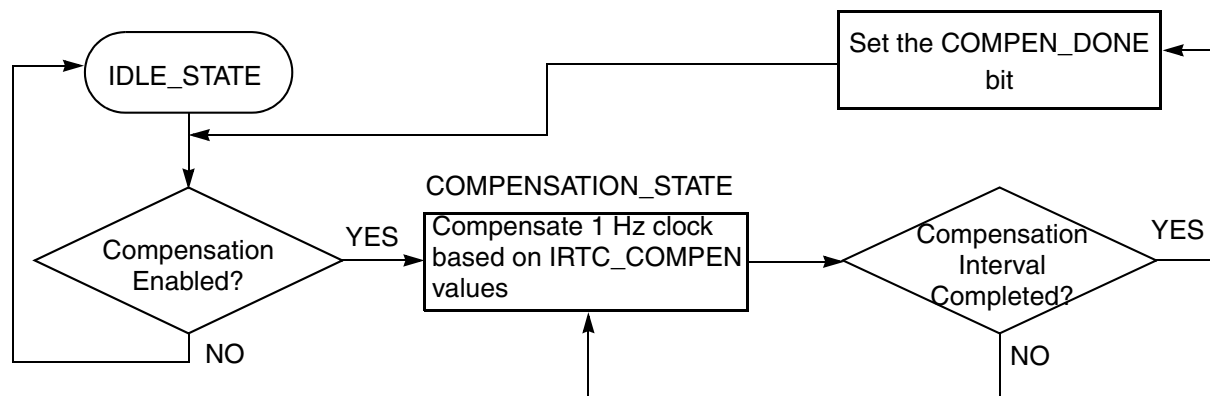


Figure 16-27. Compensation Control Flow

#### Recommendation for Optimal Compensation:

Since the addition and removal of pulses is done in the first second of the compensation interval, the CPU has the option of finding the compensation factor over a period of time and then calculating the correction factor per second and enable compensation hardware every second for uniform 1 Hz clock periods.

### 16.6.2 Write Protection Logic

This logic protects the IRTC registers and Standby RAM from any spurious updates that can happen due run-away code. The logic is based on a state machine that monitors the values written to WE[1:0] bits of the IRTC\_CTRL register. By default unconditional write access is allowed to these bits only.

To enable write protection, “10” is written on these bits. To disable write protection, the sequence “00 – 01 – 11 – 10” is written onto these bits.

After a power on reset, the write-protect mechanism is disabled, allowing the user code to calibrate the IRTC clock, set the time in the clock registers, and set the date in the calendar registers. Once calibration and time & date settings are done, the user code should enable write protection mode. If not, the registers are put into write protect mode 15 seconds after power on. In case the write protect mode is unlocked to update registers, then the write protect mode is enabled 2 seconds after unlock if not done by CPU.

Any write access made to the register space when write protection is enabled (i.e. registers in locked mode) will cause the transfer error signal to be asserted.

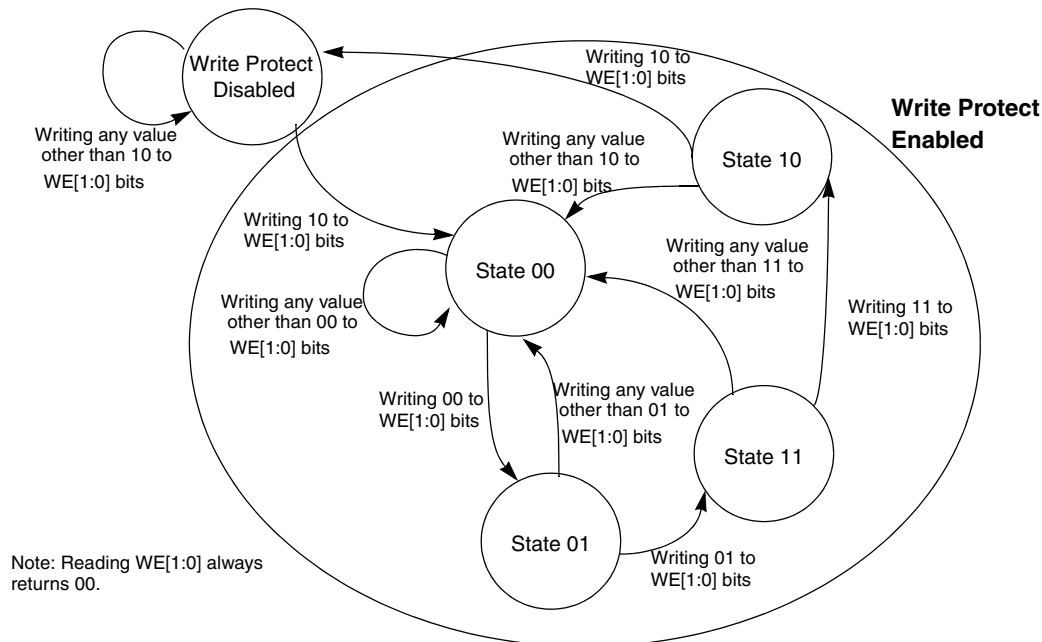


Figure 16-28. Write Protection State Machine

### 16.6.3 Tamper Detection Logic

The tamper detect mechanism is used to detect any intrusion made to system where this block is used. The tamper logic supports up to eight tamper signal inputs which can be used for internal tamper events (i.e. battery removal, etc.) or external tamper events (i.e. off chip tamper switches). IRTC stores these individual tamper events and generates a common interrupt when any tamper event occurs.

#### Internal Tamper Events: Battery removal when MCU is powered OFF:

The battery removal is detected during system power off by using a flop that is asserted on power-on reset. Because there is no difference between a proper shutdown and this tamper condition, this is considered as a tamper always. Now it is up to the firmware to differentiate the tamper condition from normal power up. One way is to ignore this tamper interrupt when the MCU or application is in service mode and simply reset the tamper interrupt. For other conditions it is taken as a tamper.

#### Internal Tamper Events: Battery removal when MCU is powered ON:

The analog circuitry which monitors the battery voltage indicates when the battery voltage is removed. This signal is used by the tamper circuit to indicate a tamper provided MCU power is ON. A flop present will be cleared by CPU during calibration. Any change of state on this signal is detected as a tamper.

### External Tamper Events: Off Chip Tamper Indication:

An external off-chip tamper switch is also used to monitor tampering external to the MCU. For example, this signal can indicate whether the energy meter case is open or not. Again a flop present inside IRTC is cleared by CPU during calibration. Any changes of the states on this signal is detected as a tamper.

A tamper filtering circuit is present on all tamper inputs to prevent any glitches triggering the tamper interrupt. The duration for a tamper signal to be asserted is indicated as tamper. This duration is user programmable from 64  $\mu$ s to 125 ms to support a variety of tamper switches. The filtered signals are then used to generate the tamper status and interrupt signals. The polarity of the tamper inputs can be selected as active high or active low.

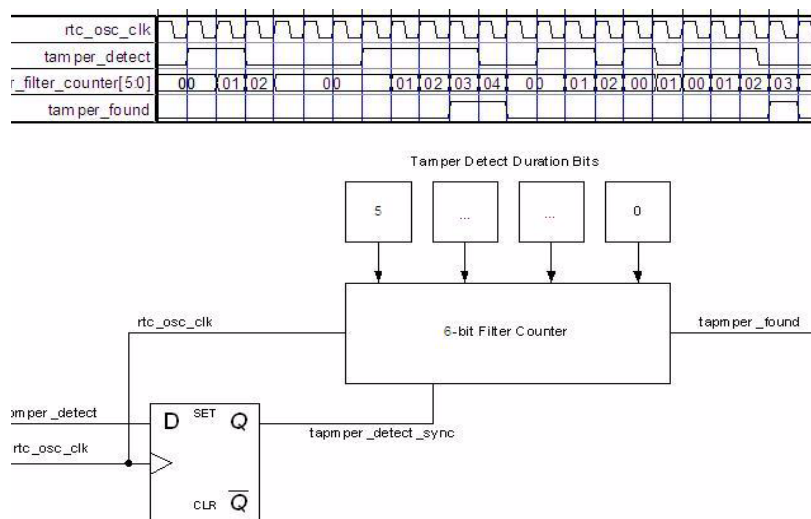


Figure 16-29. External Tamper Capture Circuit and Timing Diagram

### Tamper Detection Flow:

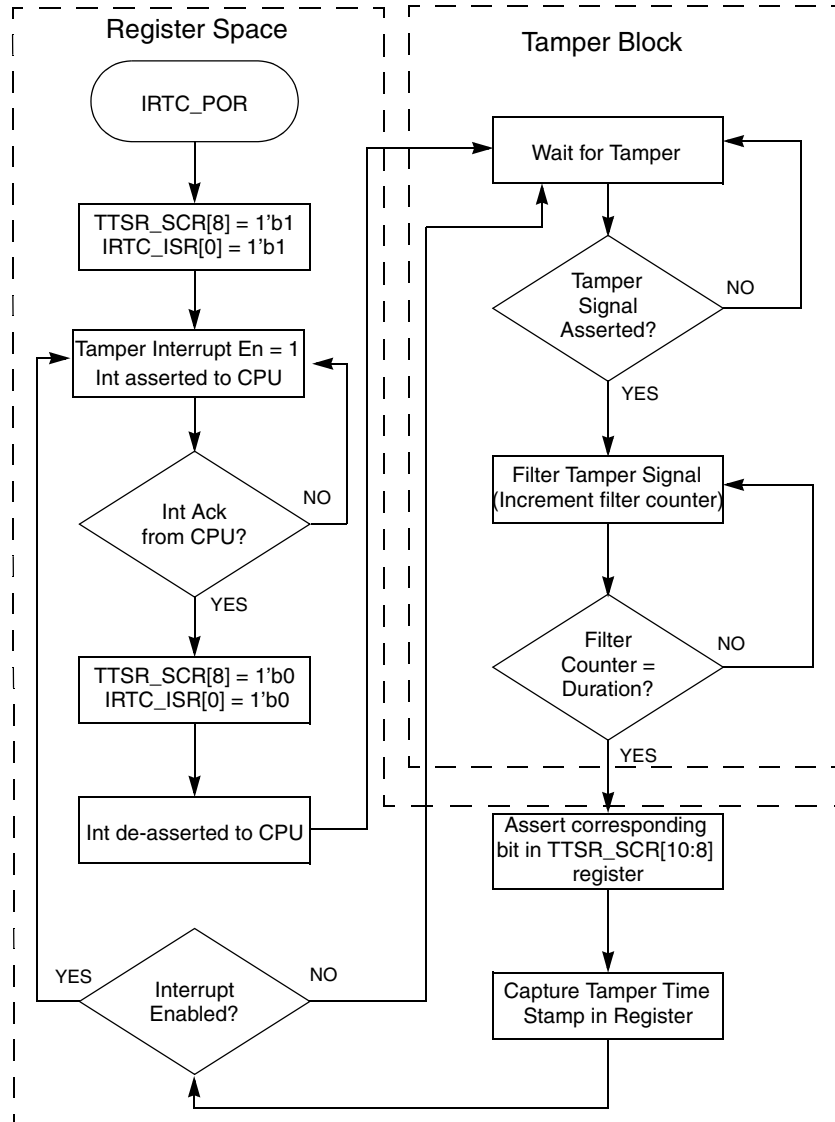


Figure 16-30. Tamper Detection Flow Diagram

**Description:**

- POR asserts tamper status bit 8 in register space (IRTC\_TTSR\_SCR[8]) and tamper interrupt status bit (in IRTC\_ISR).
- Tamper interrupt enable bit is also asserted on POR and an interrupt to CPU is indicated.
- When CPU acknowledges the tamper interrupt by writing 1'b1 to the tamper interrupt status bit (IRTC\_TTSR\_SCR[8]), the tamper status bits are cleared.
- Any tamper signal asserted externally (battery removal or external tamper) are filtered in the tamper block.
- Tamper signal is asserted when the filter counter matches the programmed filter duration.
- Based on the tamper detect signal, the appropriate status bit is asserted in the tamper status register (IRTC\_TTSR\_SCR[10:8]) irrespective the tamper bit is enabled or not.

- The tamper interrupt status bit is asserted irrespective of the assertion of corresponding control bit in IRTC\_TTSR\_SCR[2:0]. Interrupt status bit (IRTC\_ISR[0]) is asserted if corresponding control bit is asserted.
- If interrupt is enabled, CPU is interrupted by asserting the interrupt.
- Tamper status bits (in register space) are cleared when 1 is written to tamper interrupt status bit (in IRTC\_TTSR\_SCR[15:8]).
- Tamper interrupt can be armed or disabled by writing to the interrupt enable register bit (in IRTC\_IER[0]) or the individual tamper control bits (IRTC\_TTSR\_SCR[7:0]) through the Register Programming Interface.

### 16.6.4 IRTC Control Logic

This block is the heart of the IRTC. It performs and controls all chronological functions as below

- Implementing all counters for date, time and countdown timer and their related control logic
- Leap Year calculation and adjusting the day count accordingly
- Incrementing or decrementing of counters for adjustment of leap seconds
- Tracking the number of days in a month
- Automatic Daylight adjustment for time
- Alarm generation with selectable matching of different counters

Date counters are dynamic modified based on leap year, month and/or daylight saving. Additionally, MPU can add or subtract a second to take care of leap seconds. All changes are hardware controlled and triggered by software. A leap year is the year that is divisible by 4 & 400 and has an extra day in February. Daylight savings are applied in regions to shift the local time in the summer and winters. Time is shifted at a pre-defined time decided by MCU and controlled by hardware alarms for daylight saving. Day counter is also adjusted for months of 28, 29, 30, and 31 days.

Alarm is generated by the control block. The match of counters is controlled by ALARM\_BITS inside of the control register to give various alarm options of daily, monthly, yearly or one-time alarms.

### 16.6.5 IRTC Isolation Logic

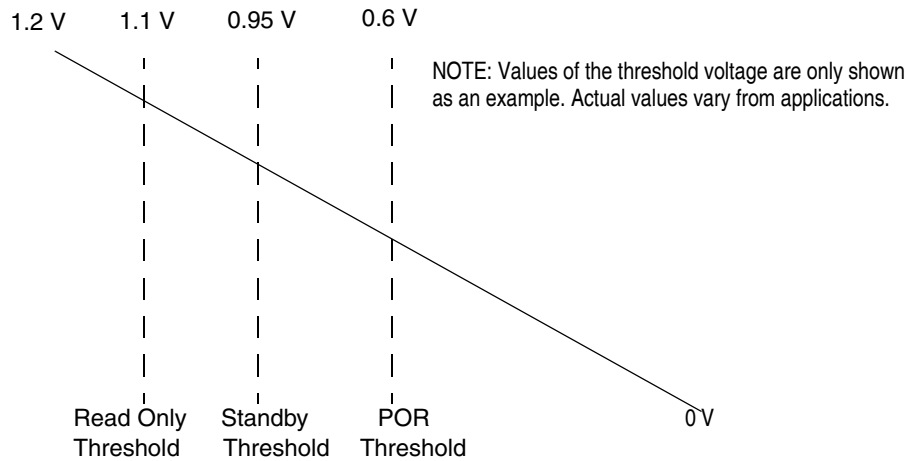
This block provides isolation and protection on signals that are coming from a switchable power domain. The IRTC block has several signals (for e.g. Register Programming Interface) coming from blocks that do not work when the power is removed. Such signals, if not isolated, can cause the IRTC to malfunction and increase the leakage current and hence more power consumption.

There are two levels of isolation provided:

- When system power falls below standby threshold (i.e. voltage below which system cannot operate properly) and battery power becomes operational, all inputs to IRTC are isolated from the rest of the chip.
- When system power falls to an intermediate (read only voltage) threshold (more than standby threshold but less than normal voltage), writes to all registers is blocked. Other signals are functional.

The voltage threshold is detected by an analog block (outside digital IRTC) which monitors the voltage level.

See figure below for threshold example.



**Figure 16-31. Thresholds for which Isolation is required**

## 16.6.6 Battery Power Architecture

Figure 16-32 shows the interaction of IRTC (Digital + Analog) with other MCU clock and power management peripherals.

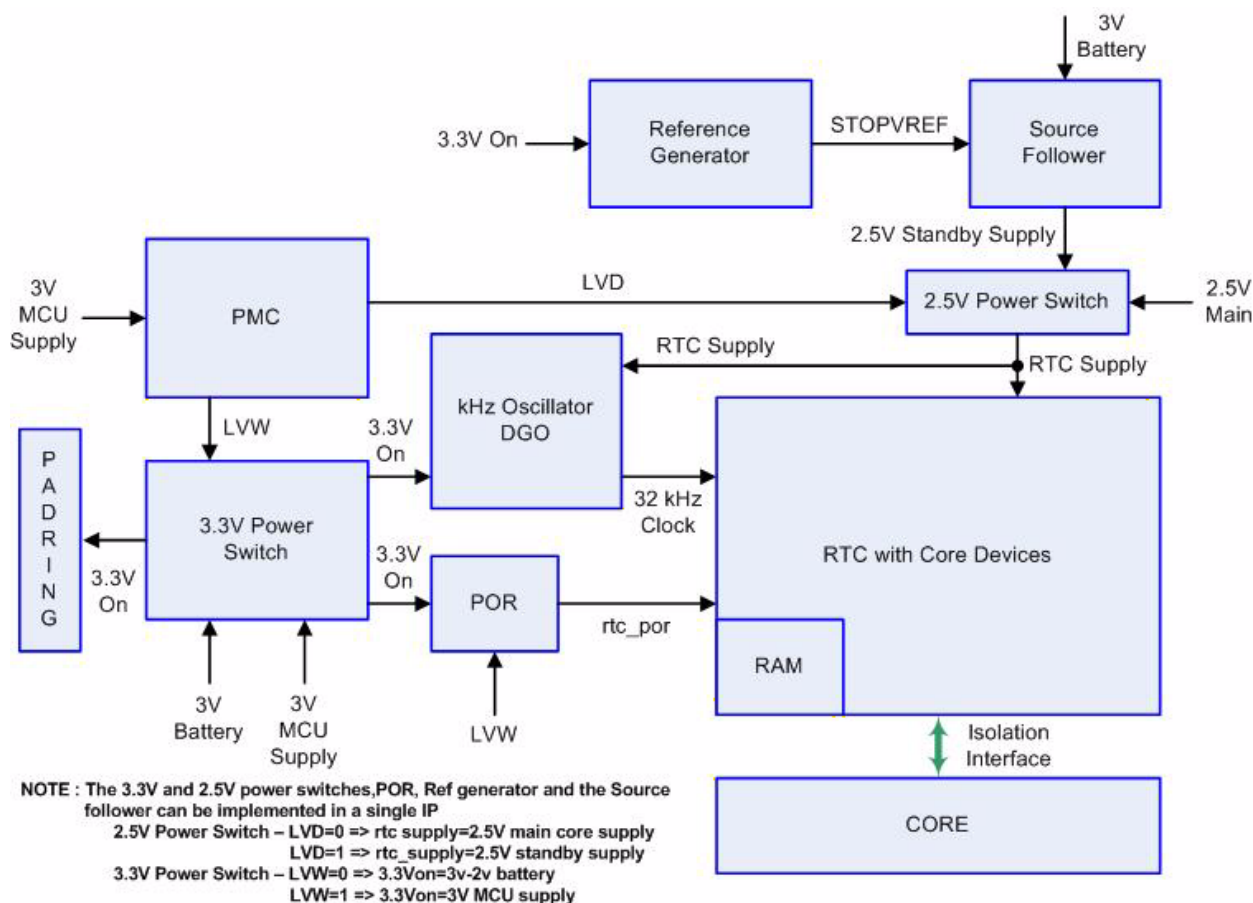


Figure 16-32. Proposed Analog Standby Switching Circuitry

**Design Blocks:**

The analog sub-block of IRTC comprises of the following:

- 2.5V power switch
- 3.3V power switch
- Single power on reset for IRTC digital
- Reference generator
- Expose mode for analog signals
- Level shifting logic and control for PMC interface
- Separate 32 kHz oscillator

**16.7 Design Assumptions**

- When MCU is in power off, no compensation correction value can be programmed. As a result, if the temperature of the system rises so as to change the oscillator clock, correction can only be done when the MCU is next powered ON. Till that time IRTC will loose some accuracy.
- The compensation values (interval and correction figures) are not programmed rapidly and hence will change after very long time.



- Leap seconds are adjusted twice a year and hence it is assumed that there will no consecutive and rapid increments or decrements to the seconds counter.
- In case more than one increment or decrements are done within 1 second interval, it will not guarantee normal operation.
- For all practical purposes, the Bus Clock is asynchronous to the 1 Hz clock used to generate the write protection time-outs. Hence after unlocking the registers, the actual duration of unlock would not be 2 seconds but less than 2 seconds. In order to have complete 2 seconds, the unlocking of registers should be done at the seconds boundary indicated by the 1 Hz interrupt (See [Section 16.5.11, “IRTC Interrupt Status Register \(IRTC\\_ISR\)”](#))
- The first event of any Sampling Timer interrupts after Power on Reset should not used to qualify any periodic interval. However, the correct periodic interval (i.e. 512 Hz or 256 Hz, etc.) should be determined using two sampling timer interrupts. The time between two interrupt would always be the correct time period.
- During Standby Mode, Bus Clock would be isolated and hence we cannot capture the tamper status in tamper registers when a tamper occurs during Standby mode. However, the interrupt status bit and time stamp will get captured. To know about the Tamper Status, CPU can use the following argument. When in Standby mode (i.e. CPU power is OFF), any tamper occurring would ONLY be external tamper (tamper status = “01”) because in case of battery removal tamper (during Standby mode) would lead to a POR thereby setting the status as “11”.



# Chapter 17

## Analog-to-Digital Converter (ADC16V1)

### 17.1 Introduction

The 16-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

#### NOTE

The registers APCTL1, APCTL2, APCTL3 and APCTL4 have no functionality in the MC9S08GW64 series. The pin control function is performed by the Mux control, please refer to [Section 6.7, “Pin Mux Controls,”](#) for further information.

$V_{BG}$  is from the MCU voltage regulator and  $V_{alt}$  is from  $V_{REF}$  module.

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it.

For details on low-power mode operation, refer to [Table 3-4 in Chapter 3 , “Modes of Operation”](#).

#### 17.1.1 ADC Clock Gating

The bus clock to each ADC can be gated on and off using the SCGC1\_ADCx bits (see [Section 5.8.17, “System Clock Gating Control 3 Register \(SCGC3\)”](#)). These bits are cleared after any reset, which disable the bus clock to this module. To conserve power, the SCGC1\_ADCx can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### 17.1.2 Hardware Trigger

The hardware triggers of ADC0 and ADC1 are provided from PDB channel 1 and 2 correspondingly, when ADTRG is set in ADCxSC2.

When enabled, ADCx is triggered each time PDB channel  $x$  TriggerA output is asserted. The PDB channel  $x$  PreTriggerA and PreTriggerB set the ADHWTSA and ADHWTB correspondingly.

For details on ADC hardware trigger function, refer to [Section 17.5.4, “Hardware Trigger and Channel Selects”](#). For details on PDB, refer [Chapter 22, “Programmable Delay Block \(S08PDBV1\).”](#)

### 17.1.3 ADC Alternate Clock

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

## 17.2 ADC Connections

Assignment of device ADC pins is spread over each of the available ADCs as shown in [Table 17-1](#).

**Table 17-1. ADC Channel Assignments**

ADCH	Input function of ADC0		Input function of ADC1	
	DIFFn=0	DIFFn=1	DIFFn=0	DIFFn=1
00000(0)	DADP0	DADP0/DADM0	DADM1	_1
00001(1)	DADM0	–	DADP1	DADP1/DADM1
00010(2)	AD2	–	–	–
00011(3)	–	–	AD3	–
00100(4)	AD4 <sup>2</sup>	–	RESERVED	–
00101(5)	AD5	–	–	–
00110(6)	VLL1	–	VREFH	–
00111(7)	VCAP1	–	VREFL	–
01000(8)	–	–	AD6	–
01001(9)	–	–	AD7	–
01010(10)	VLL2	–	–	–
01011(11)	VCAP2	–	RESERVED	–
01100(12)	AD8	–	RESERVED	–
01101(13)	AD9	–	RESERVED	–
01110(14)	RESERVED	–	RESERVED	–
01111(15)	VREF OUT Internal	–	VREF OUT Internal	–
10000(16)	–	–	AD11	–
10001(17)	PMC VREF 1.2 V	–	AD10	–
10010(18)	–	–	–	–
10011(19)	–	–	–	–
10100(20)	AD12	–	PRACMP0 DAC OUT	–
10101(21)	AD13	–	PRACMP1 DAC OUT	–
10110(22)	–	–	AD15	–

Table 17-1. ADC Channel Assignments (continued)

ADCH	Input function of ADC0		Input function of ADC1	
10111(23)	–	–	AD14	–
11000(24)	–	–	PRACMP2 DAC OUT	–
11001(25)	–	–	–	–
11010(26)	Temperature Sensor	–	Temperature Sensor	–
11011(27)	Bandgap	–	Bandgap	–
11100(28)	–	–	–	–
11101(29)	–	–	–	–
11110(30)	–	–	–	–
11111(31)	–	–	–	–

<sup>1</sup> The unused channels marked with '–' are connected to VREFL.

<sup>2</sup> Users must not select AD4–AD15 as a channel if  $V_{LL3} > V_{DDA}$ .

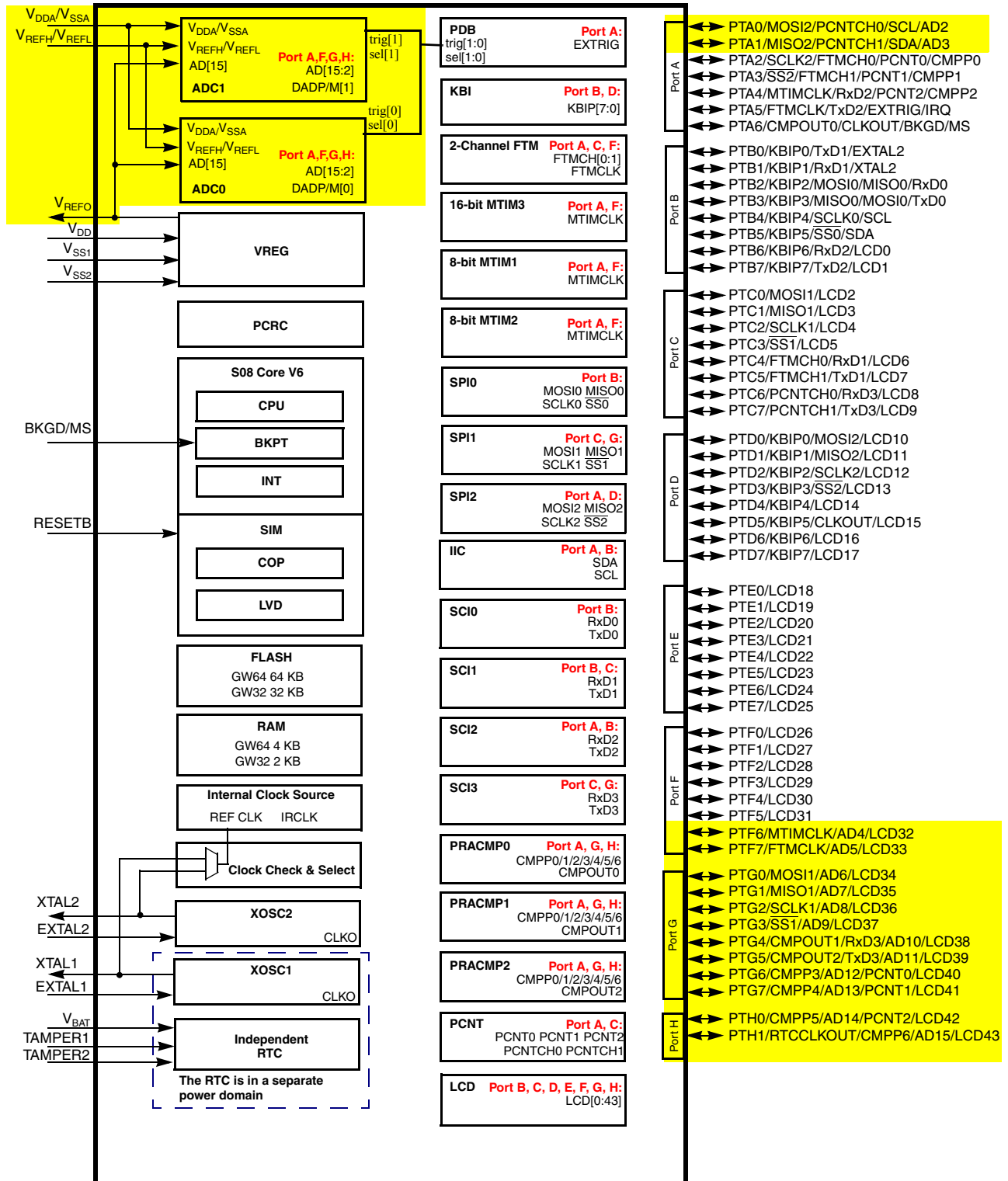


Figure 17-1. MC9S08GW64 Series Block Diagram Highlighting ADC Modules and Pins

## 17.2.1 Features

Features of the ADC module include:

- Linear successive approximation algorithm with up to 16-bit resolution
- Up to 4 pairs of differential and 24 single-ended external analog inputs
- Output Modes: Differential 16-bit, 13-bit, 11-bit and 9-bit modes, or single-ended 16-bit, 12-bit, 10-bit and 8-bit modes
- Output formatted in 2's complement 16b sign extended for differential modes
- Output in right-justified unsigned format for single-ended
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete / Hardware average 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 with option to output the clock
- Selectable asynchronous hardware conversion trigger with hardware channel select
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- Temperature sensor
- Hardware average function
- Selectable voltage reference, Internal, External, or Alternate
- Self-Calibration mode

## 17.2.2 Block Diagram

[Figure 17-2](#) provides a block diagram of the ADC module.

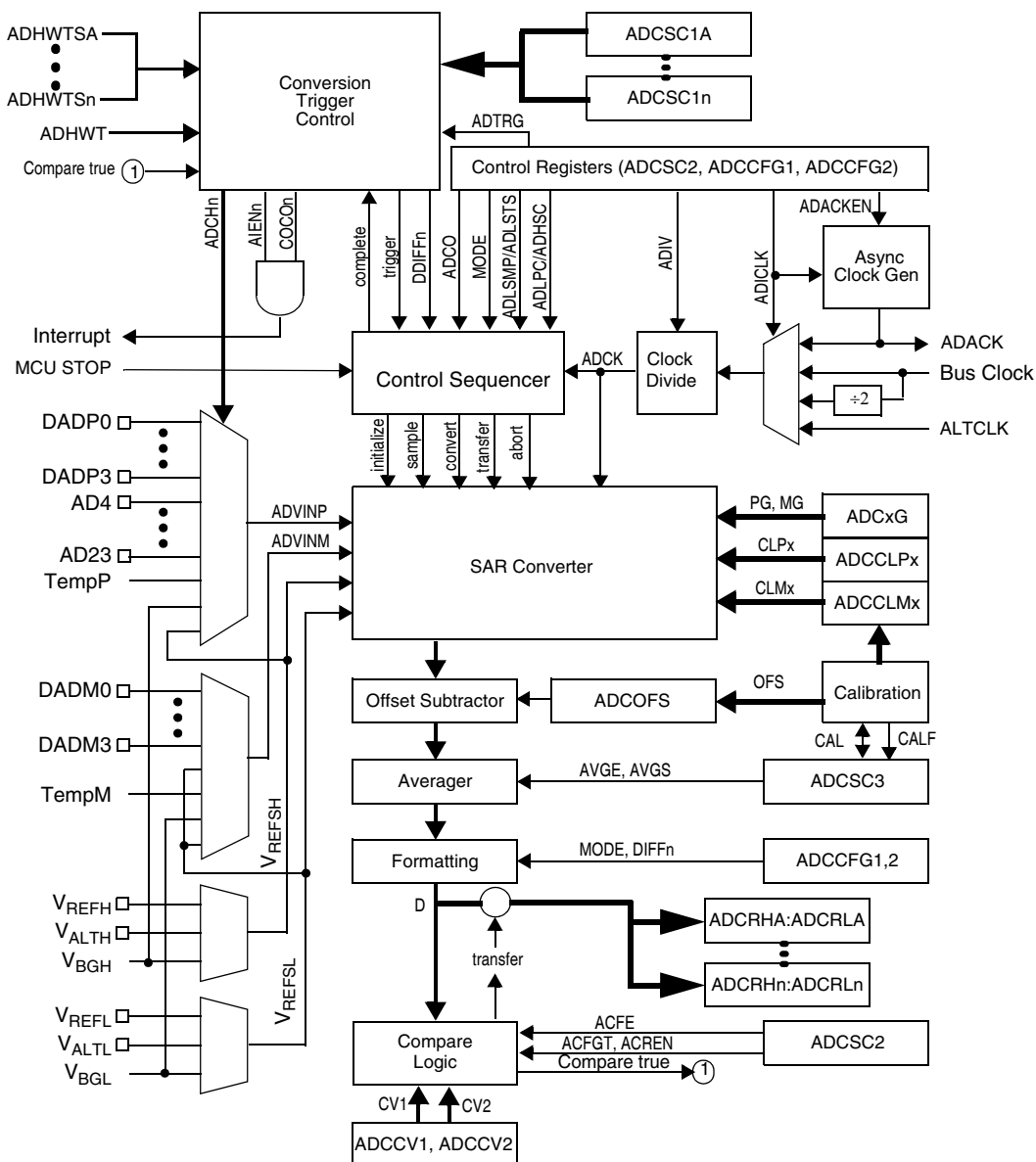


Figure 17-2. ADC Block Diagram

## 17.3 External Signal Description

The ADC module supports up to 4 pairs of differential inputs and 24 single-ended inputs. Each differential pair requires two inputs, DADPx and DADMx. The ADC also requires four supply/reference/ground connections.

Table 17-2. Signal Properties

Name	Function
DADP0-DADP3	Differential Analog Channel Inputs
DADM0-DADM3	Differential Analog Channel Inputs



**Table 17-2. Signal Properties**

Name	Function
AD4–AD23	Analog Channel inputs
$V_{REFSH}$	Voltage Reference Select High
$V_{REFSL}$	Voltage Reference Select Low
$V_{DDA}$	Analog power supply
$V_{SSA}$	Analog ground

### 17.3.1 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.

### 17.3.2 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  $V_{SS}$ .

### 17.3.3 Voltage Reference Select High ( $V_{REFSH}$ )

$V_{REFSH}$  is the high reference voltage for the converter.

The ADC can be configured to accept one of three voltage reference pairs for  $V_{REFSH}$ . Each pair contains a positive reference which must be between the minimum Ref Voltage High (defined in Appendix A) and  $V_{DDA}$ , and a ground reference which must be at the same potential as  $V_{SSA}$ . The three pairs are external ( $V_{REFH}$  and  $V_{REFL}$ ), alternate ( $V_{ALTH}$  and  $V_{ALTTL}$ ) and the internal bandgap ( $V_{BGH}$  and  $V_{BGL}$ ). These voltage references are selected using the REFSEL bits in the ADCSC2 register. The alternate ( $V_{ALTH}$  and  $V_{ALTTL}$ ) voltage reference pair may select additional external pins or internal sources depending on MCU configuration. Consult the module introduction for information on the Voltage References specific to this MCU.

In some packages,  $V_{REFH}$  is connected in the package to  $V_{DDA}$ . If externally available, the positive reference(s) may be connected to the same potential as  $V_{DDA}$  or may be driven by an external source to a level between the minimum Ref Voltage High (defined in Appendix A) and the  $V_{DDA}$  potential ( $V_{REFH}$  must never exceed  $V_{DDA}$ ).

### 17.3.4 Voltage Reference Select Low ( $V_{REFL}$ )

$V_{REFSL}$  is the low reference voltage for the converter. The ADC can be configured to accept one of three voltage reference pairs for  $V_{REFSL}$ . Each pair contains a positive reference which must be between the minimum Ref Voltage High (defined in Appendix A) and  $V_{DDA}$ , and a ground reference which must be at the same potential as  $V_{SSA}$ . The three pairs are external ( $V_{REFH}$  and  $V_{REFL}$ ), alternate ( $V_{ALTH}$  and  $V_{ALTTL}$ ) and the internal bandgap ( $V_{BGH}$  and  $V_{BGL}$ ). These voltage references are selected using the REFSEL bits. The alternate ( $V_{ALTH}$  and  $V_{ALTTL}$ ) voltage reference pair may select additional external pins or internal

sources depending on MCU configuration. Consult the module introduction for information on the Voltage References specific to this MCU.

In some packages,  $V_{REFL}$  is connected in the package to  $V_{SSA}$ . If externally available, connect the ground reference(s) to the same voltage potential as  $V_{SSA}$ .

### 17.3.5 Analog Channel Inputs (ADx)

The ADC module supports up to 24 single-ended analog inputs. A single-ended input is selected for conversion through the ADCHn channel select bits when the DIFFn bit in the ADCSC1n register is low.

### 17.3.6 Differential Analog Channel Inputs (DADx)

The ADC module supports up to 4 differential analog channel inputs. Each differential analog input is a pair of external pins (DADPx and DADMx) referenced to each other to provide the most accurate analog to digital readings. A differential input is selected for conversion through the ADCHn channel select bits when the DIFFn bit in the ADCSC1n register bit is high.

## 17.4 Register Definition

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

- Status and channel control registers, ADCSC1A:ADCSC1n
- Configuration registers, ADCCFG1 and ADCCFG2
- Data result registers, ADCRHA:ADCRLA to ADCRHn:ADCRLn
- Compare value registers, ADCCV1H, ADCCV1L, ADCCV2H, and ADCCV2L
- General status and control registers, ADCSC2 and ADCSC3
- Configuration registers, ADCCFG1 and ADCCFG2
- Offset Correction Registers, ADCOFSH and ADCOFSL
- Plus-input gain registers, ADCPGH and ADCPGL
- Minus-input gain registers, ADCMGH and ADCMGL
- Plus-side general calibration registers, ADCCLP0, ADCCLP1, ADCCLP2, ADCCLP3H, ADCCLP3L, ADCCLP4H, ADCCLP4L, ADCCLSP, ADCCLDP
- Minus-side general calibration registers, ADCCLM0, ADCCLM1, ADCCLM2, ADCCLM3H, ADCCLM3L, ADCCLM4H, ADCCLM4L, ADCCLSM, ADCCLDM
- Pin enable registers, APCTL1, APCTL2, APCTL3, and APCTL4

### 17.4.1 Status and Control Registers 1 (ADCSC1A:ADCSC1n)

This section describes the function of the ADC status and channel control registers, ADCSC1A through ADCSC1n. ADCSC1A is used for both software and hardware trigger modes of operation. ADCSC1B -ADCSC1n indicate potentially multiple ADCSC1 registers for use only in hardware trigger mode. Consult the module introduction for information on the number of ADCSC1n registers specific to this MCU. The ADCSC1A to ADCSC1n registers have identical fields, and are used in a “ping-pong”

approach to control ADC operation. At any one point in time, only one of the ADCSC1A to ADCSC1n registers is actively controlling ADC conversions. Updating ADCSC1A while ADCSC1n is actively controlling a conversion is allowed (and vice-versa for any of the ADCSC1n registers specific to this MCU). Writing ADCSC1A while ADCSC1A is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), writes to ADCSC1A subsequently initiates a new conversion (if the ADCHn bits are equal to a value other than all 1s). Similarly, writing any of the ADCSC1n registers while that specific ADCSC1n register is actively controlling a conversion aborts the current conversion. Any of the ADCSC1B -ADCSC1n registers are not used for software trigger operation and therefore writes to the ADCSC1B -ADCSC1n registers do not initiate a new conversion.



**Figure 17-3. Status and Channel Control Register 1n (ADCSC1n)**

**Table 17-3. ADCSC1:ADCSC1n Field Descriptions**

Field	Description
7 COCOn	<b>Conversion Complete Flag</b> - The COCOn flag is a read-only bit that is set each time a conversion is completed when the compare function is disabled (ACFE=0) and the hardware average function is disabled (AVGE=0). When the compare function is enabled (ACFE=1), the COCOn flag is set upon completion of a conversion only if the compare result is true. When the hardware average function is enabled (AVGE=1), the COCOn flag is set upon completion of the selected number of conversions (determined by the AVGS bits). The COCOn flag will also set at the completion of a Calibration sequence. The COCOn bit is cleared when the respective ADCSCn is written or when the respective ADCRLn is read. 0 Conversion not completed 1 Conversion completed
6 AIENn	<b>Interrupt Enable</b> - AIENn enables conversion complete interrupts. When COCOn becomes set while the respective AIENn is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 DIFFn	<b>Differential Mode Enable</b> - DIFFn configures the ADC to operate in differential mode. When enabled this mode automatically selects from the differential channels, changes the conversion algorithm and the number of cycles to complete a conversion. 0 Single-ended conversions and input channels are selected 1 Differential conversions and input channels are selected
4:0 ADCHn	<b>Input Channel Select</b> - The ADCHn bits form a 5-bit field that selects one of the input channels. The input channel decode is dependent upon the value of the DIFFn bit as detailed in <a href="#">Table 17-4</a> . The successive approximation converter subsystem is turned off when the channel select bits are all set (ADCHn = 11111). 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 17-4. Input Channel Select

ADCHn	Input Selected when DIFFn=0	Input Selected when DIFFn=1
00000–00011	DADP0-DADP3	DAD0-DAD3 <sup>1</sup>
00100-10111	AD4-AD23	Reserved
11000-11001	Reserved	Reserved
11010	Temp Sensor (single-ended)	Temp Sensor (differential)
11011	Bandgap (single-ended)	Bandgap (differential)
11100	Reserved	Reserved
11101	$V_{REFSH}$ <sup>2</sup>	$-V_{REFSH}$ <sup>2</sup> (differential)
11110	$V_{REFSL}$ <sup>2</sup>	Reserved
11111	Module disabled	

<sup>1</sup> DAD0-DAD3 are associated with the input pin pairs DADPx and DADMx.

<sup>2</sup> Voltage Reference selected is determined by the REFSEL bits in the ADCSC2 register. Refer to [Section 17.5.3](#) for more information on voltage reference selection.

## 17.4.2 Configuration Register 1(ADCCFG1)

ADCCFG1 selects the mode of operation, clock source, clock divide, and configure for low power or long sample time.

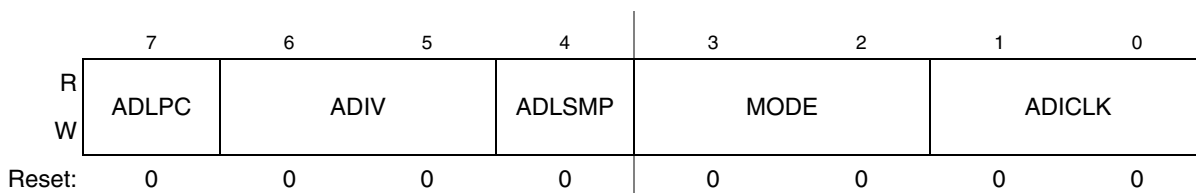


Figure 17-4. Configuration Register (ADCCFG1)

Table 17-5. ADCCFG1 Register Field Descriptions

Field	Description
7 ADLPC	<b>Low-Power Configuration</b> - ADLPC controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 Normal power configuration 1 Low power configuration: The power is reduced at the expense of maximum clock speed.
6:5 ADIV	<b>Clock Divide Select</b> - ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. <a href="#">Table 17-6</a> shows the available clock configurations.
4 ADLSMP	<b>Sample Time Configuration</b> - ADLSMP selects between different sample times based on the conversion mode selected. This bit 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. When ADLSMP=1, the Long Sample Time Select bits (ADLSTS[1:0]) can select the extent of the long sample time. 0 Short sample time 1 Long sample time (The ADLTS bits can select the extent of the long sample time)

**Table 17-5. ADCCFG1 Register Field Descriptions (continued)**

Field	Description
3:2 MODE	<b>Conversion Mode Selection</b> - MODE bits are used to select between the ADC resolution mode. See <a href="#">Table 17-7</a> .
1:0 ADICLK	<b>Input Clock Select</b> - ADICLK bits select the input clock source to generate the internal clock ADCK. See <a href="#">Table 17-8</a> .

**Table 17-6. Clock Divide Select**

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

**Table 17-7. Conversion Modes**

MODE	DIFFn	Conversion Mode Description
00	0	single-ended 8-bit conversion
00	1	Differential 9-bit conversion with 2s complement output
01	0	single-ended 12-bit conversion
01	1	Differential 13-bit conversion with 2s complement output
10	0	single-ended 10-bit conversion
10	1	Differential 11-bit conversion with 2s complement output
11	0	single-ended 16-bit conversion
11	1	Differential 16-bit conversion with 2s complement output

**Table 17-8. Input Clock Select**

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

### 17.4.3 Configuration Register 2 (ADCCFG2)

ADCCFG2 selects differential mode, the special high speed configuration for very high speed conversions, and selects the long sample time duration during long sample mode.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
W								
Reset:	0	0	0	0	0	0	0	0

Figure 17-5. Configuration Register 2(ADCCFG2)

Table 17-9. ADCCFG2 Register Field Descriptions

Field	Description
3 ADACKEN	<b>Asynchronous clock output enable</b> - ADACKEN enables the ADC's asynchronous clock source and the clock source output regardless of the conversion and input clock select (ADICLK bits) status of the ADC. Based on MCU configuration the asynchronous clock may be used by other modules (see module introduction section). Setting this bit allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced since the ADACK clock is already operational. 0 Asynchronous clock output disabled; Asynchronous clock only enabled if selected by ADICLK and a conversion is active 1 Asynchronous clock and clock output enabled regardless of the state of the ADC
2 ADHSC	<b>High Speed Configuration</b> - ADHSC configures the ADC for very high speed operation. The conversion sequence is altered (4 ADCK cycles added to the conversion time) to allow higher speed conversion clocks. 0 Normal conversion sequence selected 1 High speed conversion sequence selected (4 additional ADCK cycles to total conversion time)
1:0 ADLSTS	<b>Long Sample Time Select</b> - ADLSTS selects between the extended sample times when long sample time is selected (ADLSMP=1). This allows 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. 00 Default longest sample time (20 extra ADCK cycles; 24 ADCK cycles total) 01 12 extra ADCK cycles; 16 ADCK cycles total sample time 10 6 extra ADCK cycles; 10 ADCK cycles total sample time 11 2 extra ADCK cycles; 6 ADCK cycles total sample time

### 17.4.4 Data Result Registers (ADCRHA:ADCRLA to ADCRHn:ADCRLn)

The Data Result Registers (ADCRHA:ADCRLA to ADCRHn:ADCRLn) contain the result of an ADC conversion of the channel selected by the respective status and channel control register (ADCSC1A:ADCSC1n). For every ADCSC1A:ADCSC1n status and channel control register, there is a respective ADCRHA:ADCRLA to ADCRHn:ADCRLn data result register. Consult the module introduction for information on the number of ADCRHn:ADCRLn registers specific to this MCU. Reading ADCRHn prevents the ADC from transferring subsequent conversion results into the result registers until ADCRLn is read. If ADCRLn is not read until after the next conversion is completed, the intermediate conversion result is lost. In 8-bit single-ended mode, there is no interlocking with ADCRLn.

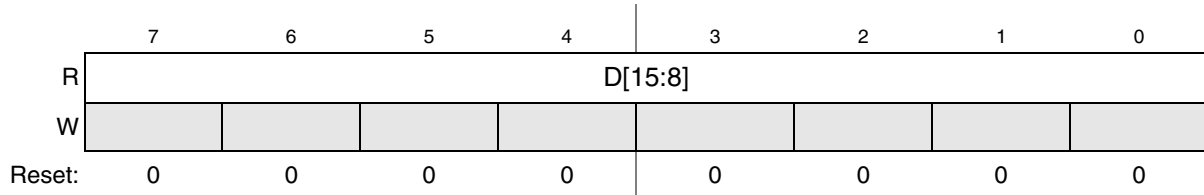


Figure 17-6. Data Result High Register (ADCRHn)

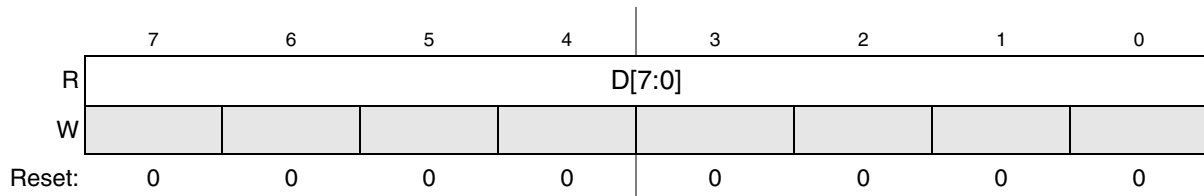


Figure 17-7. Data Result Low Register (ADCRLn)

ADCRHn contains the upper bits of the result of a conversion based on the conversion mode. ADCRLn contains the lower eight bits of the result of a conversion, or all eight bits of an 8-bit single-ended conversion. Unused bits in the ADCRHn register are cleared in unsigned right justified modes and carry the sign bit (MSB) in sign extended 2's complement modes. For example when configured for 10-bit single-ended mode, D[15:10] are cleared. When configured for 11-bit differential mode, D[15:10] carry the sign bit (bit 10 extended through bit 15).

Table 17-10 below describes the behavior of the data result registers in the different modes of operation.

**Table 17-10. Data Result Register Description**

Conversion Mode	Data Result Register bits																Format
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
16b differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	signed 2's complement
16b single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
13b differential	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	sign extended 2's complement
12b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
11b differential	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	sign extended 2's complement
10b single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
9b differential	S	S	S	S	S	S	S	S	D	D	D	D	D	D	D	D	sign extended 2's complement
8b single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

S: Sign bit or sign bit extension.

D: Data (2's complement data if indicated).

## 17.4.5 Compare Value Registers (ADCCV1H:ADCCV1L & ADCCV2H:ADCCV2L)

The Compare Value Registers (ADCCV1H:ADCCV1L & ADCCV2H:ADCCV2L) contain a compare value used to compare with the conversion result when the compare function is enabled (ACFE=1). This register is formatted the same for both bit position definition and value format (unsigned or sign-extended 2's complement) as the Data Result Registers (ADCRHn:ADCRLn) in the different modes of operation (See Table 17-10). Therefore, the compare function only uses the compare value register bits that are related to the ADC mode of operation.

The compare value 2 registers (ADCCV2H:ADCCV2L) are utilized only when the compare range function is enabled (ACREN=1).

In all modes except 8-bit single-ended conversions, the ADCCV1H register holds the upper bits of the first compare value. In 8-bit single-ended mode, ADCCV1H is not used during compare. In all conversion modes, the ADCCV1L register holds the lower 8 bits of the first compare value. The compare function is further detailed in Section 17.5.6.



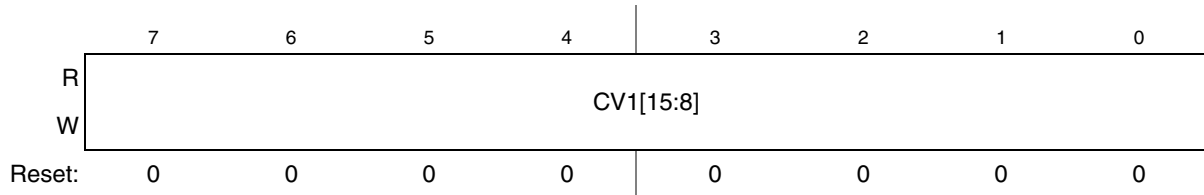


Figure 17-8. Compare Value 1 High Register (ADCCV1H)

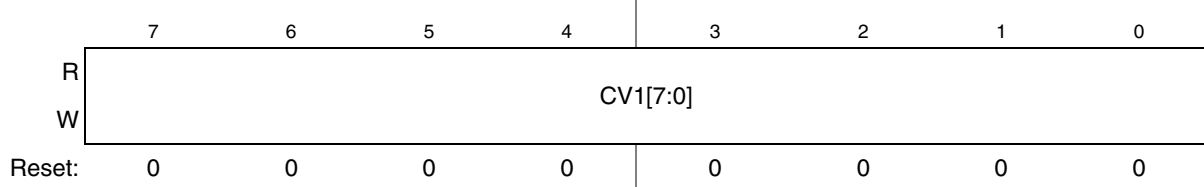


Figure 17-9. Compare Value 1 Low Register(ADCCV1L)

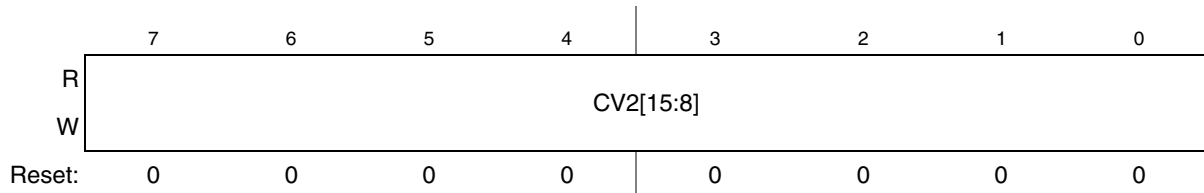


Figure 17-10. Compare Value 2 High Register (ADCCV2H)

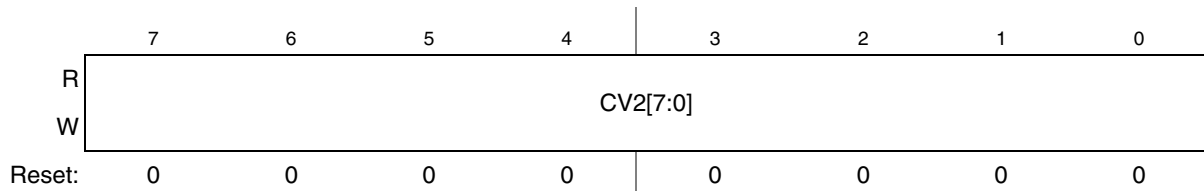


Figure 17-11. Compare Value 2Low Register(ADCCV2L)

### 17.4.6 Status and Control Register 2 (ADCSC2)

The ADCSC2 register contains the conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.

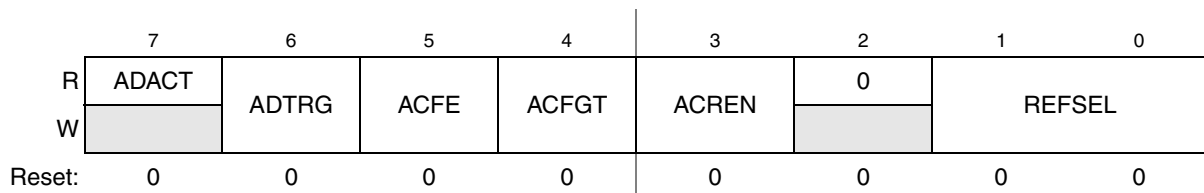


Figure 17-12. Status and Control Register 2 (ADCSC2)

Table 17-11. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	<b>Conversion Active</b> - ADACT indicates that a conversion or hardware averaging is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	<b>Conversion Trigger Select</b> - ADTRG selects the type of trigger used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1A. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input after a pulse of the ADHWTSn input. Refer to <a href="#">Section 17.5.5.1</a> for more information on initiating conversions. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	<b>Compare Function Enable</b> - ACFE enables the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	<b>Compare Function Greater Than Enable</b> - ACFGT configures the compare function to check the conversion result relative to the compare value register(s) (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) based upon the value of ACREN. The ACFE bit must be set for ACFGT to have any effect. The compare function modes are further detailed in <a href="#">Table 17-25</a> in <a href="#">Section 17.5.6</a> . 0 Configures Less Than Threshold, Outside Range Not Inclusive and Inside Range Not Inclusive functionality based on the values placed in the ADCCV1 and ADCCV2 registers. 1 Configures Greater Than Or Equal To Threshold, Outside Range Inclusive and Inside Range Inclusive functionality based on the values placed in the ADCCV1 and ADCCV2 registers.
3 ACREN	<b>Compare Function Range Enable</b> - ACREN configures the compare function to check the conversion result of the input being monitored is either between or outside the range formed by the compare value registers (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) determined by the value of ACFGT. The ACFE bit must be set for ACFGT to have any effect. The compare function modes are further detailed in <a href="#">Table 17-25</a> in <a href="#">Section 17.5.6</a> . 0 Range function disabled. Only the compare value 1 register (ADCCV1H:ADCCV1L) is compared. 1 Range function enabled. Both compare value registers (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) are compared.
1:0 REFSEL	<b>Voltage Reference Selection</b> - REFSEL bits select the voltage reference source used for conversions. Refer to <a href="#">Section 17.5.3</a> for more information on voltage reference selection. 00 Default voltage reference pin pair (External pins $V_{REFH}$ and $V_{REFL}$ ). 01 Alternate reference pair ( $V_{ALTH}$ and $V_{ALTl}$ ). This pair may be additional external pins or internal sources depending on MCU configuration. Consult the module introduction for information on the Voltage Reference specific to this MCU. 10 Internal bandgap reference and associated ground reference ( $V_{BGH}$ and $V_{BGL}$ ). 11 Reserved - Selects default voltage reference ( $V_{REFH}$ and $V_{REFL}$ ) pin pair.

## 17.4.7 Status and Control Register 3 (ADCSC3)

The ADCSC3 register controls the calibration, continuous convert and hardware averaging functions of the ADC module.

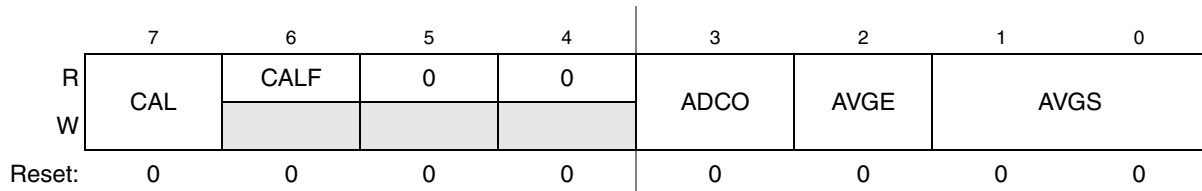


Figure 17-13. Status and Control Register 3 (ADCSC3)

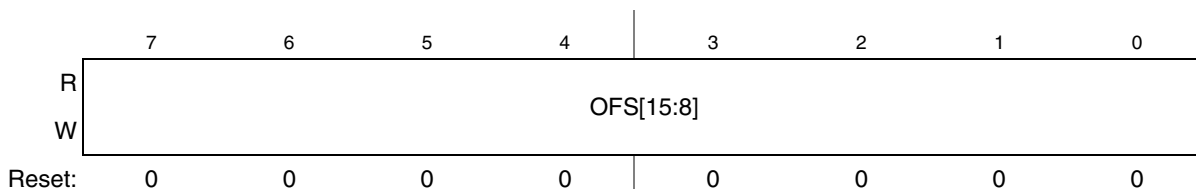
Table 17-12. ADCSC3 Register Field Descriptions

Field	Description
7 CAL	<b>Calibration</b> - CAL begins the calibration sequence when set. This bit stays set while the calibration is in progress and is cleared when the calibration sequence is complete. The CALF bit must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and the CALF bit will set. Setting the CAL bit will abort any current conversion.
6 CALF	<b>Calibration Failed Flag</b> - CALF displays the result of the calibration sequence. The calibration sequence will fail if ADTRG = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes. The CALF bit is cleared by writing a 1 to this bit. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
3 ADCO	<b>Continuous Conversion Enable</b> - ADCO enables continuous conversions. Refer to <a href="#">Section 17.5.5.1</a> for more information on initiating conversions. 0 One conversion or one set of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion.
2 AVGE	<b>Hardware average enable</b> - AVGE enables the hardware average function of the ADC. 0 Hardware average function disabled 1 Hardware average function enabled
1:0 AVGS	<b>Hardware Average select</b> - AVGS determine how many ADC conversions will be averaged to create the ADC average result. 00 - 4 Samples averaged 01 - 8 Samples averaged 10 - 16 Samples averaged 11 - 32 Samples averaged

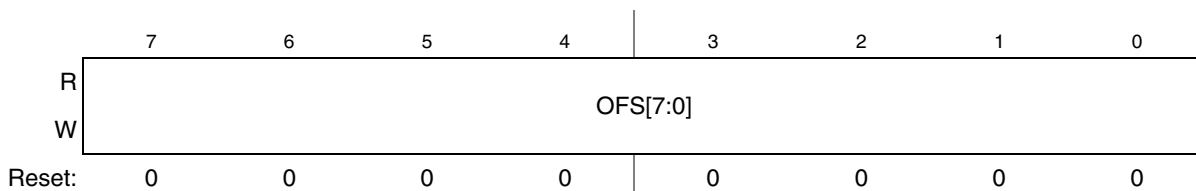
## 17.4.8 ADC Offset Correction Register (ADCOFSH:ADCOFSL)

The ADC Offset Correction Register (ADCOFSH:ADCOFSL) contains the user selected or calibration generated offset error correction value. This register is a 2's complement, left justified, 16b value formed by the concatenation of ADCOFSH and ADCOFSL. The value in the offset correction registers (ADCOFSH:ADCOFSL) is subtracted from the conversion and the result is transferred into the result registers (ADCRHn:ADCRLn). If the result is above the maximum or below the minimum result value, it

is forced to the appropriate limit for the current mode of operation. For additional information please see [Section 17.5.8](#)



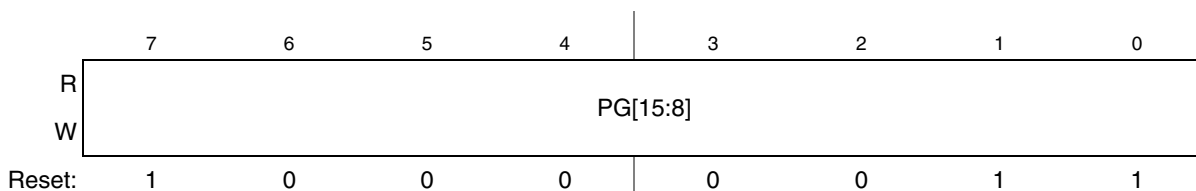
**Figure 17-14. Offset Calibration High Register (ADCOFSH)**



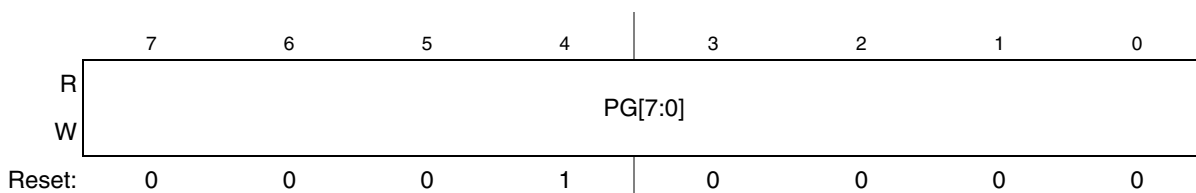
**Figure 17-15. Offset Calibration Low Register (ADCOFSL)**

### 17.4.9 ADC Plus-Side Gain Register (ADCPGH:ADCPGL)

The Plus-Side Gain Register (ADCPGH:ADCPGL) contains the gain error correction for the plus-side input in differential mode or the overall conversion in single-ended mode. ADCPGH:ADCPGL represent a 16 bit floating point number representation of the gain adjustment factor, with the decimal point fixed between ADPG15 and ADPG14. This register must be written by the user with the value described in the calibration procedure or the gain error specifications may not be met.



**Figure 17-16. ADC Plus Gain High Register (ADCPGH)**



**Figure 17-17. ADC Plus Gain Low Register (ADCPGL)**

### 17.4.10 ADC Minus-Side Gain Register (ADCMGH:ADCMGL)

The Minus-Side Gain Register (ADCMGH:ADCMGL) contains the gain error correction for the minus-side input in differential mode. This register is ignored in single-ended mode. ADCMGH:ADCMGL represent a 16 bit floating point number representation of the gain adjustment

factor, with the decimal point fixed between ADPG15 and ADPG14. This register must be written by the user with the value described in the calibration procedure or the gain error specifications may not be met.

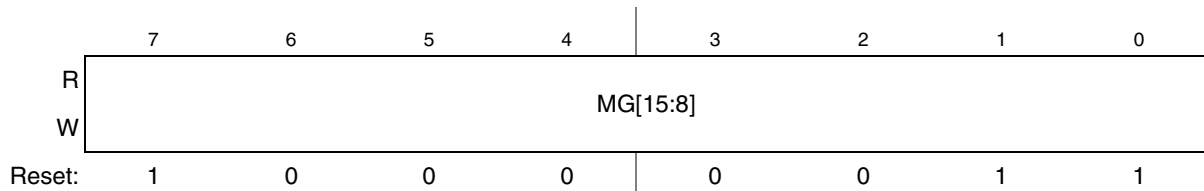


Figure 17-18. ADC Gain Register (ADCMGH)

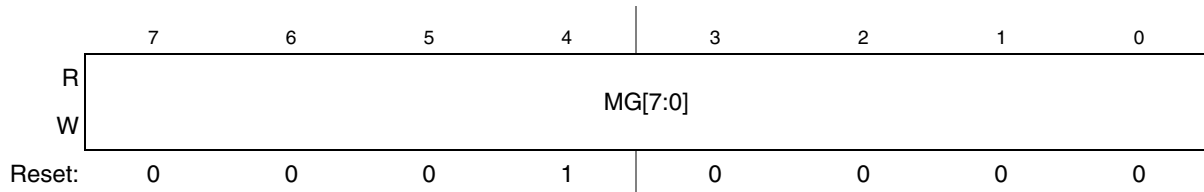


Figure 17-19. ADC Gain Register (ADCMGL)

### 17.4.11 ADC Plus-Side General Calibration Value Registers (ADCCLPx)

The Plus-Side General Calibration Value Registers (ADCCLPx) contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLP0[5:0], CLP1[6:0], CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. ADCCLPx are automatically set once the self calibration sequence is done (CAL is cleared). If these registers are written by the user after calibration, the linearity error specifications may not be met.



Figure 17-20. Plus-Side General Calibration Register (ADCCLPD)



Figure 17-21. Plus-Side General Calibration Register (ADCCLPS)

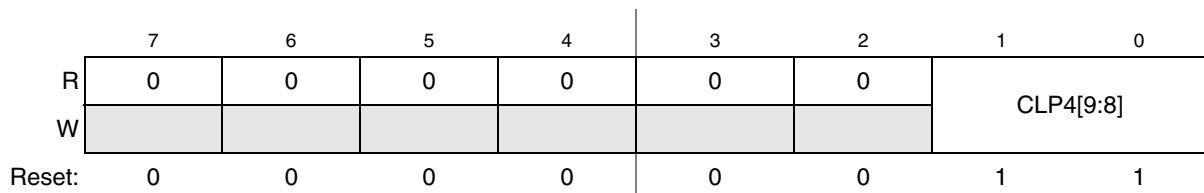


Figure 17-22. Plus-Side General Calibration Register (ADCCLP4H)

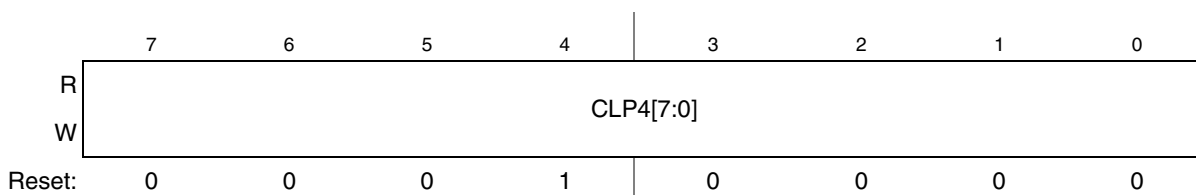


Figure 17-23. Plus-Side General Calibration Register (ADCCLP4L)

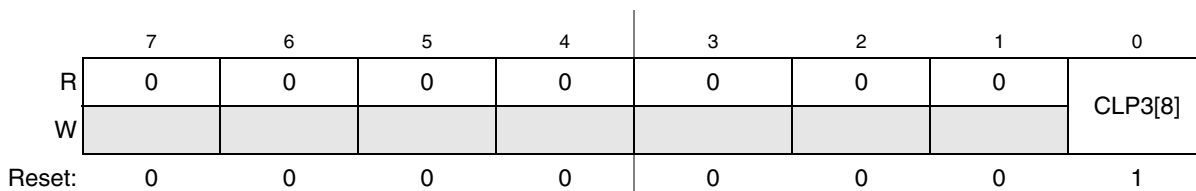


Figure 17-24. Plus-Side General Calibration Register (ADCCLP3H)

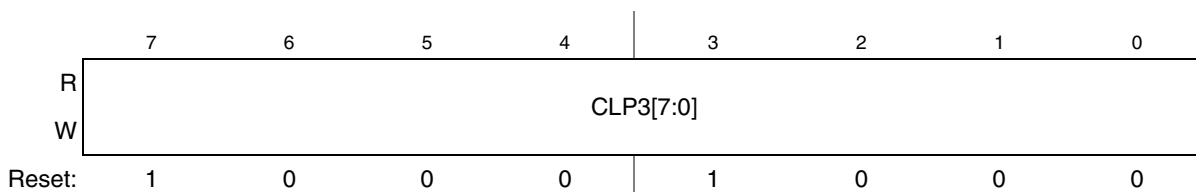


Figure 17-25. Plus-Side General Calibration Register (ADCCLP3L)

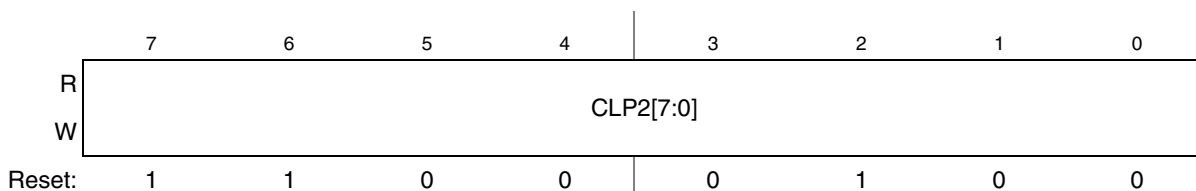


Figure 17-26. Plus-Side General Calibration Register (ADCCLP2)

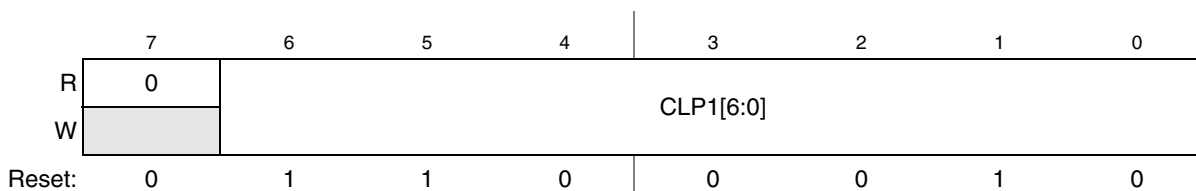


Figure 17-27. Plus-Side General Calibration Register (ADCCLP1)



Figure 17-28. Plus-Side General Calibration Register (ADCCLP0)

## 17.4.12 ADC Minus-Side General Calibration Value Registers (ADCCLMx)

ADCCLMx contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLM0[5:0], CLM1[6:0], CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. ADCCLMx are automatically set once the self calibration sequence is done (CAL is cleared). If these registers are written by the user after calibration, the linearity error specifications may not be met.



Figure 17-29. Minus-Side General Calibration Register (ADCCLMD)



Figure 17-30. Minus-Side General Calibration Register (ADCCLMS)

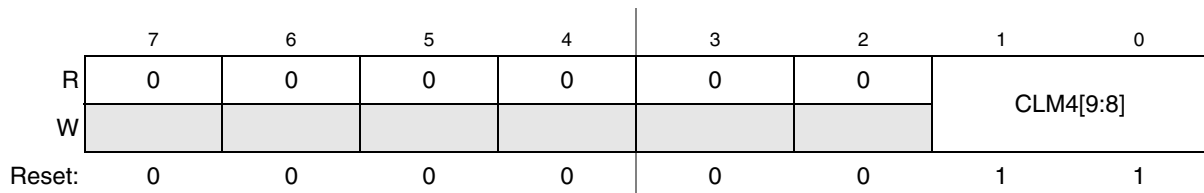


Figure 17-31. Minus-Side General Calibration Register (ADCCLM4H)

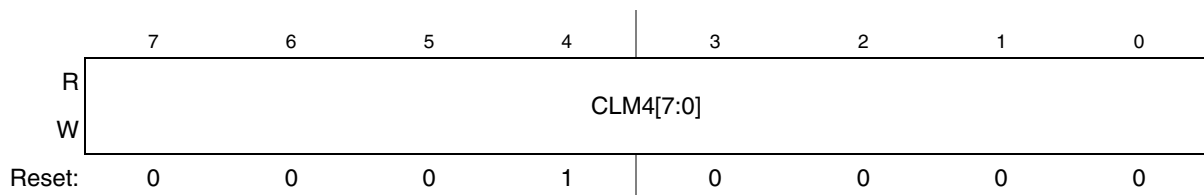


Figure 17-32. Minus-Side General Calibration Register (ADCCLM4L)

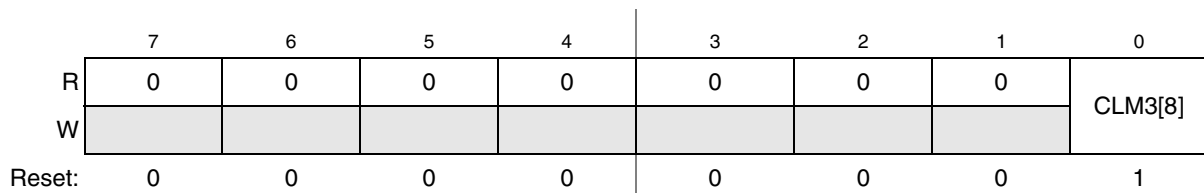


Figure 17-33. Minus-Side General Calibration Register (ADCCLM3H)

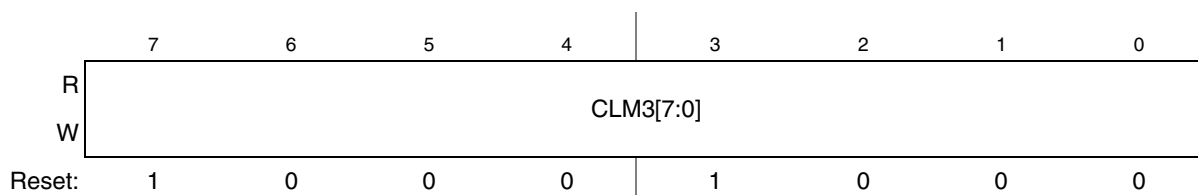


Figure 17-34. Minus-Side General Calibration Register (ADCCLM3L)

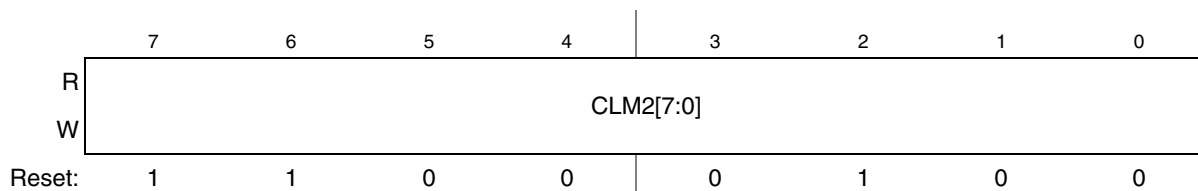


Figure 17-35. Minus-Side General Calibration Register (ADCCLM2)

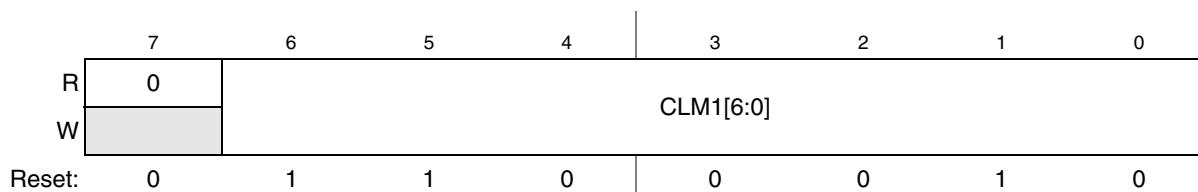


Figure 17-36. Minus-Side General Calibration Register (ADCCLM1)



Figure 17-37. Minus-Side General Calibration Register (ADCCLM0)

### 17.4.13 Pin Control 1 Register (APCTL1)

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

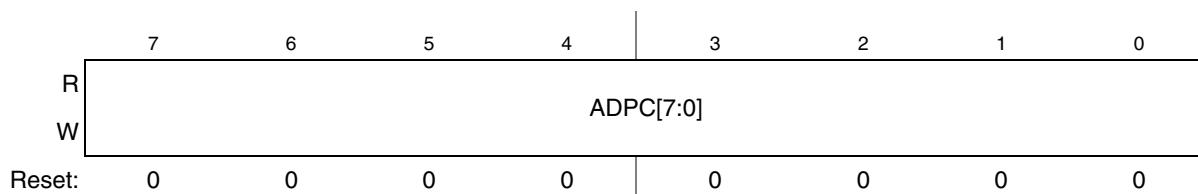


Figure 17-38. Pin Control 1 Register (APCTL1)

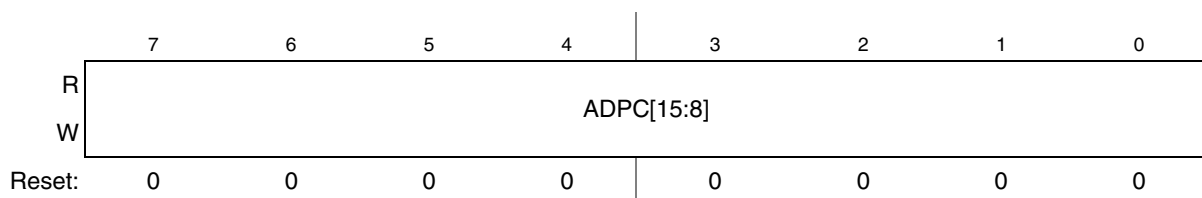


**Table 17-13. APCTL1 Register Field Descriptions**

Field	Description
7 ADPC7	<b>ADC Pin Control 7</b> - ADPC7 controls the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	<b>ADC Pin Control 6</b> - ADPC6 controls the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	<b>ADC Pin Control 5</b> - ADPC5 controls the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	<b>ADC Pin Control 4</b> - ADPC8 controls the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	<b>ADC Pin Control 3</b> - ADPC3 controls the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	<b>ADC Pin Control 2</b> - ADPC2 controls the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled
1 ADPC1	<b>ADC Pin Control 1</b> - ADPC1 controls the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	<b>ADC Pin Control 0</b> - ADPC0 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

#### 17.4.14 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.

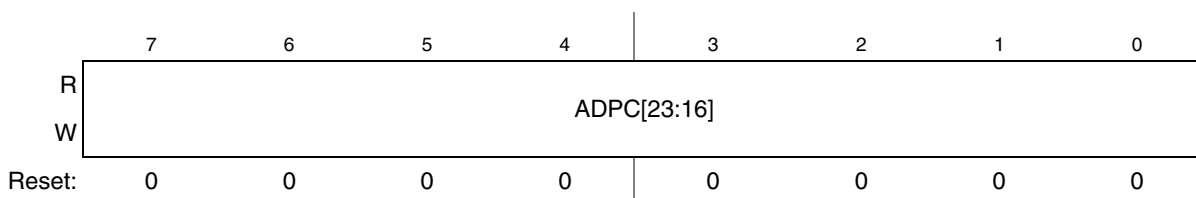
**Figure 17-39. Pin Control 2 Register (APCTL2)**

**Table 17-14. APCTL2 Register Field Descriptions**

Field	Description
7 ADPC15	<b>ADC Pin Control 15</b> - ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	<b>ADC Pin Control 14</b> - ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	<b>ADC Pin Control 13</b> - ADPC13 controls the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	<b>ADC Pin Control 12</b> - ADPC12 controls the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	<b>ADC Pin Control 11</b> - ADPC11 controls the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	<b>ADC Pin Control 10</b> - ADPC10 controls the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled
1 ADPC9	<b>ADC Pin Control 9</b> - ADPC9 controls the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	<b>ADC Pin Control 8</b> - ADPC8 controls the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

### 17.4.15 Pin Control 3 Register (APCTL3)

APCTL3 controls channels 23–16 of the ADC module.

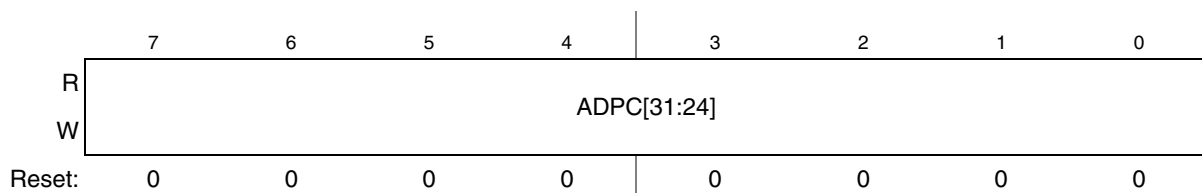
**Figure 17-40. Pin Control 3 Register (APCTL3)**

**Table 17-15. APCTL3 Register Field Descriptions**

Field	Description
7 ADPC23	<b>ADC Pin Control 23</b> - ADPC23 controls the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	<b>ADC Pin Control 22</b> - ADPC22 controls the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	<b>ADC Pin Control 21</b> - ADPC21 controls the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	<b>ADC Pin Control 20</b> - ADPC20 controls the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	<b>ADC Pin Control 19</b> - ADPC19 controls the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	<b>ADC Pin Control 18</b> - ADPC18 controls the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled
1 ADPC17	<b>ADC Pin Control 17</b> - ADPC17 controls the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	<b>ADC Pin Control 16</b> - ADPC16 controls the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

### 17.4.16 Pin Control 4 Register (APCTL4)

APCTL4 controls channels DAD0-DAD3 of the ADC module. When DIFFn=1, channels DAD0-DAD3 use the input pin pairs DADPx and DADMx to form a differential conversion. When DIFFn=0, channels DAD0-DAD3 use only the input pins DADPx and the DADMx input pins are ingored.

**Figure 17-41. Pin Control 4 Register (APCTL4)**

**Table 17-16. APCTL4 Register Field Descriptions**

Field	Description
7 ADPC31	<b>ADC Pin Control 31</b> - ADPC31 controls the pin associated with channel AD31. 0 AD31 pin I/O control enabled 1 AD31 pin I/O control disabled
6 ADPC30	<b>ADC Pin Control 30</b> - ADPC30 controls the pin associated with channel AD30. 0 AD30 pin I/O control enabled 1 AD30 pin I/O control disabled
5 ADPC29	<b>ADC Pin Control 29</b> - ADPC29 controls the pin associated with channel AD29. 0 AD29 pin I/O control enabled 1 AD29 pin I/O control disabled
4 ADPC28	<b>ADC Pin Control 28</b> - ADPC28 controls the pin associated with channel AD28. 0 AD28 pin I/O control enabled 1 AD28 pin I/O control disabled
3 ADPC27	<b>ADC Pin Control 27</b> - ADPC27 controls the pin associated with channel AD27. 0 AD27 pin I/O control enabled 1 AD27 pin I/O control disabled
2 ADPC26	<b>ADC Pin Control 26</b> - ADPC26 controls the pin associated with channel AD26. 0 AD26 pin I/O control enabled 1 AD26 pin I/O control disabled
1 ADPC25	<b>ADC Pin Control 25</b> - ADPC25 controls the pin associated with channel AD25. 0 AD25 pin I/O control enabled 1 AD25 pin I/O control disabled
0 ADPC24	<b>ADC Pin Control 24</b> - ADPC24 controls the pin associated with channel AD24. 0 AD24 pin I/O control enabled 1 AD24 pin I/O control disabled

## 17.5 Functional Description

The ADC module is disabled during reset, stop2 or when the ADCHn bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle and the asynchronous clock output enable is disabled (ADACKEN=0), the module is in its lowest power state. The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications the ADC module must be calibrated using the on chip calibration function. Calibration is recommended to be done after any reset. See [Section 17.5.7](#) for details on how to perform calibration.

When the conversion is completed, the result is placed in the data registers (ADCRHn and ADCRLn). The conversion complete flag (COCON) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (AIENn=1).

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

The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average function is enabled by setting the AVGE bit and operates with any of the conversion modes and configurations.

### 17.5.1 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. 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 with using the ADIV bits.
- 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. Conversions are possible using ADACK as the input clock source while the MCU is in stop3 mode. Refer to [Section 17.5.5.4](#) for more information.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may 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.

### 17.5.2 Input Select and Pin Control

The pin control registers (APCTL1, APCTL2, APCTL3, and APCTL4) disable the I/O port control of the pins used as analog inputs. When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

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

### 17.5.3 Voltage Reference Selection

The ADC can be configured to accept one of three voltage reference pairs as the reference voltage ( $V_{REFSH}$  and  $V_{REFSL}$ ) used for conversions. Each pair contains a positive reference which must be between the minimum Ref Voltage High (defined in Appendix A) and  $V_{DDA}$ , and a ground reference which must be at the same potential as  $V_{SSA}$ . The three pairs are external ( $V_{REFH}$  and  $V_{REFL}$ ), alternate ( $V_{ALTH}$  and  $V_{ALTL}$ ) and the internal bandgap ( $V_{BGH}$  and  $V_{BGL}$ ). These voltage references are selected using the REFSEL bits in the ADCSC2 register. The alternate ( $V_{ALTH}$  and  $V_{ALTL}$ ) voltage reference pair may select additional external pins or internal sources depending on MCU configuration. Consult the module introduction for information on the Voltage References specific to this MCU.

## 17.5.4 Hardware Trigger and Channel Selects

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

When the ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of the ADHWT after a hardware trigger select event (ADHWTSn) has occurred. If a conversion is in progress when a rising edge of a trigger occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed and until conversion gets aborted the ADC will continue to do conversions on the same ADC Status and Control register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The hardware trigger select event (ADHWTSn) must be set prior to and during the receipt of the ADHWT signal. If these conditions are not met the converter may ignore the trigger or use the incorrect configuration. If a hardware trigger select event gets asserted during a conversion it must stay asserted until end of current conversion and remain set until the receipt of the ADHWT signal to trigger a new conversion. The channel and status fields selected for the conversion will depend on the active trigger select signal (ADHWTSn active selects ADCSC1A; ADHWTSn active selects ADCSC1n).

### NOTE:

Asserting more than one hardware trigger select signal (ADHWTSn) at the same time will result in unknown results. To avoid this, only select one hardware trigger select signal (ADHWTSn) prior to the next intended conversion.

When the conversion is completed, the result is placed in the data registers associated with the ADHWTSn received (ADHWTSn active selects ADCRHA:ADCRLA; ADHWTSn active selects ADCRHn:ADCRLn). The conversion complete flag associated with the ADHWTSn received (COCON) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (AIENn=1).

## 17.5.5 Conversion Control

Conversions can be performed as determined by the MODE bits and the DIFFn bit as shown in [Table 17-7](#).

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, hardware average and automatic compare of the conversion result to a software determined compare value.

### 17.5.5.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1A (with ADCHA bits not all 1's) if software triggered operation is selected (ADTRG=0).

- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected (ADTRG=1) and a hardware trigger select event (ADHWTSn) has occurred. The channel and status fields selected will depend on the active trigger select signal (ADHWTSn active selects ADCSC1A; ADHWTSn active selects ADCSC1n; if neither is active the off condition is selected).

**NOTE:**

Selecting more than one hardware trigger select signal (ADHWTSn) prior to a conversion completion will result in unknown results. To avoid this, only select one hardware trigger select signal (ADHWTSn) prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled (ADCO=1).

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation (ADTRG=0), continuous conversions begin after ADCSC1A is written and continue until aborted. In hardware triggered operation (ADTRG=1 and one ADHWTSn event has occurred), continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions is completed. In software triggered operation, conversions begin after ADCSC1A is written. In hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions are also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

### 17.5.5.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRHn and ADCRLn. If the compare functions are disabled, this is indicated by the setting of COCON. If hardware averaging is enabled, COCON sets only if the last of the selected number of conversions is complete. If the compare function is enabled, COCON sets and conversion result data is transferred only if the compare condition is true. If both hardware averaging and compare functions are enabled then COCON sets only if the last of the selected number of conversions is complete and the compare condition is true. An interrupt is generated if AIENn is high at the time that COCON is set. In all modes except 8-bit single-ended conversions, a blocking mechanism prevents a new result from overwriting previous data in ADCRHn and ADCRLn if the previous data is in the process of being read (the ADCRHn register has been read but the ADCRLn register has not). When blocking is active, the conversion result data transfer is blocked, COCON 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 conversion result data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

**NOTE:**

If continuous conversions are enabled, the blocking mechanism could result in the loss of data occurring at specific timepoints. To avoid this issue, the data must be read in fewer cycles than an ADC conversion time, accounting for interrupt or software polling loop latency.

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.

**17.5.5.3 Aborting Conversions**

Any conversion in progress is aborted when:

- Writing ADCSC1A while ADCSC1A is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), a write to ADCSC1A initiates a new conversion (if the ADCHA bits are equal to a value other than all 1s). Writing any of the ADCSC1(B-n) registers while that specific ADCSC1(B-n) register is actively controlling a conversion aborts the current conversion. The ADCSC1(B-n) registers are not used for software trigger operation and therefore writes to the ADCSC1(B-n) registers do not initiate a new conversion.
- A write to any ADC register besides the ADCSC1A:ADCSC1n registers occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset or enters stop2 mode.
- The MCU enters stop3 mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRHn and ADCRLn, are not altered. The data registers continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset or stop2, ADCRHA:ADCRLA and ADCRHn:ADCRLn return to their reset states.

**17.5.5.4 Power Control**

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source but the asynchronous clock output is disabled (ADACKEN=0), the ADACK clock generator will also remain in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled (ADACKEN=1), it will remain active regardless of the state of the ADC or the MCU power mode.

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



### 17.5.5.5 Sample Time and Total Conversion Time

The total conversion time depends upon: the sample time (as determined by ADLSMP and ADLSTS bits), the MCU bus frequency, the conversion mode (as determined by MODE and DIFFn bits), the high speed configuration (ADHSC bit), and the frequency of the conversion clock ( $f_{ADCK}$ ).

The ADHSC bit is used to configure a higher clock input frequency. This will allow faster overall conversion times. In order to meet internal A/D converter timing requirements the ADHSC bit adds additional ADCK cycles. Conversions with ADHSC = 1 take four more ADCK cycles. ADHSC should be used when the ADCLK exceeds the limit for ADHSC = 0.

After the module becomes active, sampling of the input begins. ADLSMP and ADLSTS select between sample times based on the conversion mode that is selected. 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 ADCRHn and ADCRLn 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). 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. The maximum total conversion time for all configurations is summarized in the equation below. Refere to tables 1-20 through 1-24 for the variables referenced in the equation.

#### Conversion Time Equation

*Eqn. 17-1*

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

**Table 17-17. Single or First Continuous Time Adder (SFCAdder)**

ADLSMP	ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
1	x	0x, 10	3 ADCK cycles + 5 bus clock cycles
1	1	11	3 ADCK cycles + 5 bus clock cycles <sup>1</sup>
1	0	11	5 $\mu$ s + 3 ADCK cycles + 5 bus clock cycles
0	x	0x, 10	5 ADCK cycles + 5 bus clock cycles
0	1	11	5 ADCK cycles + 5 bus clock cycles <sup>1</sup>
0	0	11	5 $\mu$ s + 5 ADCK cycles + 5 bus clock cycles

<sup>1</sup> ADACKEN must be 1 for at least 5us prior to the conversion is initiated to achieve this time

**Table 17-18. Average Number Factor (AverageNum)**

AVGE	AVGS[1:0]	Average Number Factor (AverageNum)
0	xx	1
1	00	4
1	01	8
1	10	16
1	11	32

**Table 17-19. Base Conversion Time (BCT)**

Mode	Base Conversion Time (BCT)
8b se	17 ADCK cycles
9b diff	27 ADCK cycles
10b s.e.	20 ADCK cycles
11b diff	30 ADCK cycles
12b s.e.	20 ADCK cycles
13b diff	30 ADCK cycles
16b s.e.	25 ADCK cycles
16b diff	34 ADCK cycles

**Table 17-20. Long Sample Time Adder (LSTAdder)**

ADLSMP	ADLSTS	Long Sample Time Adder (LSTAdder)
0	xx	0 ADCK cycles
1	00	20 ADCK cycles
1	01	12 ADCK cycles
1	10	6 ADCK cycles
1	11	2 ADCK cycles

**Table 17-21. High Speed Conversion time Adder (HSCAdder)**

ADHSC	High Speed Conversion Time Adder (HSCAdder)
0	0 ADCK cycles
1	4 ADCK cycles

**NOTE:**

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

### 17.5.5.6 Conversion Time Examples

The following examples use equation 1-3 and the information provided in tables 1-20 through 1-24.

#### 17.5.5.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is: 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, long sample time disabled and high speed conversion disabled. The conversion time for a single conversion is calculated by using equation 1-3 and the information provided in tables 1-20 through 1-24. The table below list the variables of equation 1-3.

**Table 17-22. Typical Conversion Time**

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
AverageNum	1
BCT	20 ADCK cycles
LSTAdder	0
HSCAdder	0

The resulting conversion time is generated using the parameters listed in table1-25. So for Bus clock equal to 8Mhz and ADCK equal to 8Mhz the resulting conversion time is 3.75  $\mu$ s.

#### 17.5.5.6.2 Long conversion time configuration

A configuration for long ADC conversion is: 16-bit differential mode, with the bus clock selected as the input clock source, the input clock divide-by-8 ratio selected, and a bus frequency of 8 MHz, long sample time enabled and configured for longest adder and high speed conversion disabled. Average enabled for 32 conversions. The conversion time for this conversion is calculated by using equation 1-3 and the information provided in tables 1-20 through 1-24. The table below list the variables of equation 1-3.

**Table 17-23. Typical Conversion Time**

Variable	Time
SFCAdder	3 ADCK cycles + 5 bus clock cycles
AverageNum	32
BCT	34 ADCK cycles
LSTAdder	20 ADCK cycles
HSCAdder	0

The resulting conversion time is generated using the parameters listed in table1-25. So for Bus clock equal to 8Mhz and ADCK equal to 1Mhz the resulting conversion time is 57.625us(AverageNum). This results in a total conversion time of 1.844ms.

### 17.5.5.6.3 Short conversion time configuration

A configuration for short ADC conversion is: 8-bit single ended mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 20 MHz, long sample time disabled and high speed conversion enabled. The conversion time for this conversion is calculated by using equation 1-3 and the information provided in tables 1-20 through 1-24. The table below list the variables of equation 1-3.

**Table 17-24. Typical Conversion Time**

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
AverageNum	1
BCT	17 ADCK cycles
LSTAdder	0 ADCK cycles
HSCAdder	4

The resulting conversion time is generated using the parameters listed in table 1-25. So for Bus clock equal to 20Mhz and ADCK equal to 20Mhz the resulting conversion time is 1.625  $\mu$ s.

### 17.5.5.7 Hardware Average Function

The hardware average function can be enabled (AVGE=1) to perform a hardware average of multiple conversions. The number of conversions is determined by the AVGS[1:0] bits, which select 4, 8, 16 or 32 conversions to be averaged. While the hardware average function is in progress the ADACT bit will be set.

After the selected input is sampled and converted, the result is placed in an accumulator from which an average is calculated once the selected number of conversions has been completed. When hardware averaging is selected the completion of a single conversion will not set the COCON bit.

If the compare function is either disabled or evaluates true, after the selected number of conversions are completed, the average conversion result is transferred into the data result registers, ADCRHn and ADCRLn, and the COCON bit is set. An ADC interrupt is generated upon the setting of COCON if the respective ADC interrupt is enabled (AIENn=1).

#### NOTE:

The hardware average function can perform conversions on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the hardware average is complete if AIENn was set.

## 17.5.6 Automatic Compare Function

The compare function can be configured to check if the result is less than or greater-than-or-equal-to a single compare value, or if the result falls within or outside a range determined by two compare values. The compare mode is determined by ACFG, ACREN and the values in the compare value registers (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L). After the input is sampled and converted,

the compare values (ADCCV1H:ADCCV1L and ADCCV2H:ADCCV2L) are used as described in Table 17-25. There are six compare modes as shown in Table 17-25.

**Table 17-25. Compare Modes**

ACFGT	ACREN	ADCCV1 relative to ADCCV2	Function	Compare Mode Description
0	0	-	Less than threshold	Compare true if the result is less than the ADCCV1 registers.
1	0	-	Greater than or equal to threshold	Compare true if the result is greater than or equal to ADCCV1 registers.
0	1	Less than or equal	Outside range, not inclusive	Compare true if the result is less than ADCCV1 <b>Or</b> the result is Greater than ADCCV2
0	1	Greater than	Inside range, not inclusive	Compare true if the result is less than ADCCV1 <b>And</b> the result is greater than ADCCV2
1	1	Less Than or equal	Inside range, inclusive	Compare true if the result is greater than or equal to ADCCV1 <b>And</b> the result is less than or equal to ADCCV2
1	1	Greater than	Outside range, inclusive	Compare true if the result is greater than or equal to ADCCV1 <b>Or</b> the result is less than or equal to ADCCV2

With the ADC range enable bit set, ADCREN =1, if compare value register 1 (ADCCV1 value) is less than or equal to the compare value register 2 (ADCCV2 value), setting ACFGT will select a trigger-if-inside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-outside-compare-range, not-inclusive-of-endpoints function.

If ADCCV1 is greater than the ADCCV2, setting ACFGT will select a trigger-if-outside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-inside-compare-range, not-inclusive-of-endpoints function.

If the condition selected evaluates true, COCOn is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCOn is not set and the conversion result data will not be transferred to the result register. If the hardware averaging function is enabled, the compare function compares the averaged result to the compare values. The same compare function definitions apply. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (AIENn=1).

**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.

## 17.5.7 Calibration Function

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration must be run or valid calibration values written after any reset and before a conversion is initiated. The calibration function sets the offset calibration value and the plus-side and minus-side calibration values.

The offset calibration value is automatically stored in the ADC Offset Correction Registers (ADCOFSH and ADCOFSL) and the plus-side and minus-side calibration values are automatically stored in the ADC Plus-Side and Minus-Side Calibration registers (CLPD, CLPS, CLP4, CLP3, CLP2, CLP1, CLP0 and CLMD, CLMS, CLM4, CLM3, CLM2, CLM1, CLM0). The user must configure the ADC correctly prior to calibration, and must generate the plus-side and minus-side gain calibration results and store them in the ADC GAIN registers (ADCPGH and ADCPGL) after the calibration function completes.

Prior to calibration, the user must configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time and the high speed configuration according to the application's clock source availability and needs. If the application uses the ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected, or multiple calibrations can be done for the different configurations. The input channel, conversion mode continuous function, compare function, hardware average function, resolution mode, and differential/single-ended mode are all ignored during the calibration function.

To initiate calibration, the user sets the CAL bit and the calibration will automatically begin if the ADTRG bit = 0. If ADTRG = 1, the CAL bit will not get set and the calibration fail flag (CALF) will be set. While calibration is active, no ADC register can be written and no stop mode may be entered or the calibration routine will be aborted causing the CAL bit to clear and the CALF bit to set. At the end of a calibration sequence the COCO bit of the ADSC1A register will be set. The AIEN1 bit can be used to allow an interrupt to occur at the end of a calibration sequence. If at the end of calibration routine the CALF bit is not set, the automatic calibration routine completed successfully.

To complete calibration, the user must generate the gain calibration values using the following procedure:

- Initialize (clear) a 16b variable in RAM.
- Add the following plus-side calibration results CLP0, CLP1, CLP2, CLP3, CLP4, and CLPS to the variable.
- Divide the variable by two.
- Set the MSB of the variable.
- The previous two steps can be achieved by setting the carry bit, rotating-right through the carry bit on the high byte and again on the low byte.
- Store the value in the plus-side gain calibration registers ADCPGH and ADCPGL.
- Repeat procedure for the minus-side gain calibration value.

When complete the user may reconfigure and use the ADC as desired. A second calibration may also be performed if desired by clearing and again setting the CAL bit.

Overall the calibration routine may take as many as 14000 ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen. For an 8MHz clock source this is about 1.7msec. To reduce this latency, the calibration values (offset, plus- and minus-side gain, and plus- and minus-side calibration values) may be stored in flash after an initial calibration and recovered prior to the first ADC conversion. This should reduce the calibration latency to 20 register store operations on all subsequent power, reset, or stop2 mode recoveries.

## 17.5.8 User Defined Offset Function

The ADC Offset Correction Register (ADCOFSH:ADCOFSL) contains the user selected or calibration generated offset error correction value. This register is a 2's complement, left justified, 16b value formed by the concatenation of ADCOFSH and ADCOFSL. The value in the offset correction registers (ADCOFSH:ADCOFSL) is subtracted from the conversion and the result is transferred into the result registers (ADCRHn:ADCRLn). If the result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation. For additional information please see [Section 17.5.8](#)

- The formatting of the ADC Offset Correction Register is different from the Data Result Registers (ADCRHn:ADCRLn) to preserve the resolution of the calibration value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8b single-ended mode, the bits OFS[14:7] are subtracted from D[7:0]; bit OFS[15] indicates the sign (negative numbers are effectively added to the result) and bits OFS[6:0] are ignored. The same bits are used in 9b differential mode since bit OFS[15] indicates the sign bit, which maps to bit D[8]. For 16b differential mode, all bits OFS[15:0] are directly subtracted from the conversion result data D[15:0]. Finally, in 16b single-ended mode, there is no bit in the Offset Correction Register corresponding to the least significant result bit D[0], so odd values (-1 or +1, etc.) cannot be subtracted from the result. ADCOFSH is automatically set according to calibration requirements once the self calibration sequence is done (CAL is cleared). The user may write ADCOFSH:ADCOFSL to override the calibration result if desired. If the Offset Correction Register is written by the user to a value that is different from the calibration value, the ADC error specifications may not be met. It is recommended that the value generated by the calibration function be stored in memory before overwriting with a user specified value. NOTE: There is an effective limit to the values of Offset that can be set by the user. If the magnitude of the offset is too great the results of the conversions will cap off at the limits.

The offset calibration function may be employed by the user to remove application offsets or DC bias values. The Offset Correction Registers ADCOFSH and ADCOFSL may be written with a number in 2's complement format and this offset will be subtracted from the result (or hardware averaged value). To add an offset, store the negative offset in 2's complement format and the effect will be an addition. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value (the minimum value for single-ended conversions is 0x0000; for a differential conversion 0x8000).

To preserve accuracy, the calibrated offset value initially stored in the ADCOFS registers must be added to the user defined offset. For applications which may change the offset repeatedly during operation, it is recommended to store the initial offset calibration value in flash so that it can be recovered and added to any user offset adjustment value and the sum stored in the ADCOFS registers.

## 17.5.9 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 17-2 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{\text{TEMP}} - V_{\text{TEMP25}}) \div m) \quad \text{Eqn. 17-2}$$

where:

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

For temperature calculations, use the  $V_{\text{TEMP25}}$  and  $m$  values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates  $V_{\text{TEMP}}$  and compares to  $V_{\text{TEMP25}}$ . If  $V_{\text{TEMP}}$  is greater than  $V_{\text{TEMP25}}$  the cold slope value is applied in Equation 17-2. If  $V_{\text{TEMP}}$  is less than  $V_{\text{TEMP25}}$  the hot slope value is applied in Equation 17-2.

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

### 17.5.10 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.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCON and generates an ADC interrupt to wake the MCU from wait mode if the respective ADC interrupt is enabled ( $\text{AIENn}=1$ ). If the hardware averaging function is enabled the COCON will set (and generate an interrupt if enabled) when the selected number of conversions are complete. If the compare function is enabled the COCON will set (and generate an interrupt if enabled) only if the compare conditions are met. If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from wait mode unless a new conversion is initiated by the hardware trigger.

### 17.5.11 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.



### 17.5.11.1 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 the ADC registers, including ADCRHn and ADCRLn, are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

### 17.5.11.2 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.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCON and generates an ADC interrupt to wake the MCU from stop3 mode if the respective ADC interrupt is enabled ( $AIENn = 1$ ). The result register will contain the data from the first completed conversion that occurred during stop3 mode. If the hardware averaging function is enabled the COCON will set (and generate an interrupt if enabled) when the selected number of conversions are complete. If the compare function is enabled the COCON will set (and generate an interrupt if enabled) only if the compare conditions are met. If a single conversion is selected and the compare is not true, the ADC will return to its idle state and cannot wake the MCU from stop3 mode unless a new conversion is initiated by another hardware trigger.

#### 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 conversion result data transfer blocking mechanism (discussed in [Section 17.5.5.2, "Completing Conversions"](#)) is cleared when entering stop3 and continuing ADC conversions.

## 17.5.12 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.

## 17.6 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-, 12-, or 16-bit single-ended resolution or 9-, 11-, 13-, or 16-bit differential resolution, single or continuous conversion, and a polled or interrupt approach,

among many other options. Refer to [Table 17-6](#), [Table 17-7](#), and [Table 17-8](#) 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.

## 17.6.1 ADC Module Initialization Example

### 17.6.1.1 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. Calibrate the ADC by following the calibration instructions in [Section 17.5.7](#).
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 3 (ADCSC3) to select whether conversions will be continuous or completed only once (ADCO) and to select whether to perform hardware averaging.
4. Update status and control register (ADCSC1:ADCSC1n) to select whether conversions will be single-ended or differential and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

### 17.6.1.2 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 ÷ 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.

**ADCSC1A = 0x41 (%01000001)**

Bit 7	COCOA	0	Read-only flag which is set when a conversion completes.
-------	-------	---	--

Bit 6	AIENA	1	Conversion complete interrupt enabled.
Bit 5	ADCOA	0	One conversion only (continuous conversions disabled).
Bit 4:0	ADCHA	00001	Input channel 1 selected as ADC input channel.

**ADCRHA/LA = 0xxx**

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

**ADCCVH/L = 0xxx**

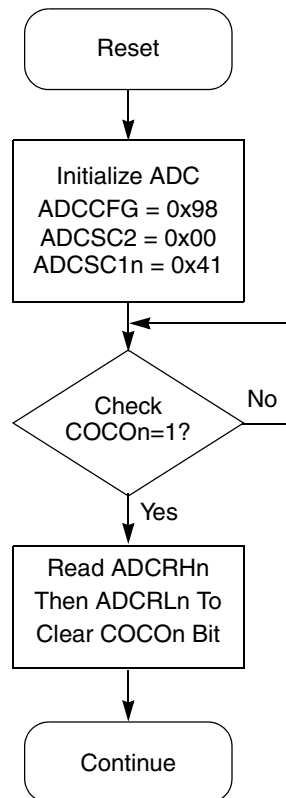
Holds compare value when compare function enabled.

**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 17-42. Initialization Flowchart for Example**

## 17.7 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.

## 17.7.1 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.

### 17.7.1.1 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.

### 17.7.1.2 Analog Voltage Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs used by the converter,  $V_{REFSH}$  and  $V_{REFSL}$ .  $V_{REFSH}$  is the high reference voltage for the converter.  $V_{REFSL}$  is the low reference voltage for the converter.

The ADC can be configured to accept one of three voltage reference pairs for  $V_{REFSH}$  and  $V_{REFSL}$ . Each pair contains a positive reference and a ground reference. The three pairs are external ( $V_{REFH}$  and  $V_{REFL}$ ), alternate ( $V_{ALTH}$  and  $V_{ALTL}$ ) and the internal bandgap ( $V_{BGH}$  and  $V_{BGL}$ ). These voltage references are selected using the REFSEL bits in the ADCSC2 register. The alternate ( $V_{ALTH}$  and  $V_{ALTL}$ ) voltage reference pair may select additional external pins or internal sources depending on MCU configuration. Consult the module introduction for information on the Voltage References specific to this MCU.

In some packages, the external or alternate pairs are connected in the package to  $V_{DDA}$  and  $V_{SSA}$ , respectively. One of these positive references may be shared on the same pin as  $V_{DDA}$  on some devices. One of these ground references may be shared on the same pin as  $V_{SSA}$  on some devices.

If externally available, the positive reference may be connected to the same potential as  $V_{DDA}$  or may be driven by an external source to a level between the minimum Ref Voltage High (defined in Appendix A) and the  $V_{DDA}$  potential (the positive reference must never exceed  $V_{DDA}$ ). If externally available, the ground reference must be connected to the same voltage potential as  $V_{SSA}$ . The voltage reference pairs 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).

### 17.7.1.3 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  $\mu\text{F}$  capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to  $V_{SSA}$ .

For proper conversion, the input voltage must fall between  $V_{REFH}$  and  $V_{REFL}$ . If the input is equal to or exceeds  $V_{REFH}$ , the converter circuit converts the signal to 0xFFF (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.

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

## 17.7.2 Sources of Error

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

### 17.7.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7k $\Omega$  and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 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 and changing the ADLSTS bits (to increase the sample window) or decreasing ADCK frequency to increase sample time.

### 17.7.2.2 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  $V_{DDA} / (2^N \cdot I_{LEAK})$  for less than 1/4LSB leakage error ( $N = 8$  in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

### 17.7.2.3 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  $V_{\text{REFH}}$  to  $V_{\text{REFL}}$ .
- There is a 0.1  $\mu\text{F}$  low-ESR capacitor from  $V_{\text{DDA}}$  to  $V_{\text{SSA}}$ .
- If inductive isolation is used from the primary supply, an additional 1  $\mu\text{F}$  capacitor is placed from  $V_{\text{DDA}}$  to  $V_{\text{SSA}}$ .
- $V_{\text{SSA}}$  (and  $V_{\text{REFL}}$ , if connected) is connected to  $V_{\text{SS}}$  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 stop instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces  $V_{\text{DD}}$  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  $V_{\text{DD}}$  noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01  $\mu\text{F}$  capacitor ( $C_{\text{AS}}$ ) on the selected input channel to  $V_{\text{REFL}}$  or  $V_{\text{SSA}}$  (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.

### 17.7.2.4 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:

$$1 \text{ lsb} = (V_{\text{REFH}} - V_{\text{REFL}}) / 2^N \quad \text{Eqn. 17-3}$$

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.

### 17.7.2.5 Linearity Errors

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

- Zero-scale error ( $E_{ZS}$ ) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width ( $1/2$  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 ( $E_{FS}$ ) — 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  $1$ LSB in 12-bit mode). If the last conversion is  $0x3FE$ , the difference between the actual  $0x3FE$  code width and its ideal ( $1$ LSB) 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.

### 17.7.2.6 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 17.7.2.3](#) 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 18

# FlexTimer Module (S08FTMV3)

### 18.1 Introduction

MC9S08GW64 series contain one two-channel FTM module. It supports traditional input capture, output compare, or edge-aligned PWM on each channel. A control bit allows the FTM to be configured such that all channels may be used for center-aligned PWM functions. Timing functions are based on a 16-bit counter with prescaler and modulo features to control frequency and range of control applications, and the center-aligned PWM capability extends the field of application to motor control applications.

#### 18.1.1 FTM External Clock Source

The FTM clock source is selectable as bus clock, prescaled bus clock, fixed system clock or an external clock source which is FTMCLK pin, MTIMCLK or MTIM1 overflow.

#### 18.1.2 FTM Valid Function

MC9S08GW64 series support FTM module's 0~1 channels. The channel 2 to channel 7 are not available. Please ignore the registers for FTM channel 2 to channel 7.

#### NOTE

For FlexTimer module, MC9S08GW64 series only support TPM function which is the basic function of FlexTimer. The other function of FlexTimer is not available.

#### 18.1.3 FTM Clock Gating

The bus clock to the FTM module can be gated on and off using the FTM bit in SCGC4. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the FTM bit can be cleared to disable the clock to this module. See [Section 5.7, "Peripheral Clock Gating,"](#) for details.

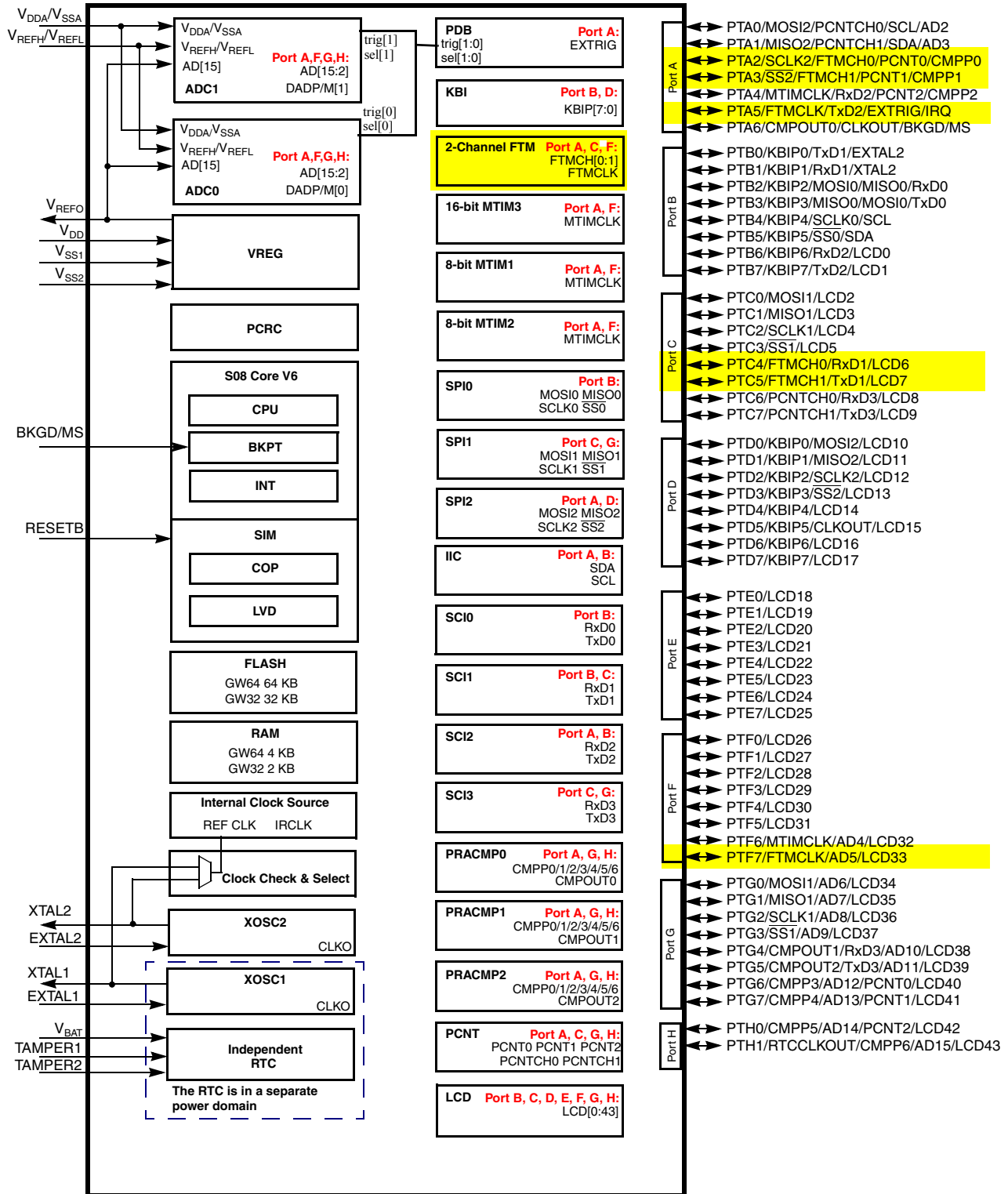


Figure 18-1. MC9S08GW64 Series Block Diagram Highlighting FTM Module and Pins

### 18.1.4 Features

The FTM features include:

- FTM source clock is selectable
  - Source clock can be the system clock, the fixed frequency clock, or an external clock
  - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than the system clock
  - Selecting external clock connects FTM clock to a chip level input pin therefore allowing to synchronize the FTM counter with an off chip clock source
- 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 selectable 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
- 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
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- Backwards compatible with TPM

### 18.1.5 Modes of Operation

When the MCU is in active BDM background or BDM foreground mode, the FTM temporarily suspends all counting until the MCU returns to normal user operating mode. 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. If the FTM does not need to produce a real time reference or provide the interrupt sources needed to wake the MCU from wait mode, the power can then be saved by disabling FTM functions before entering wait mode.

### 18.1.6 Block Diagram

The FTM uses one input/output (I/O) pin per channel, FTMCH<sub>n</sub> (FTM channel (n)) where n is the channel number (0–7).

Figure 18-2 shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable final value and its counting can be up or up-down.

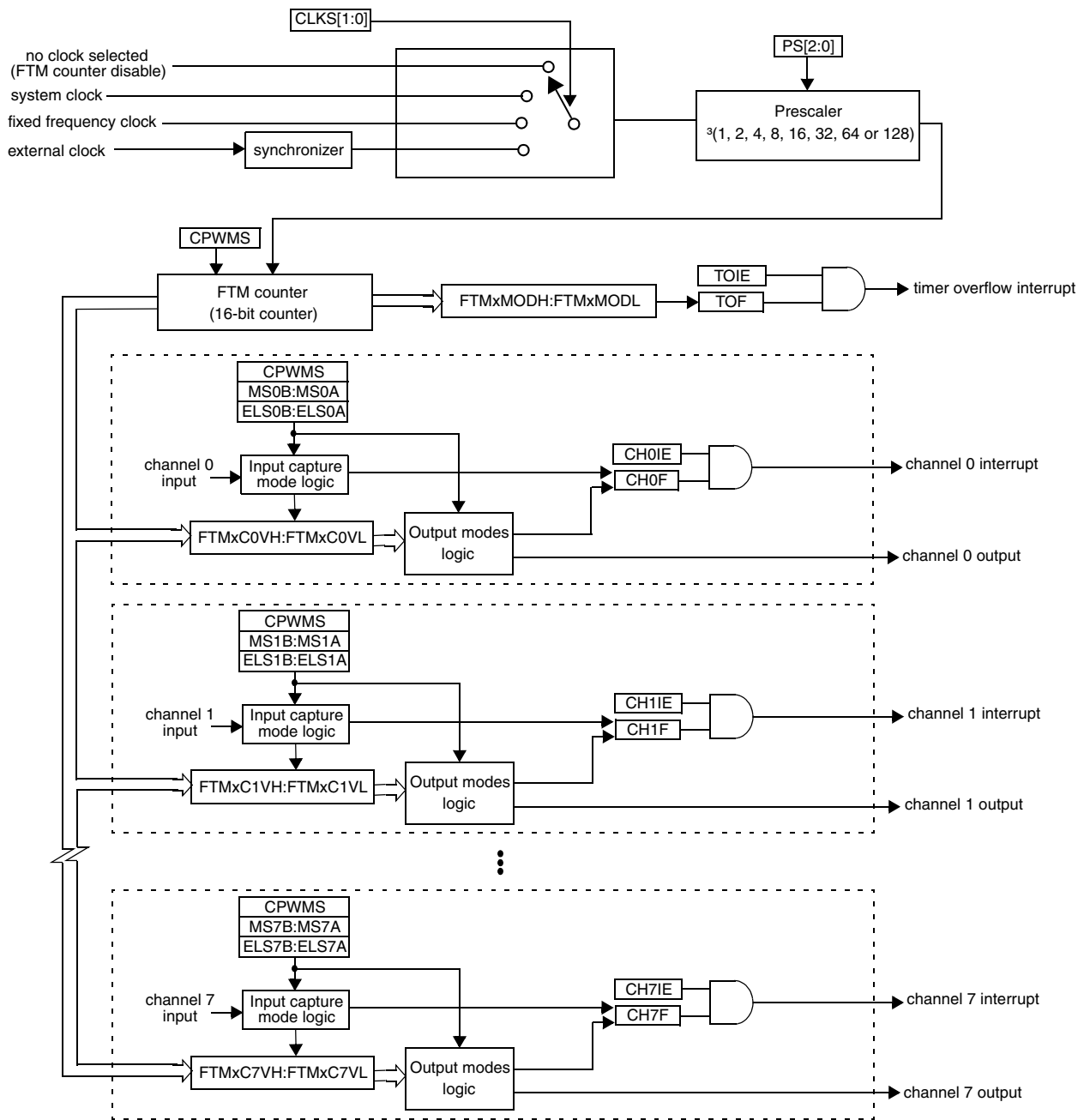


Figure 18-2. FTM Block Diagram

18.2 Signal Description

Table 18-1 shows the user-accessible signals for the FTM.

**Table 18-1. Signal Properties**

Name	Function
EXTCLK	FTM external clock – FTM external clock can be selected to drive the FTM counter.
FTMCHn <sup>1</sup>	FTM channel (n) – I/O pin associated with FTM channel (n).

<sup>1</sup> n = channel number (0 to 7)

### 18.2.1 EXTCLK — FTM External Clock

The external clock input signal is used as the FTM counter clock if selected by CLKS[1:0] bits in the FTMSC register. This clock signal must not exceed 1/4 of system clock frequency. The FTM counter prescaler selection and settings are also used when an external clock is selected.

### 18.2.2 FTMCHn — FTM Channel (n) I/O Pin

Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel. Please see [Table 18-7](#) for the control registers that define the channel modes.

## 18.3 Memory Map and Register Definition

This section provides a detailed description of all FTM registers accessible to the end user.

### 18.3.1 Module Memory Map

This section presents a high-level summary of the FTM registers and how they are mapped. Table 18-2 shows the registers for an 8-channel FTM. Register names include the FTM number (x) because it is common for MCU systems to include more than one FTM.

**Table 18-2. 8-channel FTM Module Memory Map**

Address	Use	Access	Section/Page
Base+0x0000	FTM Status and Control (FTMSC)	R/W	<a href="#">18.3.3/-455</a>
Base+0x0001	FTM Counter Register High (FTMCNTH)	R <sup>1</sup>	<a href="#">18.3.4/-456</a>
Base+0x0002	FTM Counter Register Low (FTMCNTL)	R <sup>1</sup>	<a href="#">18.3.4/-456</a>
Base+0x0003	FTM Modulo Register High (FTMMODH)	R/W	<a href="#">18.3.5/-457</a>
Base+0x0004	FTM Modulo Register Low (FTMMODL)	R/W	<a href="#">18.3.5/-457</a>
Base+0x0005	FTM Channel 0 Status and Control (FTMC0SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x0006	FTM Channel 0 Value High (FTMC0VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0007	FTM Channel 0 Value Low (FTMC0VL)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0008	FTM Channel 1 Status and Control (FTMC1SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x0009	FTM Channel 1 Value High (FTMC1VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x000A	FTM Channel 1 Value Low (FTMC1VL)	R/W	<a href="#">18.3.7/-459</a>
Base+0x000B	FTM Channel 2 Status and Control (FTMC2SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x000C	FTM Channel 2 Value High (FTMC2VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x000D	FTM Channel 2 Value Low (FTMC2VL)	R/W	<a href="#">18.3.7/-459</a>
Base+0x000E	FTM Channel 3 Status and Control (FTMC3SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x000F	FTM Channel 3 Value High (FTMC3VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0010	FTM Channel 3 Value Low (FTMC3VL)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0011	FTM Channel 4 Status and Control (FTMC4SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x0012	FTM Channel 4 Value High (FTMC4VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0013	FTM Channel 4 Value Low (FTMC4VL)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0014	FTM Channel 5 Status and Control (FTMC5SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x0015	FTM Channel 5 Value High (FTMC5VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0016	FTM Channel 5 Value Low (FTMC5VL)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0017	FTM Channel 6 Status and Control (FTMC6SC)	R/W	<a href="#">18.3.6/-458</a>
Base+0x0018	FTM Channel 6 Value High (FTMC6VH)	R/W	<a href="#">18.3.7/-459</a>
Base+0x0019	FTM Channel 6 Value Low (FTMC6VL)	R/W	<a href="#">18.3.7/-459</a>

**Table 18-2. 8-channel FTM Module Memory Map (continued)**

Base+0x001A	FTM Channel 7 Status and Control (FTMC7SC)	R/W	18.3.6/-458
Base+0x001B	FTM Channel 7 Value High (FTMC7VH)	R/W	18.3.7/-459
Base+0x001C	FTM Channel 7 Value Low (FTMC7VL)	R/W	18.3.7/-459

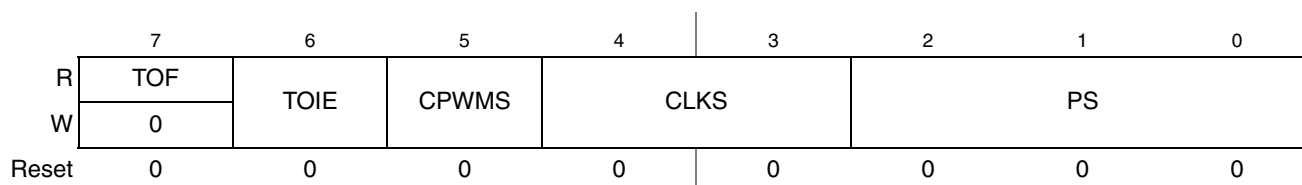
<sup>1</sup> Read-only for normal access. However, any write to FTMCNTH or FTMCNTL causes the 16-bit FTM counter to be updated with the value 0x0000.

## 18.3.2 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, FTMCnSC refers to FTM (x) and channel (n). FTM1C2SC is the status and control register for channel 2 of FTM1.

## 18.3.3 FTM Status and Control Register (FTMSC)

FTMSC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

**Figure 18-3. FTM Status and Control Register (FTMSC)****Table 18-3. FTMSC Field Descriptions**

Field	Description
7 TOF	Timer overflow flag bit. This read/write bit is set when the FTM counter passes the value in the FTMMODH:FTMMODL registers. The TOF bit is cleared by reading FTMSC register while TOF is set and then writing a logic 0 to TOF bit. If another FTM overflow occurs between the read and write operations, the write operation has no effect, therefore TOF remains set indicating an overflow has occurred. In this case a TOF interrupt request is not lost due to clearing sequence for a previous TOF. TOF bit is cleared out from reset. Writing a logic 1 to TOF has no effect. 0 FTM counter has not 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. 0 FTM counter operates in up counting mode. 1 FTM counter operates in up-down counting mode.

**Table 18-3. FTMSC Field Descriptions (continued)**

Field	Description
4–3 CLKS	Clock source selection. As shown in <a href="#">Table 18-4</a> , this 2-bit field is used to select one of the three FTM counter clock sources. Note that if CLKS[1:0] = 0:0, no clock is selected to the FTM counter which is equivalent as disabling the counter.
2–0 PS	Prescale factor selection. This 3-bit field selects one of 8 division factors for the clock source selected by CLKS[1:0] bits as shown in <a href="#">Table 18-5</a> . The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits.

**Table 18-4. FTM Clock Source Selection**

CLKS	FTM Clock Source to Prescaler Input
00	No clock selected (FTM counter disable)
01	System clock
10	Fixed frequency clock
11	External clock

**Table 18-5. 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

### 18.3.4 FTM Counter Registers (FTMCNTH:FTMCNTL)

The two read-only FTM counter registers contain the high and low bytes of the value in the FTM counter. Reading either byte (FTMCNTH or FTMCNTL) 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 (FTMSC).



Reset clears the FTM counter registers. Writing any value to FTMCNTH or FTMCNTL updates the FTM counter (FTMCNTH:FTMCNTL) with its initial value (0x0000) 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 FTMCNTH updates the 16-bit counter with its initial value (0x0000)							
Reset	0	0	0	0	0	0	0	0

**Figure 18-4. FTM Counter Register High (FTMCNTH)**

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to FTMCNTL updates the 16-bit counter with its initial value (0x0000)							
Reset	0	0	0	0	0	0	0	0

**Figure 18-5. FTM Counter Register Low (FTMCNTL)**

When BDM is active, the FTM counter is frozen (this is the value that you may read); 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 you were in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution.

### 18.3.5 FTM Counter Modulo Registers (FTMMODH:FTMMODL)

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 18.4.3, “Counter”](#)).

Reset sets the FTM counter modulo registers to 0x0000.

Writing to either byte (FTMMODH or FTMMODL) latches the value into a buffer. The registers are updated with the value of their write buffer according to [Section 18.4.8, “Load of the Registers With Write Buffers.”](#)

This write coherency mechanism may be manually reset by writing to the FTMSC 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								
Reset	0	0	0	0	0	0	0	0

Figure 18-6. FTM Counter Modulo Register High (FTMMODH)

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

Figure 18-7. FTM Counter Modulo Register Low (FTMMODL)

It is recommended to initialize the FTM counter (write to FTMCNTH or FTMCNTL) before writing to the FTM modulo registers to avoid confusion about when the first counter overflow will occur.

### 18.3.6 FTM Channel (n) Status and Control Register (FTMCnSC)

FTMCnSC 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 18-8. FTM Channel (n) Status and Control Register (FTMCnSC)

Table 18-6. FTMCnSC Field Descriptions

Field	Description
7 CHnF	Channel (n) flag bit. This read/write bit is set when an event occurs on channel (n). The CHnF bit is cleared by reading FTMCnSC register while CHnF is set and then writing a logic 0 to CHnF bit. If another event occurs between the read and write operations, the write operation has no effect, therefore CHnF remains set indicating an event has occurred. In this case a CHnF interrupt request is not lost due to clearing sequence for a previous CHnF. CHnF bit is cleared out from reset. Writing a logic 1 to CHnF has no effect. 0 No channel (n) event occurred. 1 Channel (n) event occurred.
6 CHnIE	Channel (n) interrupt enable. This read/write bit enables interrupts from channel (n). Reset clears CHnIE. 0 Channel (n) interrupt requests disabled (use software polling). 1 Channel (n) interrupt requests enabled.
5 MSnB	Mode select B bit for FTM channel (n). This bit is used for further selections in the channel (n) logic. Its functionality is dependent on the channel mode. Refer to the <a href="#">Table 18-7</a> for details.

**Table 18-6. FTMCnSC Field Descriptions (continued)**

Field	Description
4 MSnA	Mode select A bit for FTM channel (n). This bit is used for further selections in the channel (n) logic. Its functionality is dependent on the channel mode. Refer to the <a href="#">Table 18-7</a> for details.
3–2 ELSnB ELSnA	Edge or level selection bits. The functionality of ELSnB and ELSnA bits depends on the channel (n) mode as shown in <a href="#">Table 18-7</a> .

**Table 18-7. Mode, Edge, and Level Selection**

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for FTM — revert the channel pin to general purpose I/O or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on 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)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on match-up)
		X1		Low-true pulses (set output on match-up)

### 18.3.7 FTM Channel Value Registers (FTMCnVH:FTMCnVL)

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.

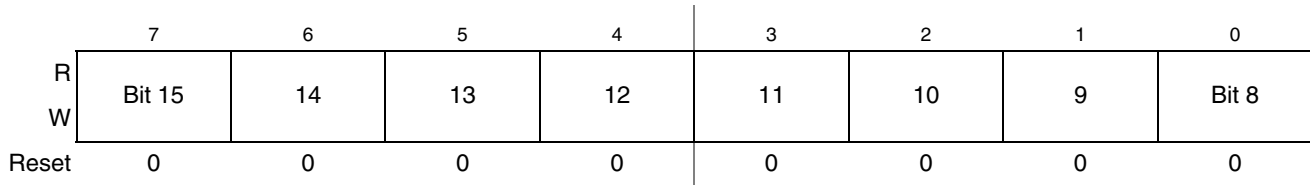


Figure 18-9. FTM Channel Value Register High (FTMCnVH)

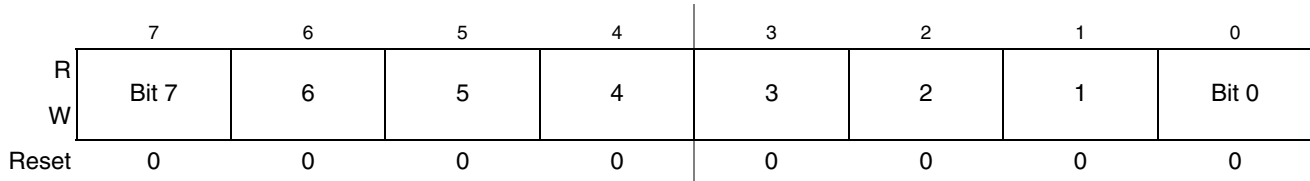


Figure 18-10. FTM Channel Value Register Low (FTMCnVL)

In input capture mode, reading either byte (FTMCnVH or FTMCnVL) 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 FTMCnSC register is written (whether BDM mode is active or not). Any write to the channel registers is 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 you were in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution. Any read of the FTMCnVH and FTMCnVL 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 (FTMCnVH or FTMCnVL) latches the value into a buffer. The registers are updated with the value of their write buffer according to [Section 18.4.8, “Load of the Registers With Write Buffers.”](#)

This write coherency mechanism may be manually reset by writing to the FTMCnSC 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 FTMCnVH and FTMCnVL 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 (while BDM was not active).

## 18.4 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 18-11.

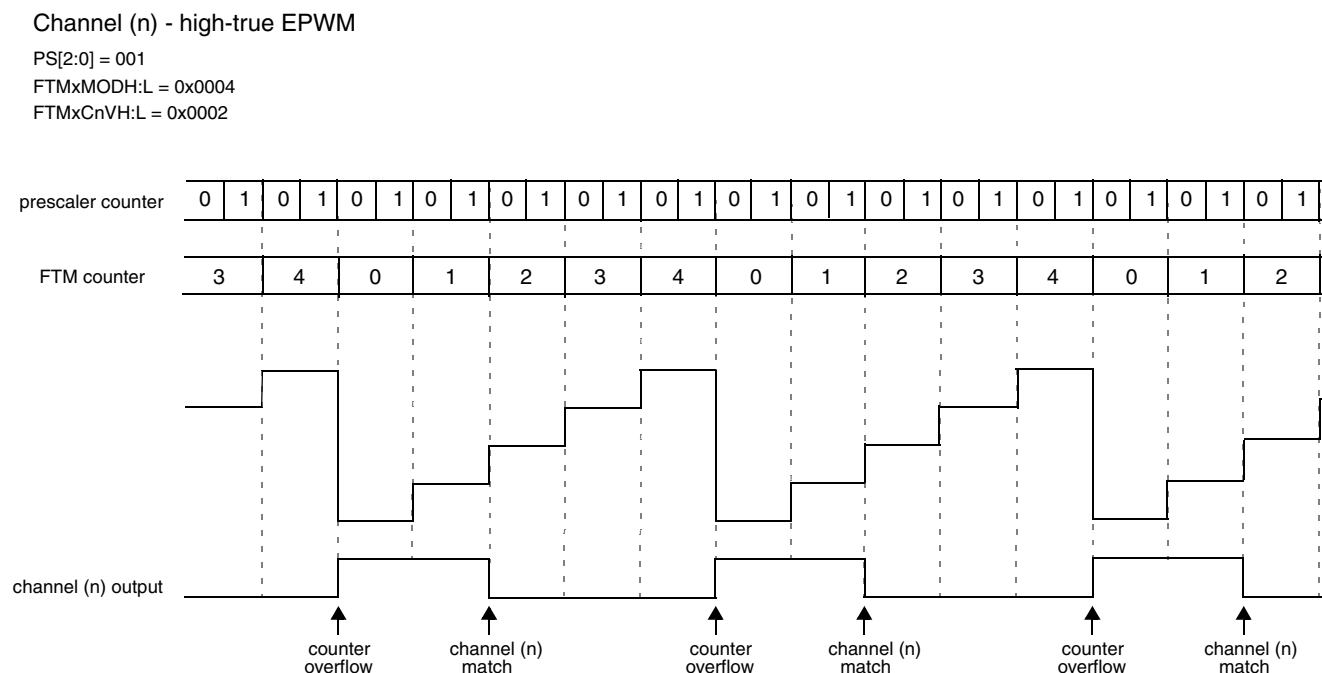


Figure 18-11. Notation Used in the FTM Block Guide

## 18.4.1 Clock Source

FTM module has only one clock domain that is the system clock.

### 18.4.1.1 Counter Clock Source

The CLKS[1:0] bits in the FTMSC register select one of three possible clock sources for the FTM counter or disable the FTM counter, see Table 18-8. After any MCU reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration. Refer the chip specific documentation for further information. Due to FTM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed the system clock frequency. The fixed frequency clock has no limitations for lower frequency operation.

The external clock passes through a synchronizer clocked by the system clock to assure that counter transitions are properly aligned to system clock transitions. Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

### 18.4.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits (Table 18-5). Figure 18-12 shows an example of the prescaler counter and FTM counter.

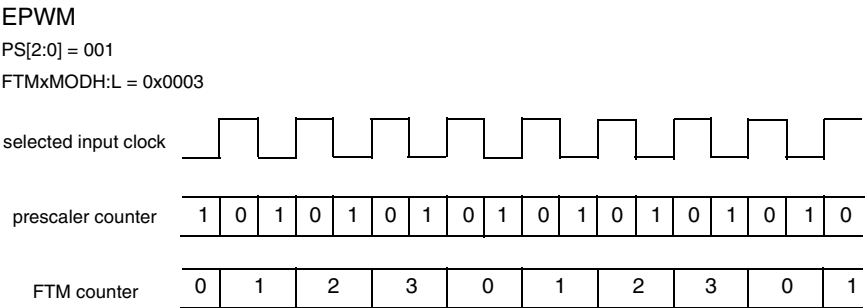


Figure 18-12. Example of the Prescaler Counter

### 18.4.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler (Section 18.4.2, “Prescaler”).

The FTM counter has two modes of operation: up or up-down counter according to the CPWMS bit.

#### 18.4.3.1 Up Counting

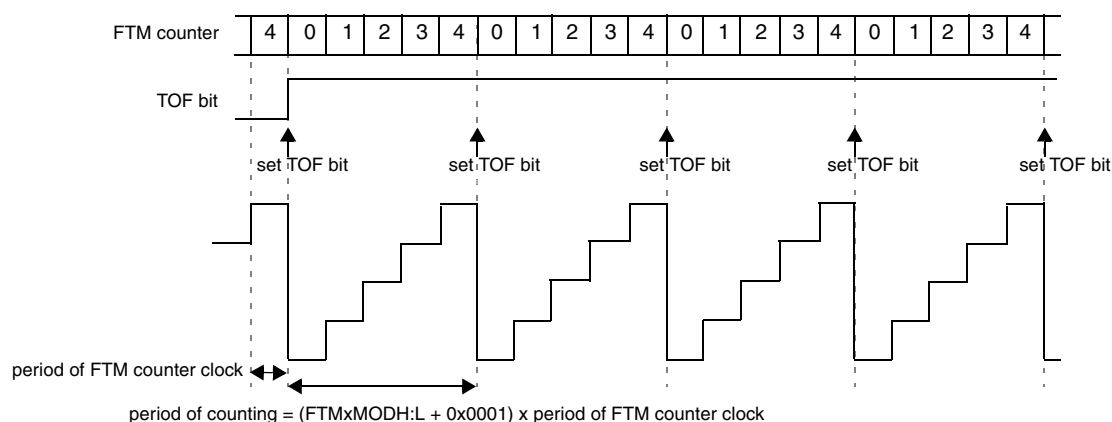
Up counting is selected when (CPWMS = 0).

The starting value of the count is 0x0000 and FTMxMODH:FTMxMODL defines the final value of the count (Figure 18-13). The value 0x0000 is loaded into the FTM counter, and the counter increments until the value of FTMxMODH:FTMxMODL is reached, at which point the counter is reloaded with the value 0x0000.

The FTM period when using up counting is  $(\text{FTMxMODH:FTMxMODL} + 0x0001) \times \text{period of the FTM counter clock}$ .

The TOF bit is set when the FTM counter changes from FTMxMODH:FTMxMODL to 0x0000.

FTM counting is up (CPWMS = 0)  
FTMxMODH:L = 0x0004



**Figure 18-13. Example of FTM Up Counting**

### 18.4.3.2 Up-Down Counting

Up-down counting is selected when (CPWMS = 1).

The starting value of the count is 0x0000 and FTMxMODH:FTMxMODL defines the final value of the count (Figure 18-14). The value 0x0000 is loaded into the FTM counter, and the counter increments until the value of FTMxMODH:FTMxMODL is reached, at which point the counter is decremented until it returns to the value 0x0000 and the up-down counting restarts.

The FTM period when using up-down counting is  $2 \times \text{FTMxMODH:FTMxMODL} \times \text{period of the FTM counter clock}$ .

The TOF bit is set when the FTM counter changes from FTMxMODH:FTMxMODL to (FTMxMODH:FTMxMODL - 1).

FTM counting is up-down (CPWMS = 1)  
FTMxMODH:L = 0x0004

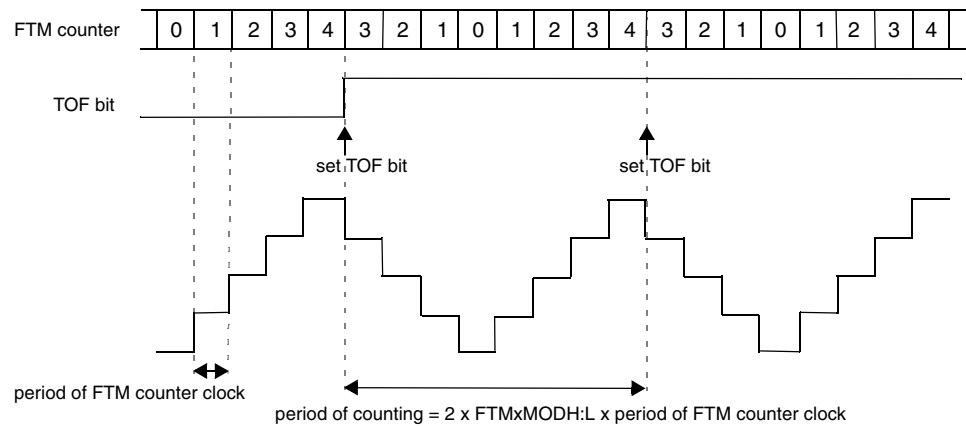


Figure 18-14. Example of Up-Down Counting

18.4.3.3 Free Running Counter

If (FTMxMODH:FTMxMODL = 0x0000 or 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 18-15).

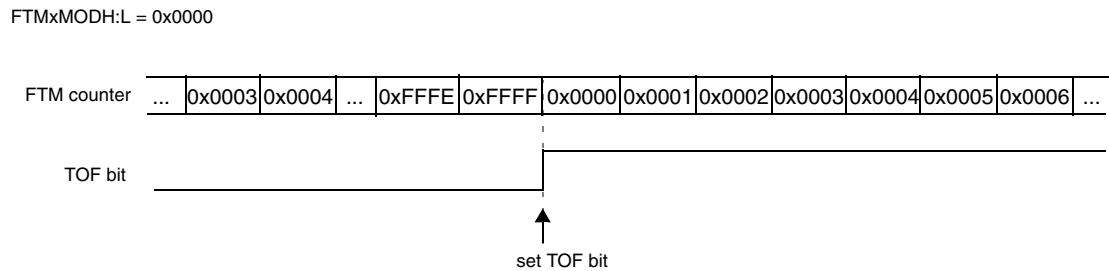


Figure 18-15. Example When the FTM Counter Is a Free Running

18.4.3.4 Counter Reset

Any write to FTMxCNTH or FTMxCNTL register resets the FTM counter to the value 0x0000 and the channels output to their initial value (except for channels in output compare mode).

18.4.4 Input Capture Mode

The input capture mode is selected when (CPWMS = 0), (MSnB:MSnA = 0:0), and (ELSnB:ELSnA ≠ 0:0).



When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the FTMxCnVH:FTMxCnVL registers, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1 (Figure 18-16).

When a channel is configured for input capture, the FTMxCHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

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. 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.

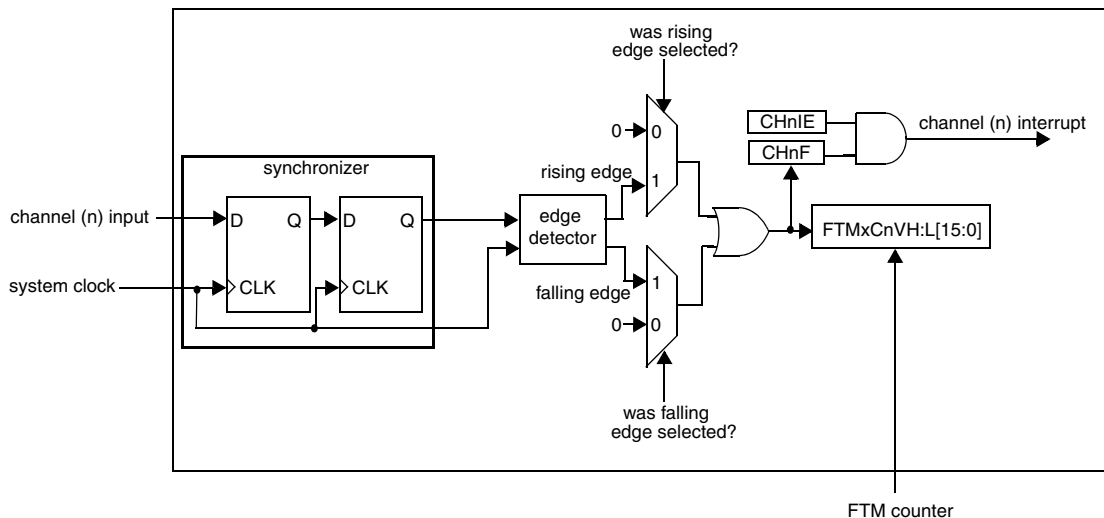


Figure 18-16. Input Capture Mode

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.

### 18.4.5 Output Compare Mode

The output compare mode is selected when (CPWMS = 0), and (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 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 CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = FTMxCnVH:FTMxCnVL).

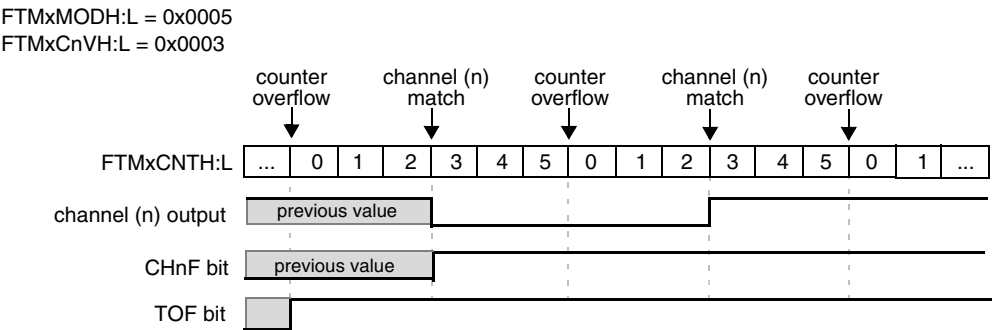


Figure 18-17. Example of the Output Compare Mode when the Match Toggles the Channel Output

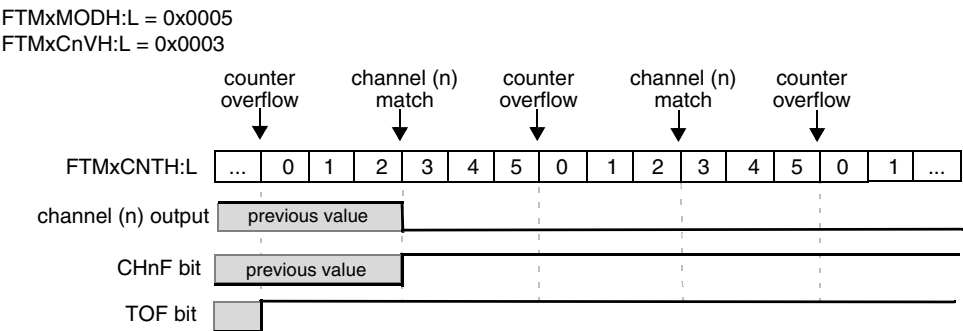


Figure 18-18. Example of the Output Compare Mode when the Match Clears the Channel Output

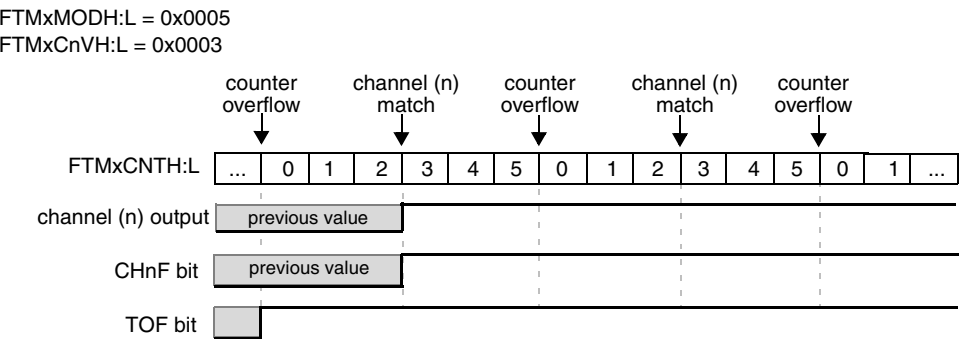


Figure 18-19. Example of the Output Compare Mode when the Match Sets the Channel Output

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.

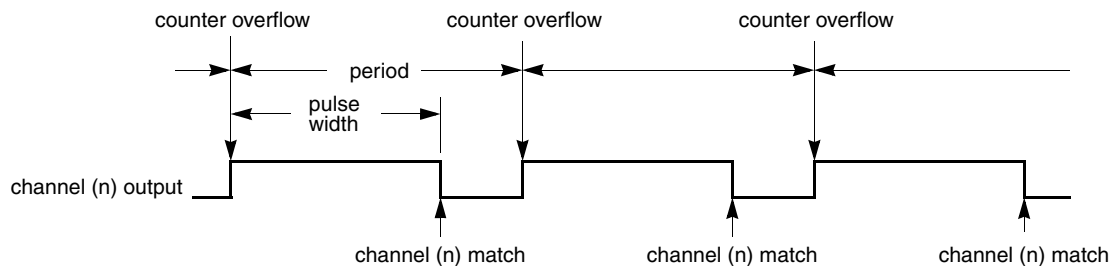
### 18.4.6 Edge-Aligned PWM (EPWM) Mode

The edge-aligned mode is selected when ( $CPWMS = 0$ ), and ( $MSnB = 1$ ).

The EPWM period is determined by ( $FTMxMODH:FTMxMODL + 0x0001$ ) and the pulse width (duty cycle) is determined by  $FTMxCnVH:FTMxCnVL$ .

The  $CHnF$  bit is set and the channel (n) interrupt is generated (if  $CHnIE = 1$ ) at the channel (n) match ( $FTM$  counter =  $FTMxCnVH:FTMxCnVL$ ), that is, at the end of the pulse width.

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.

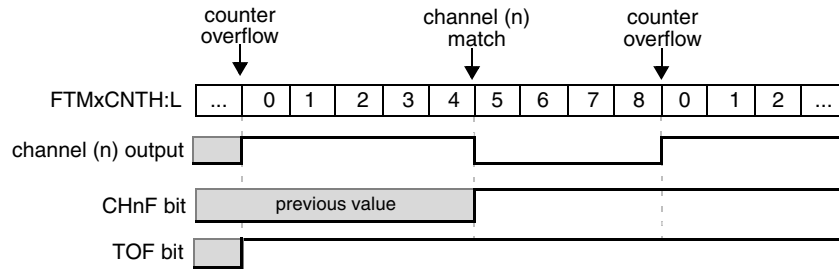


**Figure 18-20. EPWM Period and Pulse Width with  $ELSnB:ELSnA = 1:0$**

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.

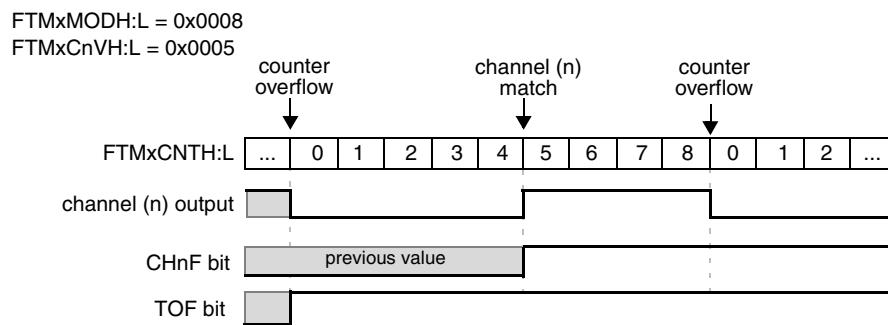
If ( $ELSnB:ELSnA = 1:0$ ), then the channel (n) output is forced high at the counter overflow (when the value  $0x0000$  is loaded into the FTM counter), and it is forced low at the channel (n) match (when the FTM counter =  $FTMxCnVH:FTMxCnVL$ ) (Figure 18-21).

FTMxMODH:L = 0x0008  
FTMxCnVH:L = 0x0005



**Figure 18-21. 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 value 0x0000 is loaded into the FTM counter), and it is forced high at the channel (n) match (when the FTM counter = FTMxCnVH:FTMxCnVL) (Figure 18-22).



**Figure 18-22. EPWM Signal with ELSnB:ELSnA = X:1**

If (FTMxCnVH:FTMxCnVL = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match. If (FTMxCnVH:FTMxCnVL > FTMxMODH:FTMxMODL), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match. Therefore, FTMxMODH:FTMxMODL must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

### 18.4.7 Center-Aligned PWM (CPWM) Mode

The center-aligned mode is selected when (CPWMS = 1).

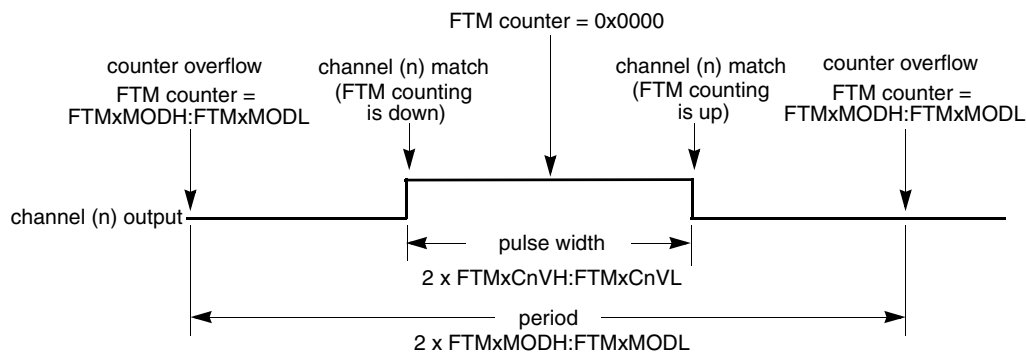
The CPWM pulse width (duty cycle) is determined by  $(2 \times \text{FTMxCnVH:FTMxCnVL})$  and the period is determined by  $(2 \times \text{FTMxMODH:FTMxMODL})$  (Figure 18-23). FTMxMODH:FTMxMODL must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches FTMxMODH:FTMxMODL and then counts down until it reaches the value 0x0000.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = FTMxCnVH:FTMxCnVL) when the FTM counting is down (at the begin of the pulse width) and when the FTM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value 0x0000.

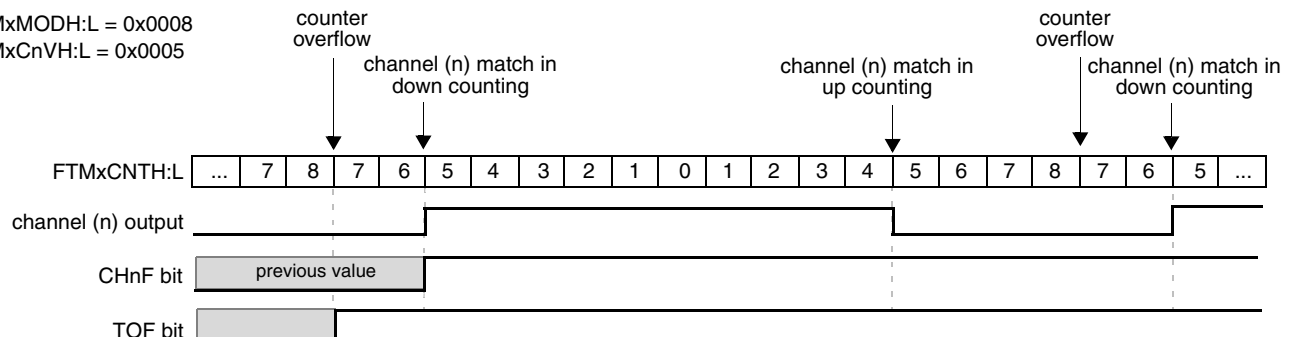
The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).



**Figure 18-23. 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 18-24).



**Figure 18-24. 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 18-25).

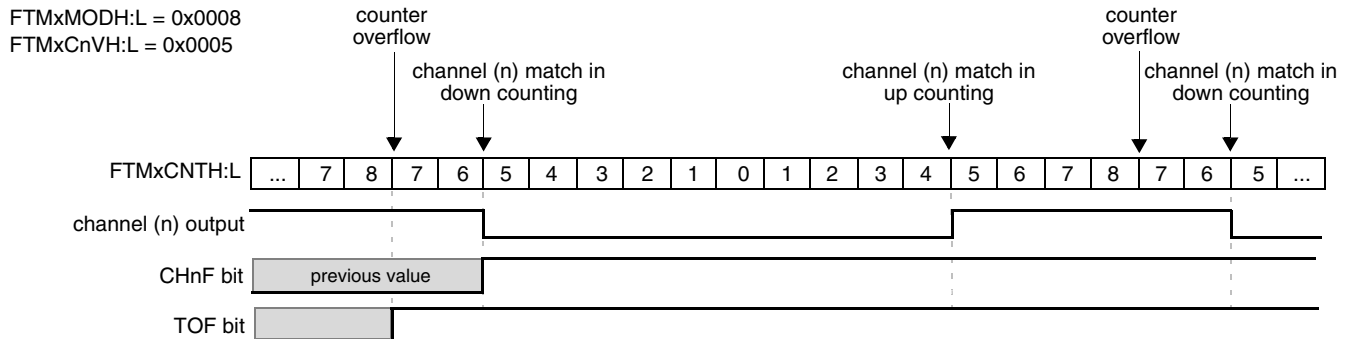


Figure 18-25. CPWM Signal with ELSnB:ELSnA = X:1

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 and CHnF bit is not set even when there is the channel (n) match.

If (FTMxCnVH:FTMxCnVL is a positive value, that is, FTMxCnVH[7] = 0), (FTMxCnVH:FTMxCnVL  $\geq$  FTMxMODH:FTMxMODL), and (FTMxMODH:FTMxMODL  $\neq$  0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by FTMxMODH:FTMxMODL is 0x0001 through 0x7FFE (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 must not be used when the FTM counter is a free running counter.

### 18.4.8 Load of the Registers With Write Buffers

If (CLKS[1:0] = 0:0), then FTMxMODH:L registers are updated when their second byte is written. If (CLKS[1:0]  $\neq$  0:0), then FTMxMODH:L registers are updated according to the CPWMS bit, that is:

- If the 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 0x0000. If the FTM counter is a free-running counter, then this update is made when the FTM counter changes from 0xFFFF to 0x0000.
- If the 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] = 0:0), then FTMxCnVH:L registers are updated when their second byte is written. If (CLKS[1:0]  $\neq$  0:0), then FTMxCnVH:L registers are updated according to the selected mode, that is:

- If the 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 the 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 0x0000. If the FTM counter is a free running counter, then this update is made when the FTM counter changes from 0xFFFF to 0x0000.
- If the 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).

### 18.4.9 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 case.

- Write any value to FTMxCNTH or FTMxCNTL registers ([Section 18.4.3.4, “Counter Reset”](#)) resets the FTM counter to the value 0x0000 and the channels output to their initial value (except for channels in output compare mode).

## 18.5 Reset Overview

The FTM is reset whenever any MCU reset occurs.

## 18.6 FTM Interrupts

### 18.6.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

### 18.6.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).





# Chapter 19

## Programmable Reference Analog Comparator (S08PRACMPV1)

### 19.1 Introduction

The PRACMP is a CMOS comparator with a programmable reference input. The comparator has up to eight input pins, each of them can be compared with any input pins. There is also a internal programmable reference generator which divides the  $V_{In}$  into 32 levels, the  $V_{In}$  can be selected from two external sources. Output of this reference generator can be one of the eight inputs of comparator. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

#### NOTE

For MC9S08GW64 Series, the  $V_{in1}$  or external reference is supplied via the  $V_{DDA}$  pin and the  $V_{in2}$  or internal reference is supplied from the PMC analog supply.

#### 19.1.1 PRACMP Input Pins CMPPx and Output pin CMPOUTx

There are three PRACMP modules (PRACMP0, PRACMP1 and PRACMP2) and seven PRACMP inputs (CMPPx, x = 0, 1 ... 6), three outputs (CMPOUTx, x = 0, 1, 2). CMPPx are shared among the three PRACMPx modules. Each PRACMPx module can select any of CMPPx inputs as its inputs. CMPOUTx belongs to the specific PRACMPx module.

The PRACMPx input pin enable register (PRACMPxC2) and output enable bit (ACOPE) in the PRACMPxC2 register are not valid. The PRACMPx input and output pins are controlled in the Pin Muxing Control block. Please refer to the [Section 6.7, “Pin Mux Controls,”](#) for more information.

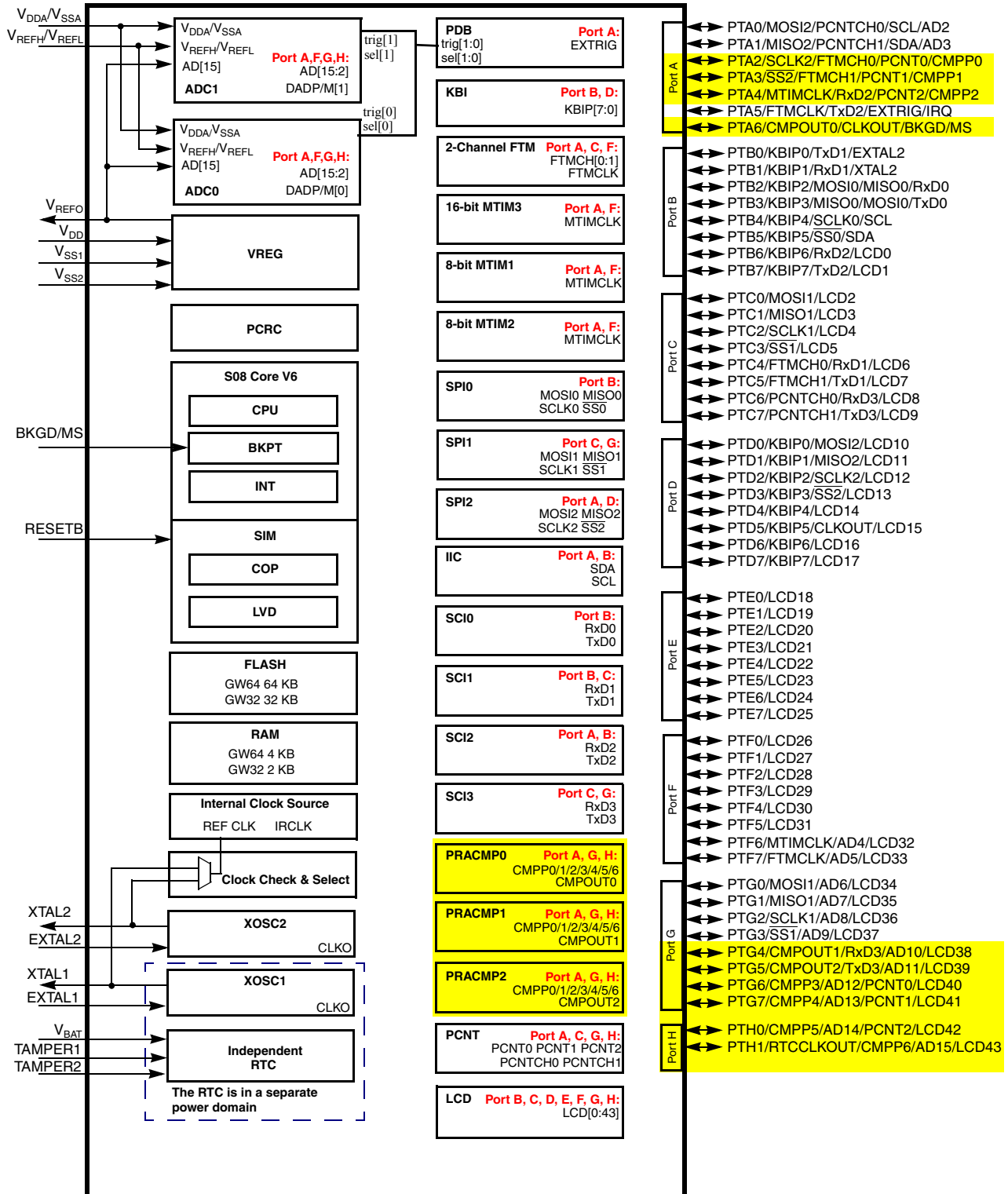
#### 19.1.2 PRACMP Internal Connection

The PRACMP1 and PRACMP2 output can be connected internally to the SCI1 RxD1 and SCI2 RxD2 input for infrared reception implementation. See [Section 2.3.10, “Interfacing the SCIs to Off-Chip Circuits,”](#) for more information.

The PRACMP0, PRACMP1 and PRACMP2 outputs can be connected internal to PCNT0, PCNT1 and PCNT2 correspondingly. See [Section 2.3.11, “PCNT\[2:0\],”](#) for more information.

### 19.1.3 PRACMP Clock Gating

The bus clock to the PRACMPx module can be gated on and off using the PRACMPx bit in SCGC5. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the PRACMPx bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.



**Figure 19-1. MC9S08GW64 Series Block Diagram Highlighting PRACMP Modules and Pins**

## 19.1.4 Features

PRACMP features include:

- On-chip programmable reference generator output ( $1/32 V_{in}$  to  $V_{in}$ , step is  $1/32 V_{in}$ ,  $V_{in}$  can be selected from external Vdd and internal regulated Vdd)
- Typically 5 mV of input offset
- Less than 40  $\mu A$  in enable mode and less than 1 nA in disable mode (excluding programmable reference generator)
- Fixed ACMP hysteresis which is from 3 mV to 20 mV
- Up to eight selectable comparator inputs; each input can be compared with any input by any polarity sequence
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output
- Remains operational in stop3 mode
- Be compatible with both vlp pmc and no-vlp pmc
- Comparator can be selected to use a hardware trigger to turn on/off

## 19.1.5 Modes of Operation

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

### 19.1.5.1 Operation in Wait Mode

The continues to operate in wait mode if enabled. The interrupt can wake up the MCU if enabled.

### 19.1.5.2 Operation in Stop Mode

The PRACMP (including PRG and ACMP) continues to operate in stop3 mode if enabled. If ACIEN is set, a PRACMP interrupt still can be generated to wake the MCU up from stop3 mode.

If the stop3 is exited by an interrupt, the PRACMP remains the setting before entering the stop. If stop3 is exited with a reset, the PRACMP goes into its reset.

The user should turn it off if its output is not used as a reference input of ACMP to save power, because the PRG consumes additional power.

In stop2 mode, the PRACMP is shut down completely. Any waking up from stop2 and stop1 brings PRACMP to its reset state.

### 19.1.5.3 Operation in Background Mode

When the MCU is in active background debug mode, the PRACMP continues operating normally.

## 19.1.6 Block Diagram

The block diagram of the PRACMP module is shown in Figure 19-2.

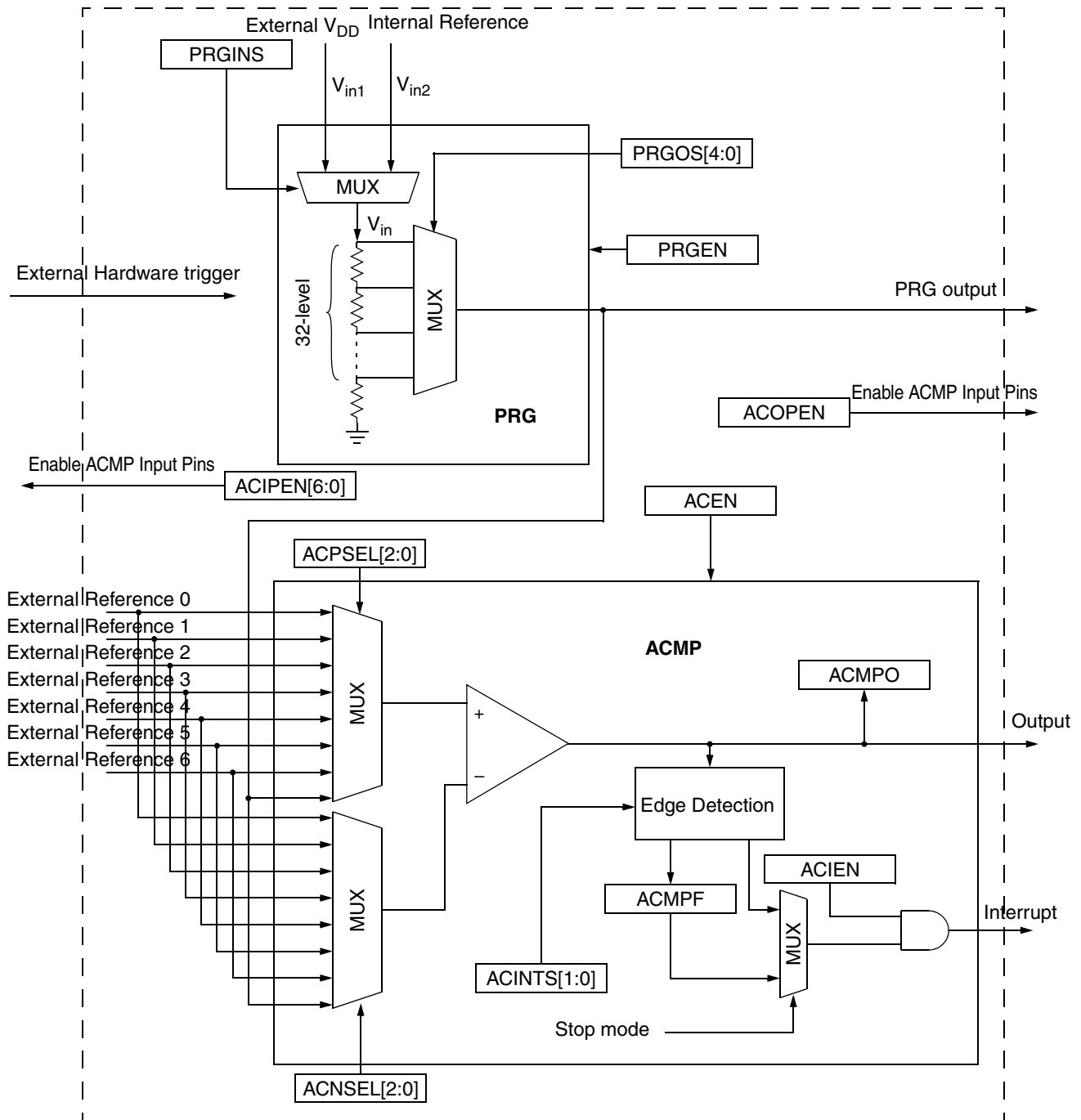


Figure 19-2. PRACMP Block Diagram

## 19.2 External Signal Description

The output of PRACMP can also be mapped to an external pin. When the output is mapped to an external pin, register bit ACOPE controls the pin to enable/disable the PRACMP output function.

## 19.3 Memory Map and Register Definition

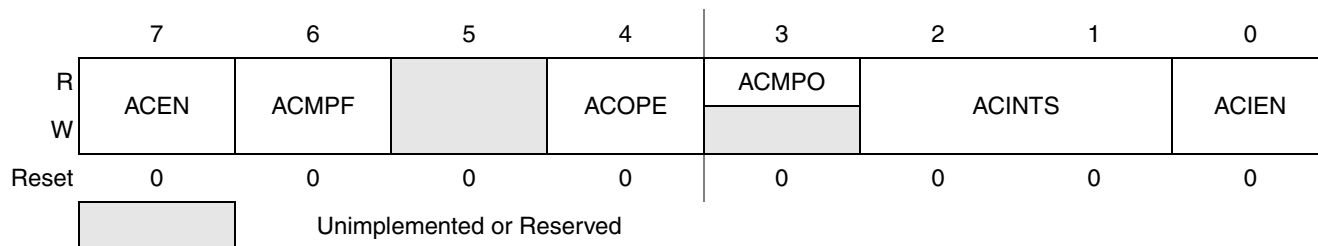
Table 19-1 is the memory map of the programmable reference analog comparator (PRACMP).

**Table 19-1. Module Memory Map**

Address	Use	Access
Base + \$0	PRACMP Control and Status Register (PRACMPxCS)	Read/Write
Base + \$1	PRACMP Control Register 0 (PRACMPxC0)	Read/Write
Base + \$2	PRACMP Control Register 1 (PRACMPxC1)	Read/Write
Base + \$3	PRACMP Control Register 2 (PRACMPxC2)	Read/Write

Refer to the direct-page register summary in the memory chapter of this reference manual for the absolute address assignments for all PRACMP registers.

### 19.3.1 PRACMP Control and Status Register (PRACMPxCS)



**Figure 19-3. PRACMPx Control and Status Register (PRACMPxCS)**

**Table 19-2. PRACMPxCS Descriptions**

Field	Description
7 ACEN	<b>ACMP enable control bit</b> <sup>1</sup> 0 The ACMP is disabled 1 The ACMP is enabled
6 ACMPF	<b>ACMP Interrupt Flag Bit</b> — Synchronously set by hardware when ACMP output has a valid edge defined by ACINTS[1:0]. The setting of this bit lags the ACMPPO 2 bus clocks. Clear ACMPF bit by writing a 0 to this bit. Writing a 1 to this bit has not effect.
4 ACOPE	<b>ACMP Output Pin Enable</b> — ACOPE enables the pad logic so that the output can be placed onto an external pin 0 Output of ACMP can't be placed onto external pin 1 Output of ACMP can be placed onto external pin
3 ACMPPO	<b>ACMP Output Bit</b> — ACMP output is synchronized by bus clock to form this bit. It changes following the ACMP output. This bit is a read only bit. Set when the output of the ACMP is high Cleared when the output of the ACMP is low After any reset or when the ACMP is disabled, this bit is read as 0.

Table 19-2. PRACMPxCS Descriptions (continued)

Field	Description
2:1 ACINTS [1:0]	<b>ACMP Interrupt Select</b> — Determines the sensitivity modes of the interrupt trigger. 00 ACMP interrupt on output falling or rising edge 01 ACMP interrupt on output falling edge 10 ACMP interrupt on output rising edge 11 Reserved
0 ACIEN	<b>ACMP Interrupt Enable</b> — Enables an ACMP CPU interrupt 1 Enable the ACMP Interrupt 0 Disable the ACMP Interrupt

<sup>1</sup> Every time to enable the PRACMP module, it is required the software to clear the ACMPF flag prior to the normal function application.

### 19.3.2 PRACMP Control Register 0 (PRACMPxC0)

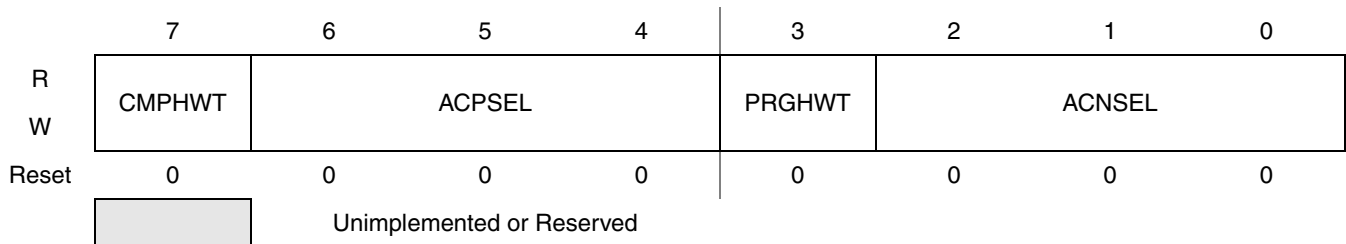


Figure 19-4. PRACMPxControl Register 0 (PRACMPxC0)

Table 19-3. PRACMPxC0 Field Descriptions

Field	Description
7 CMPHWT	<b>ACMP Hardware Trigger Select Bit</b> — ACHWT selects the external hardware trigger workwith ACEN bit to control the on/off for the ACMP block. When ACHWT is set, only if ACEN is set and hardware trigger is high can the comparator work. 0 Bypass the hardware trigger control 1 The hardware control to comparator is enabled
6:4 ACPSEL[2:0]	<b>ACMP Positive Input Select</b> 000 External reference 0 001 External reference 1 010 External reference 2 011 External reference 3 100 External reference 4 101 External reference 5 110 External reference 6 111 Internal PRG output

Table 19-3. PRACMPxC0 Field Descriptions

Field	Description
3 PRGHWT	<b>PRG Hardware Trigger Select Bit</b> — PRGHWT selects the external hardware trigger workwith PRGEN bit to control the on/off for the Porgrammable Reference Voltage Generator(DAC) block. When PRGHWT is set, only if PRGEN is set and hardware tirtgger is high can the DAC work. 0 Bypass the hardware trigger control 1 The hardware control to comparator is enabled
2:0 ACNSEL[2:0]	<b>ACMP Negative Input Select</b> 000 External reference 0 001 External reference 1 010 External reference 2 011 External reference 3 100 External reference 4 101 External reference 5 110 External reference 6 111 Internal PRG output

### 19.3.3 PRACMP Control Register 1 (PRACMPxC1)

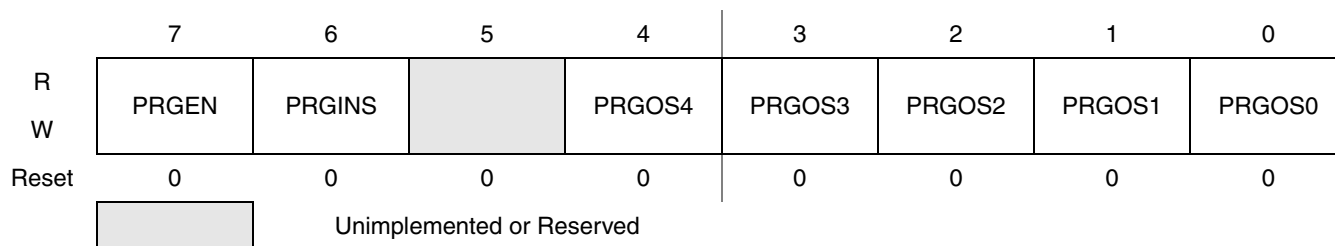


Figure 19-5. PRACMPx Control Register 1 (PRACMPxC1)

Table 19-4. PRACMPxC1 Field Descriptions

Field	Description
7 PRGEN	<b>Programmable Reference Generator Enable</b> — The PRGEN bit starts the Programmable Reference Generator operation. 0 The PRG system is disabled 1 The PRG system is enabled
6 PRGINS	<b>Programmable Reference Generator Input Selection</b> 0 The PRG selects $V_{in2}$ (internal reference, ie, internal regulated power supply or other on-chip peripherals) as the reference voltage 1 The PRG selects $V_{in1}$ (external reference, ie, external power supply) as the reference voltage
4:0 PRGOS[4:0]	<b>Programmable Reference Generator Output Selection</b> — The output voltage is selected by the following formula: $V_{output} = (V_{in}/32) \times (PRGOS[4:0] + 1)$ The $V_{output}$ range is from $V_{in}/32$ to $V_{in}$ , the step is $V_{in}/32$



Table 19-5 list the output configuration of programmable reference generator (PRG).

**Table 19-5. PRG Out Configuration**

PRGOS[4:0]	Output Voltage of PRG
00000	$1V_{In}/32$
00001	$2V_{In}/32$
00010	$3V_{In}/32$
00011	$4V_{In}/32$
00100	$5V_{In}/32$
00101	$6V_{In}/32$
00110	$7V_{In}/32$
00111	$8V_{In}/32$
01000	$9V_{In}/32$
01001	$10V_{In}/32$
01010	$11V_{In}/32$
01011	$12V_{In}/32$
01100	$13V_{In}/32$
01101	$14V_{In}/32$
01110	$15V_{In}/32$
01111	$16V_{In}/32$
10000	$17V_{In}/32$
10001	$18V_{In}/32$
10010	$19V_{In}/32$
10011	$20V_{In}/32$
10100	$21V_{In}/32$
10101	$22V_{In}/32$
10110	$23V_{In}/32$
10111	$24V_{In}/32$
11000	$25V_{In}/32$
11001	$26V_{In}/32$
11010	$27V_{In}/32$
11011	$28V_{In}/32$
11100	$29V_{In}/32$
11101	$30V_{In}/32$
11110	$31V_{In}/32$
11111	$V_{In}$

### 19.3.4 PRACMP Control Register 2 (PRACMPxC2)

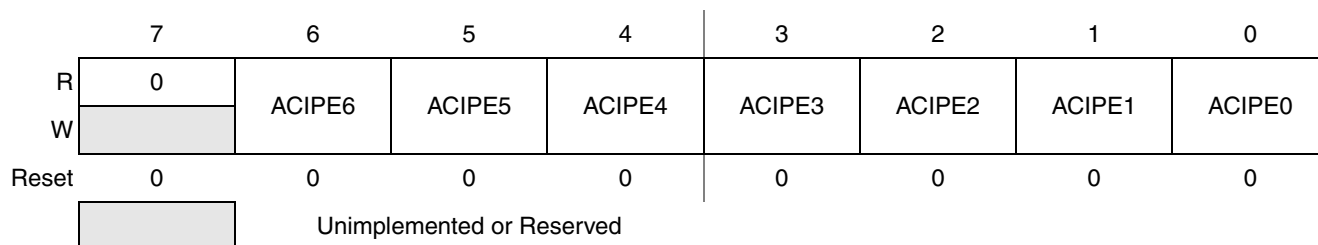


Figure 19-6. PRACMPx Control Register 2 (PRACMPxC2)

Table 19-6. PRACMPxC2 Field Descriptions

Field	Description
6:0 ACIPE6:ACIPE0	ACMP Input Pin Enable— This 7-bit register controls if the corresponding PRACMP external pin can be driven an analog input. 0 The corresponding external analog input is not allowed 1 The corresponding external analog input is allowed

## 19.4 Functional Description

The PRACMP module is functionally composed of two parts: programmable reference generator (PRG) and analog comparator (ACMP).

The programmable reference generator (PRG) includes a 32-level DAC (digital to analog convertor) and relevant control logic. PRG can select one of two reference inputs,  $V_{in1}$  (external Vdd) or  $V_{in2}$  (internal regulated Vdd), as the DAC input  $V_{in}$  by setting PRGINS bit of PRACMPxC1. After the DAC is enabled, it converts the data set in PRGOS[4:0] bits of PRACMPxC1 to a stepped analog output which is fed into ACMP as an internal reference input. This stepped analog output is also mapped out of the module. The output voltage range is from  $V_{in}/32$  to  $V_{in}$ . The step size is  $V_{in}/32$ .

The ACMP can achieve the analog comparison between positive input and negative input, and then give out a digital output and relevant interrupt. Both the positive and negative input of ACMP can be selected from the eight common inputs: seven external reference inputs and one internal reference input from the PRG output. The positive input of ACMP is selected by ACPSEL[2:0] bits of PRACMPxC0 and the negative input is selected by ACNSEL[2:0] bits of PRACMPxC0. Any pair of the eight inputs can be compared by configuring the PRACMPxC0 with the appropriate value.

After the ACMP is enabled by setting ACEN in PRACMPxC5, the comparison result appears as a digital output. Whenever a valid edge defined in ACINTS[1:0] occurs, the ACMPO bit in PRACMPxC5 register is asserted. If ACIEN is set, a PRACMP CPU interrupt occurs. The valid edge is defined by ACINTS[1:0]. When ACINTS[1:0] = 00, both the rising edge and falling edge on the ACMP output are valid. When ACINTS[1:0] = 01, only the falling edge on ACMP output is valid. When ACINTS[1:0] = 10, only rising edge on ACMP output is valid. ACINTS[1:0] = 11 is reserved.

The ACMP output is synchronized by the bus clock to generate ACMPO bit in PRACMPxC5 so that the CPU can read the comparison. In stop3 mode if the output of ACMP is changed, ACMPO can't be updated in time. The output can be synchronized and the ACMPO bit can be updated upon the waking up of the

CPU because of the availability of the bus clock. The ACMPO changes following the comparison result, so it can serve as a tracking flag that continuously indicates the voltage delta on the inputs.

If a reference input external to the chip is selected as an input of ACMP, the corresponding ACIPE bit of PRACMPxC2 should be set to enable the input from pad interface. If the output of the ACMP needs to be put onto the external pin, the ACOPE bit of PRACMPxCS must enable the ACMP pin function of pad logic.

## 19.5 Setup and Operation of PRACMP

The two parts of PRACMP (PRG and ACMP) can be set up and operated independently. But if the PRG works as an input of the ACMP, the PRG must be configured before the ACMP is enabled.

Because the input-switching can cause problems on the ACMP inputs, the user should complete the input selection before enabling the ACMP and should not change the input selection setting when the ACMP is enabled to avoid unexpected output. Similarly, because the programmable reference generator (PRG) experiences a setup delay after the PRGOS[4:0] is changed, the user should complete the setting of PRGOS[4:0] before PRG is enabled.

## 19.6 Resets

During a reset the PRACMP is configured in the default mode. Both ACMP and PRG are disabled.

## 19.7 Interrupts

If the bus clock is available when a valid edge defined in ACINTS[1:0] occurs, the ACMPPF bit in PRACMPxCS register is asserted. If ACIEN is set, a PRACMP interrupt event occurs. The ACMPPF bit remains asserted until the PRACMP interrupt is cleared by software. When in stop3 mode, a valid edge on ACMP output generates an asynchronous interrupt which can wake the MCU up from stop3. The interrupt can be cleared by writing a 0 to the ACMPPF bit.



## Chapter 20

# Programmable Cyclic Redundancy Check (PCRCV1)

### 20.1 Introduction

Programmable cyclic redundancy check (PCRC) generates 16/32-bit CRC code for error detection. The PCRC can be configured to work as a standard CRC. It provides the user with programmable polynomial, SEED and other parameters required to implement a 16-bit or 32-bit CRC standard. These parameters are detailed in further sections.

The 16/32-bit code is calculated for 8-bit/16-bit of data at a time.

#### NOTE

After writing the data, there should be 1 bus clock cycle delay, between the last write and read of CRC checksum. This is due to calculation delay of CRC checksum.

#### 20.1.1 PCRC Clock Gating

The bus clock to the PCRC module can be gated on and off using the PCRC bit in SCGC4. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the PCRC bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

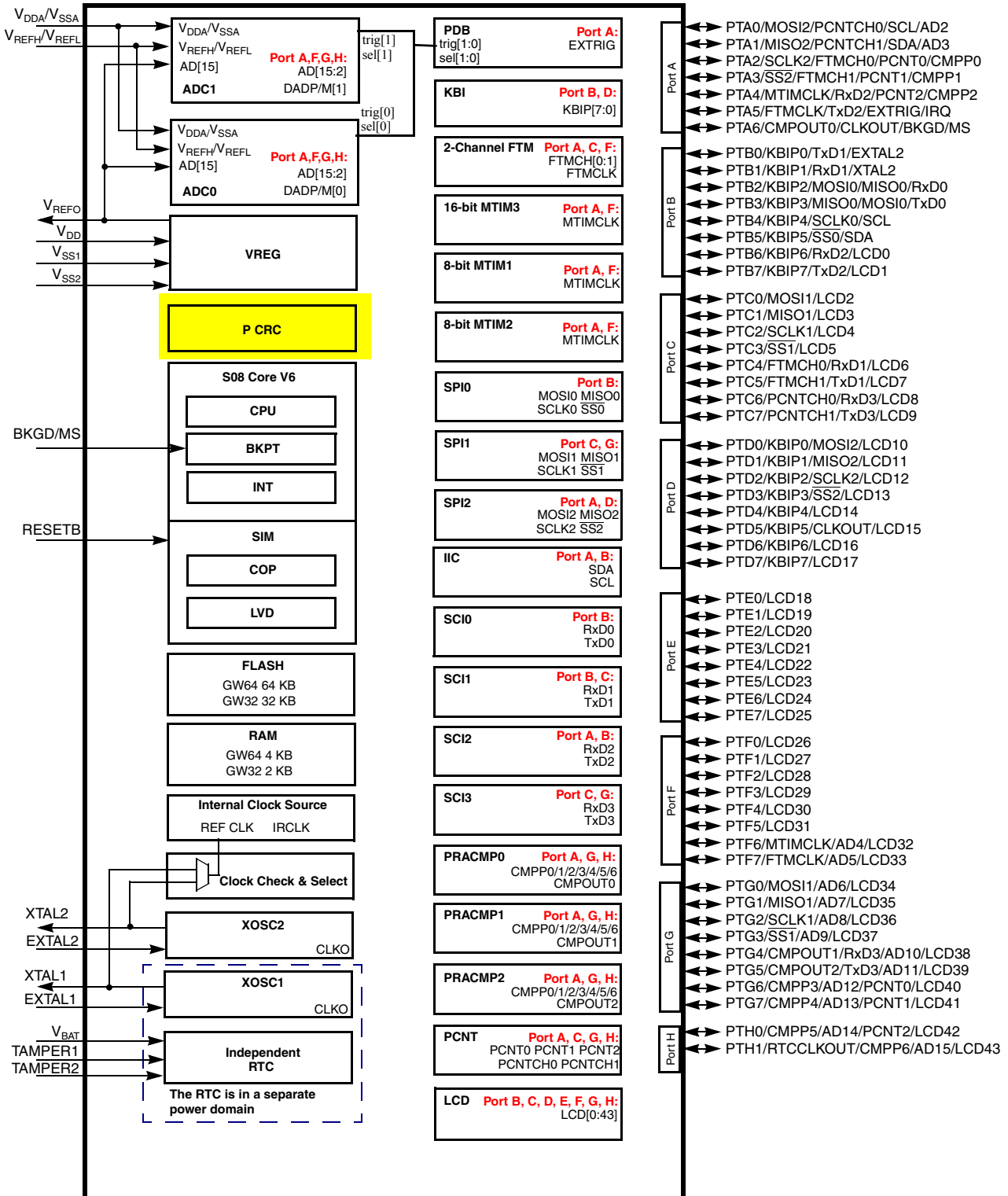


Figure 20-1. MC9S08GW64 Series Block Diagram Highlighting PCRC Module and Pins

## 20.1.2 Features

Features of the PCRC module are:

- Hardware 16/32-bit CRC generator.
- Programmable initial Seed Value.
- Programmable 16/32-bit Polynomial.
- Option to transpose input data or output data (i.e. CRC result) by bit or byte wise. This is required for certain CRC standards.
  - Byte wise transpose is not possible when accessing the CRC data register via 8-bit accesses. In this case, the user software must handle the byte wise transpose function.
- Option for Inversion of final CRC result.
- 8-bit CPU Register Programming Interface.

## 20.1.3 Block Diagram

Figure 20-2 shows the block diagram of PCRC.

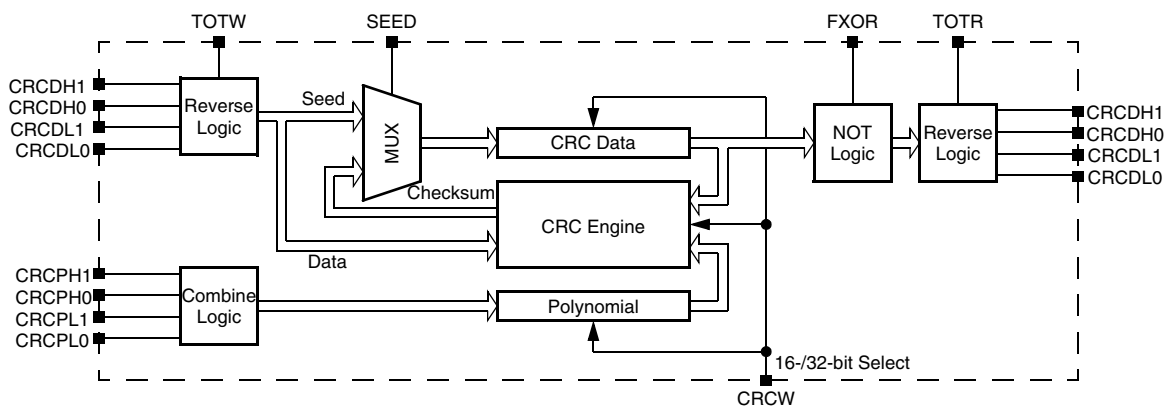


Figure 20-2. Programmable Cyclic Redundancy Check (PCRC) Block Diagram

## 20.1.4 Modes of Operation

A brief description of the PCRC in the various MCU modes is given here. Please refer to [Section 20.4, “Functional Description”](#) for more details.

### 20.1.4.1 Run Mode

This is the basic mode of operation.

### 20.1.4.2 Low Power Modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU enters in a low power mode which disables the module's clock. Calculation can resume after the clock is enabled or reset if the exit from low power mode generates a system reset. Clock gating for this module is MCU dependent.

## 20.2 External Signals Description

There is no CRC signal that connects off chip.

## 20.3 Memory Map and Register Definition

The following memory-mapped registers control and monitor operation of the PCRC.

**Table 20-1. Module Memory Map**

Address	Use	Access	Reset Value	Description
Base + \$00	CRCDH1	Read / Write	0xFF	CRC Data Bits 31:24
Base + \$01	CRCDH0	Read / Write	0xFF	CRC Data Bits 23:16
Base + \$02	CRCDL1	Read / Write	0xFF	CRC Data Bits 15:8
Base + \$03	CRCDL0	Read / Write	0xFF	CRC Data Bits 7:0
Base + \$04	CRCPH1	Read / Write	0x00	CRC Polynomial Bits 31:24
Base + \$05	CRCPH0	Read / Write	0x00	CRC Polynomial Bits 23:16
Base + \$06	CRCPL1	Read / Write	0x10	CRC Polynomial Bits 15:8
Base + \$07	CRCPL0	Read / Write	0x21	CRC Polynomial Bits 7:0
Base + \$08	CRCCTL	Read / Write	0x00	CRC Control Register

The above registers are explained in sections below.

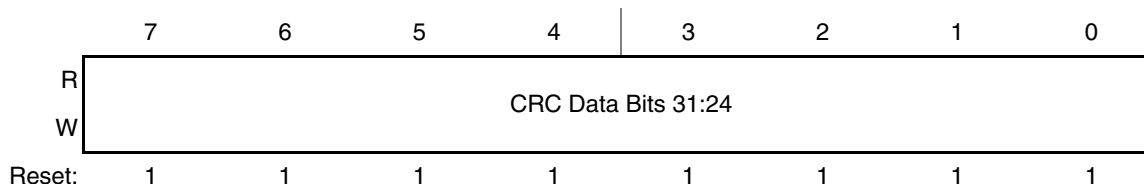
### 20.3.1 CRC Data Register (CRCDH1:CRCDH0:CRCDL1:CRCDL0)

This section describes the function of CRC data registers (CRCDH1:CRCDH0:CRCDL1:CRCDL0). The set of CRC data registers contains the value of seed, data, and checksum. When SEED bit in CRCCTL register is set, any write to the data registers will be regarded as seed for CRC module. When SEED bit is de-asserted, any write to the data registers will be regarded as data for CRC module for general CRC computation.

When programming the seed value in 16-bit CRC mode, the CRCDH1:CRCDH0 are not used and reads to these registers returns an indeterminate value. For 32-bit CRC, all registers are used.

When programming data values for CRC calculation, data must be provided in CRCDL0 register only. Writes to other bytes of data registers are ignored.

When final data written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in CRCDL1:CRCDL0. In 32-bit CRC mode, all registers are used. Reads to this register at any time returns the intermediate CRC value, provided CRC module is configured.



**Figure 20-3. CRC Data Register 3 (CRCDH1)**



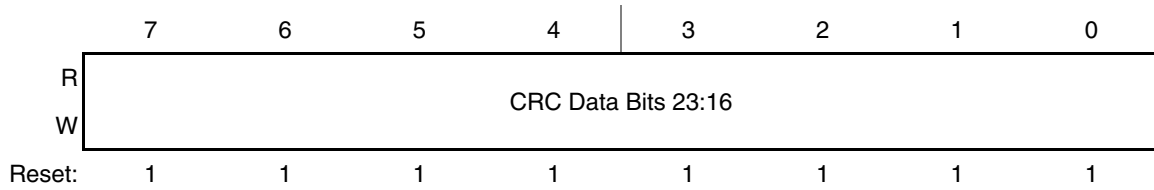


Figure 20-4. CRC Data Register 2 (CRCDH0)

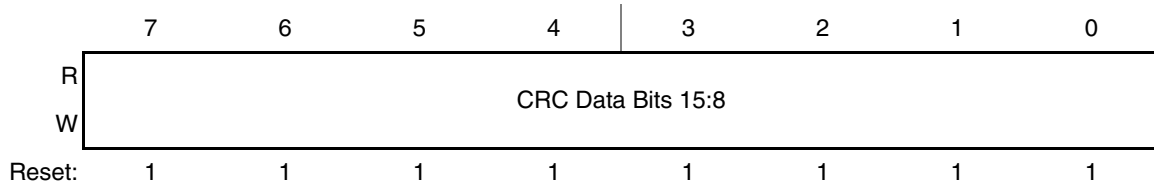


Figure 20-5. CRC Data Register 1 (CRCDL1)

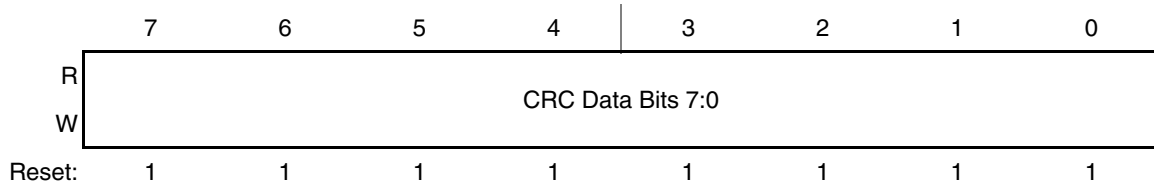


Figure 20-6. CRC Data Register 0 (CRCDL0)

### 20.3.2 CRC Polynomial Register (CRCPH1:CRCPH0:CRCPH1:CRCPH0)

This set of registers contains the value of polynomial for the CRC calculation. The registers of CRCPH1:CRCPH0 contain the upper 16-bits of CRC polynomial, which are only used in 32-bit CRC mode. Writes to CRCPH1:CRCPH0 are ignored in 16-bit CRC mode. The registers of CRCDL1:CRCDL0 contain the lower 16-bits of CRC polynomial, which are used in both 16- and 32-bit CRC modes.

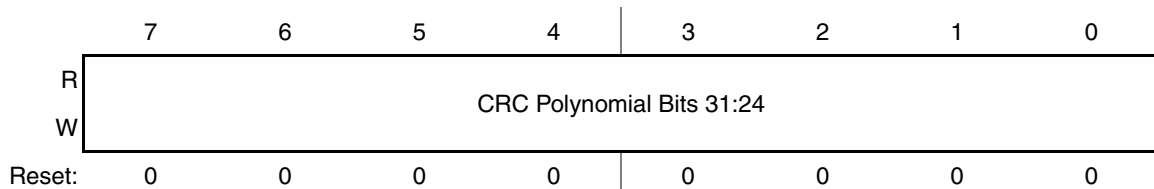


Figure 20-7. CRC Polynomial Register 3 (CRCPH1)

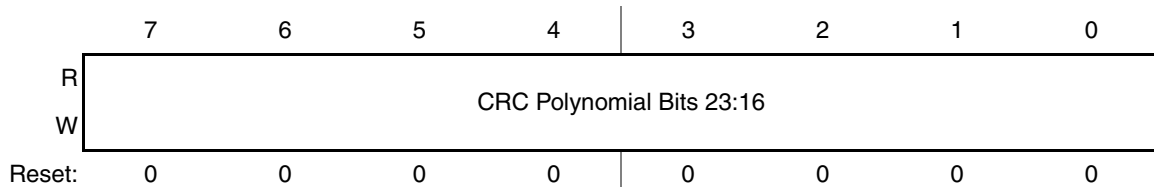


Figure 20-8. CRC Polynomial Register 2 (CRCPH0)

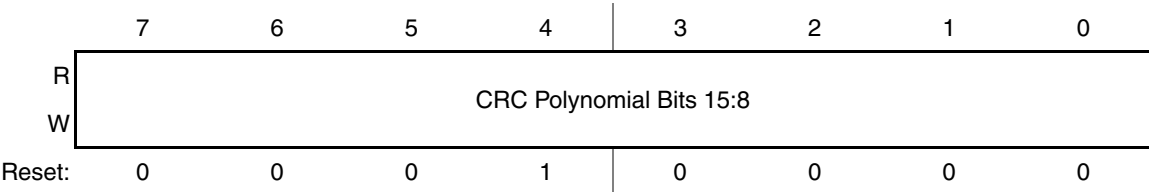


Figure 20-9. CRC Polynomial Register 1 (CRCPL1)

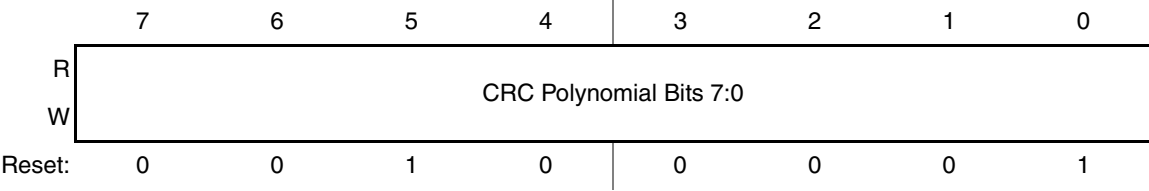


Figure 20-10. CRC Polynomial Register 0 (CRCPL0)

20.3.3 CRC Control Register (CRCCTL)

This register controls the configuration and working of the CRC module. Appropriate bits must be set before starting a new CRC calculation. A new CRC calculation is initialized by writing the seed into the CRC Data Register (after asserting SEED bit).

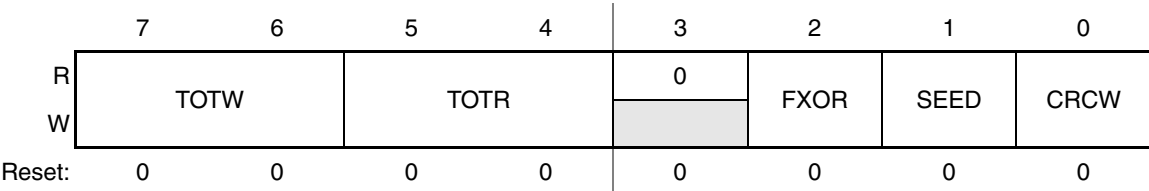


Figure 20-11. CRC Control Register (CRCCTL)

Table 20-2. CRCCTL Register Field Descriptions

Field	Description
7:6 TOTW	<b>Type of Transpose for Writes</b> — This bits define the transpose configuration of the write data to CRC data registers. Refer to <a href="#">Section 20.4.3, “Transpose Feature”</a> , for details of the transpose options available to the user.  00 – No Transpose. 01 - Bits in bytes are transposed, bytes are not transposed. 10, 11 - Invalid values. Should not be used.
5:4 TOTR	<b>Type of Transpose for Read</b> — This bits define the transpose configuration of the read value of the CRC data registers. Refer to <a href="#">Section 20.4.3, “Transpose Feature”</a> , for details of the transpose options available to the user.  00 – No Transposition. 01 - Bits in bytes are transposed, bytes are not transposed. 10, 11 - Invalid values. Should not be used.

Table 20-2. CRCCTL Register Field Descriptions

Field	Description
3	Reserved.
2 FXOR	<b>Complement Read of CRCDH/L registers</b> — Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables complement of read data on the fly.  0 – No XOR on reading. 1 – Invert or complement the read value of CRC data register
1 SEED	<b>Write to the CRC Data Registers is a SEED</b> — When asserted, writes to CRC data registers is considered a seed value else it is taken as data input over which CRC is to be computed.  0 – Writes to CRC data register are data values 1 – Writes to CRC data register are seed values
0 CRCW	<b>Width of CRC protocol</b>  0 16-bit CRC protocol. 1 32-bit CRC protocol.

## 20.4 Functional Description

### 20.4.1 CRC Initialization/Re-Initialization

To enable the CRC calculation, the user must program the SEED, POLYNOMIAL and necessary transpose and CRC result inversion parameters in their respective registers. Asserting the CRCCTL[SEED] bit, enables the programming of SEED value into the CRC data registers.

Re-asserting the SEED bit and programming a new or same seed value after a previously completed CRC calculation will re-initialize the CRC for a new CRC computation. All other parameters must be set before programming the seed value and subsequent data values.

### 20.4.2 CRC Calculations

#### 20.4.2.1 16-bit CRC

The following steps show the process to compute a 16-bit CRC

- Clear CRCW bit in CRCCTL register to enable 16-bit CRC mode
- Program the transpose and complement options in the CRCCTL as required for the CRC calculation. See [Section 20.4.3, “Transpose Feature”](#) and [Section 20.4.4, “CRC Result Complement”](#) for details.
- Write 16-bit polynomial to CRCPL1: CRCPL0. CRCPH1: CRCPH0 are not usable in 16-bit CRC mode.
- Set SEED bit in CRCCTL register to program the seed value.
- Write 16-bit seed to CRCDL1: CRCDL0. CRCDH1: CRCDH0 are not used.
- Clear SEED bit in CRCCTL register to start writing data values.

- Write data values into CRCDL0. CRC is computed on every data value write and the intermediate CRC result is stored back into CRCDL1:CRCDL0.
- When all values have been written, the final CRC result can be read out from CRCDL1:CRCDL0.
- Transpose and complement are performed on the fly while read or writing values. See [Section 20.4.3, “Transpose Feature”](#) and [Section 20.4.4, “CRC Result Complement”](#) for more details.

### 20.4.2.2 32-bit CRC

- Set CRCW bit in CRCCTL register to enable 32-bit CRC mode
- Program the transpose and complement options in the CRCCTL as required for the CRC calculation. See [Section 20.4.3, “Transpose Feature”](#) and [Section 20.4.4, “CRC Result Complement”](#) for details.
- Write 32-bit polynomial to CRCPH1:CRCPH0:CRCP1:CRCP0.
- Set SEED bit in CRCCTL register to program the seed value.
- Write 32-bit seed to CRCDH1:CRCDH0:CRCDL1:CRCDL0.
- Clear SEED bit in CRCCTL register to start writing data values.
- Write data values into CRCDL0. CRC is computed on every data value write and the intermediate CRC result is stored back into CRCDH1:CRCDH0:CRCDL1:CRCDL0.
- When all values have been written, the final CRC result can be read out from CRCDH1:CRCDH0:CRCDL1:CRCDL0.
- Transpose and complement are performed on the fly while read or writing values. See [Section 20.4.3, “Transpose Feature”](#) and [Section 20.4.4, “CRC Result Complement”](#) for more details.

## 20.4.3 Transpose Feature

Some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose separately as desired by the CRC standard. By default transpose is not enabled. The data is transposed on the fly while being read or written.

Some protocols use little endian format for the data stream to calculate CRC. In such a case, this feature comes quite handy in doing such flipping of bits. The following sections detail the features of different types of the transposition.

### 20.4.3.1 Types of Transpose

Several types of transpose functions are provided to user software in order to flip the bits and/or bytes (for both input data and CRC result, individually) according to the CRC calculation being used.

The following types of transpose functions are available for writing the data to as well as reading from the CRC data register.

1. TOTW/TOTR = 00  
No Transpose is done.

2. TOTW/TOTR = 01

Bits are in byte transposed but bytes are not transposed.

reg[7:0] becomes reg[0:7]

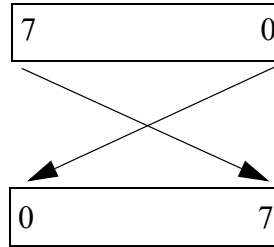


Figure 20-15. Transpose Type 01

3. TOTW/TOTR = 10, 11

Invalid values. Should not be used.

**NOTE**

Byte wise transpose cannot be performed with 8-bit accesses and hence must be done in software by the CPU.

## 20.4.4 CRC Result Complement

The CRC result complement function outputs the complement of the checksum value in CRC data registers every time the CRC data register is read out. When FXOR bit in CRCCTL register is set, the checksum is complemented. Otherwise, the raw checksum is accessed.



## Chapter 21

# Position Counter (S08PCNTV1)

### 21.1 Introduction

Position counter (PCNT) tracks the position (also called as state) of the rotatory device attached outside via sensors connected to the rotatory device.

The main parts of the PCNT are register space, forward counter (FCounter) and reverse counters (RCounter), state machine, input filter and waveform generation block.

#### NOTE

When PCNT is in the STOP3 mode, do not drop the  $V_{DD}$  under 1.8 V, or else the LVD function must be enabled in advance.

#### NOTE

When PCNT is in the STOP3 mode, 512 Hz clock is not available when LVD is disabled.

Ignore any reference to STOP4 mode in this chapter, because this device does not support it.

#### 21.1.1 PCNT Internal Connection

The PCNT0, PCNT1 and PCNT2 can be fed from PRACMP0, PRACMP1 and PRACMP2 outputs correspondingly. See [Section 2.3.11, “PCNT\[2:0\],”](#) for more information.

#### 21.1.2 PCNT Clock Gating

The bus clock to the PCNT module can be gated on and off using the PCNT bit in SCGC5. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the PCNT bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

#### NOTE

Please follow the following guide in the application:

- 1) Before enabling the Pcounter, please configure the filter clk select in the SIMIPS3 register and the PCNT sensor input select in the SIMIPS1 register.
- 2) PCNT filter's clock can be either BUSCLK, 32Khz or 512Hz clock based upon its mode of operation. User can have 32khz and 512Hz clock as a filter clock in all the PCNT modes. BUSCLK option should only be selected only in PWM/Atomic Counter mode.

3) In case PCNT is running in the PWM mode, please make all the channel sel bits to zero. Otherwise PCNT will continuously enabling/disabling PRACMP0-2, which will impact the PRACMP's operation if it is being used for some other purpose.



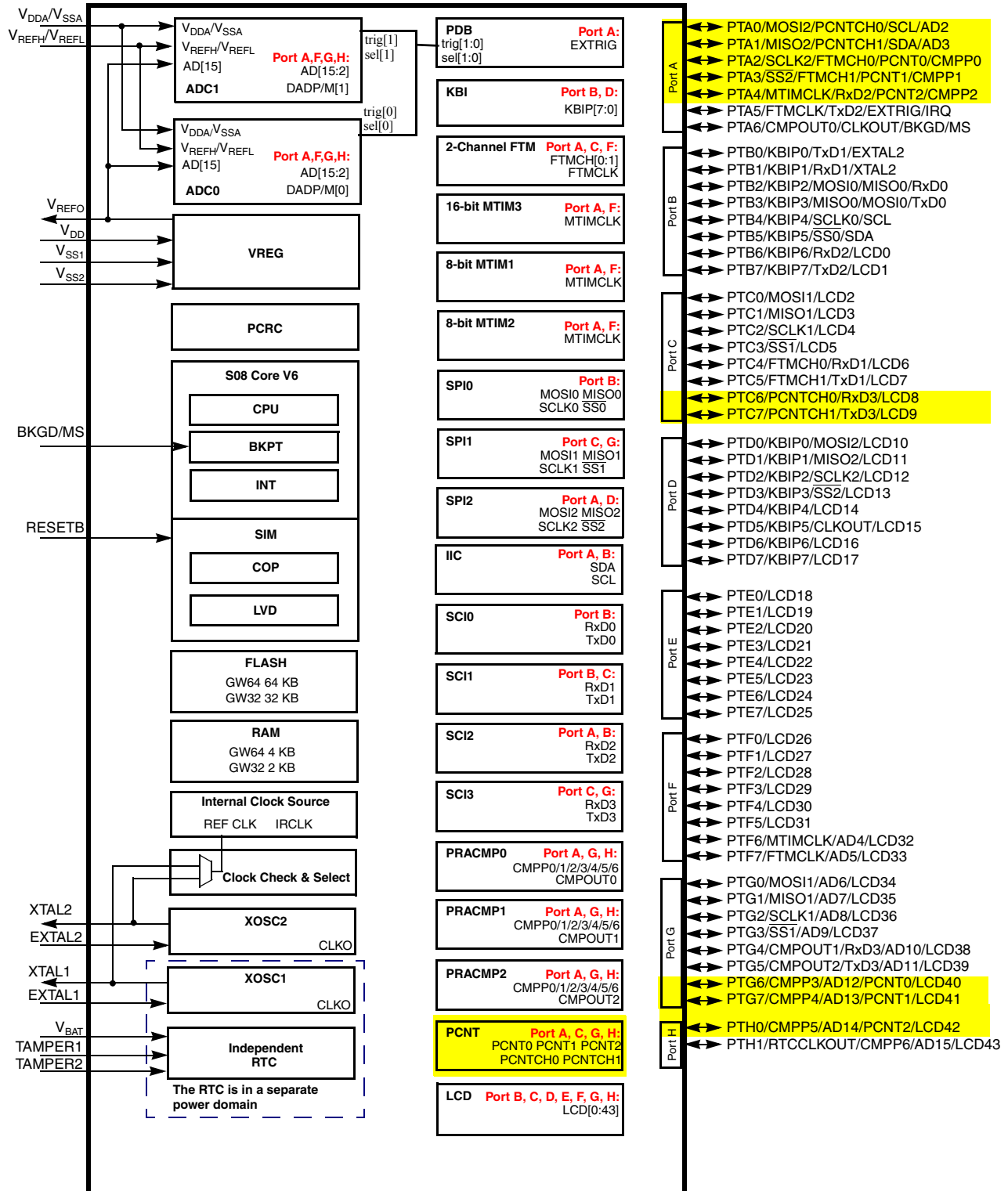


Figure 21-1. MC9S08GW64 Series Block Diagram Highlighting PCNT Module and Pins

### 21.1.3 Features

The PCounter includes these distinctive features:

- Support interfacing to one-, two- or three-pin rotatory sensor.
- Support 180 degree, gray and binary decoding mode. Two signal gray mode is also called as quadrature mode.
- Able to filter sensor signals with programmable filter width.
- Able to detect and generate interrupts on an invalid sequence.
- Generate interrupts on counter overflow.
- Generate required signaling to activate sensors .
- Generate asynchronous interrupt to wake MCU from low power modes.
- Able to function independently in stop mode.
- Modulus registers to interrupt CPU on specific count or full count.
- Able to function as limited capability PWM (i.e. edge- or centre-aligned PWM).
- Internal counter can function as a atomic counter which increments on each IPS (IP Slave Bus Interface) writes and generates an interrupt on counter overflow.

### 21.1.4 Modes of Operation

PCNT or position counter is a low power pulse sequence counter. Once configured, it is capable of working independent of the CPU in stop3 or stop4 modes. The PCounter accumulates the valid pulses input to it. These inputs are filtered from noise via internal digital filters running on a slow frequency filter clock. The internal state machine will increment the internal counters when it detects the required valid sequence on the filtered inputs. The PCounter interfaces to the CPU via an IPS (IP Slave Bus) interface i.e. the CPU register programming interface running on the bus clock. The register space of this module shadows the internal counters (running on filter clock) for CPU to read apart from maintaining other status and controlling registers.

The PCounter interfaces to various rotatory sensors that provide it with necessary pulse sequence to determine the flow direction and quantity of the flow. PCounter supports one-pin to three-pin rotatory sensors which can output a digital or analog pulse waveform. These pulse waveform inputs are filtered from noise via internal digital filters running on filter clock. The sensors are external to the MCU and require on-chip comparators to convert the analog waveforms into digital. This module can interface to the sensors which can output any of the following pulse sequence or waveform (if analog):

- Two- or three-bit binary or gray coded
- Two-bit 180 degree coded
- Two-bit quadrature (90 degree) coded
- One-bit (normal waveform)

PCounter is capable of generating the necessary signaling to enable or disable the on-chip comparators and external sensors to save on power and enable sampling when the inputs are stable. This logic works off the filter clock and hence can run standalone without CPU intervention once configured. PCounter supports edge- or center-aligned waveforms for sampling and controlling the sensors and comparators similar to the

typical PWM. It can even function as a limited capability PWM when its sensor inputs are not used and disabled.

For more details on the operation of the PCounter, refer to [Section 21.4, “Functional Description”](#).

## **21.1.5 Low Power Modes**

### **21.1.5.1 Run Mode**

In run mode, PCounter is fully operational.

### **21.1.5.2 Wait Mode**

In wait mode, PCounter is fully operational.

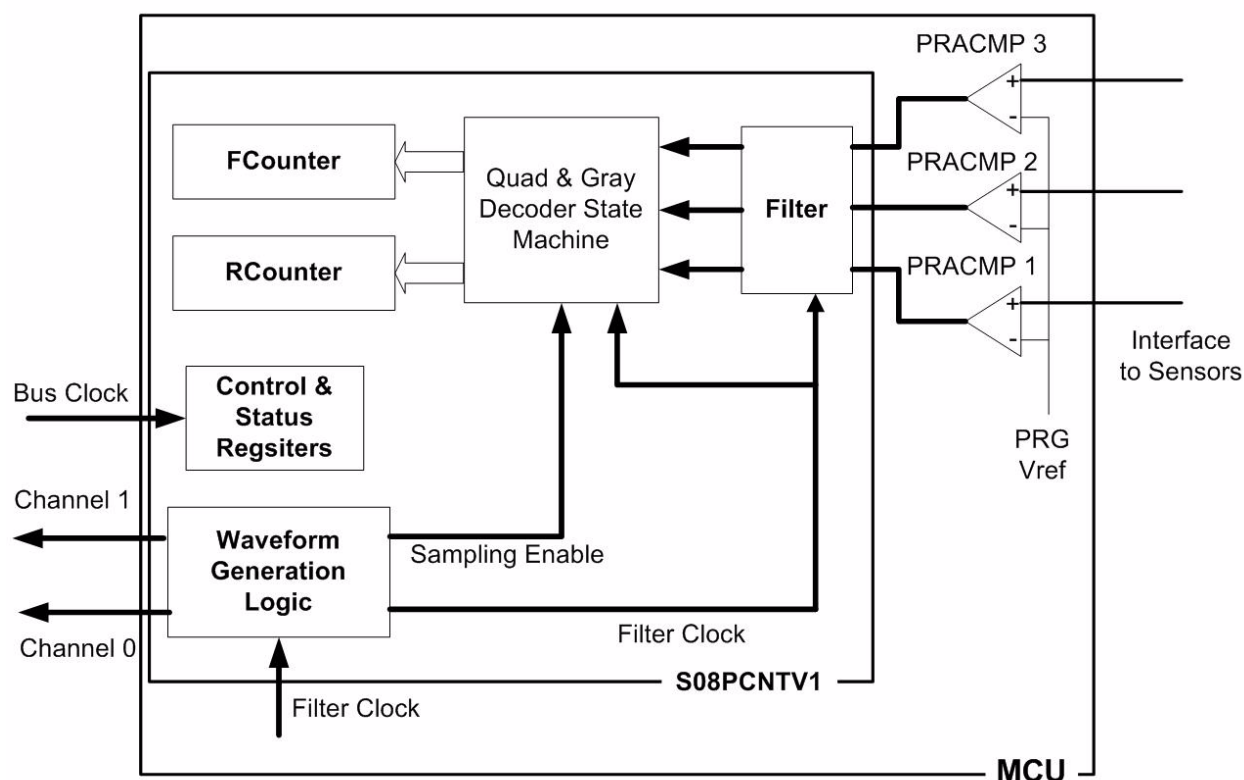
### **21.1.5.3 Stop Mode**

The PCounter has its state machine and filters running on filter\_clk. It can count in all MCU modes in which the filter\_clk and the power supply of this module is available. Asynchronous interrupt can be generated and used to wake the MCU from stop modes by the following events. These interrupts can be individually disabled.

- FCounter overflows if FCOVFIE of the PCNT\_CTRL register is set.
- RCounter overflows if RCOVFIE of the PCNT\_CTRL register is set.
- The state on Pcounter state input is invalid if SINVIE of PCNT\_CTRL is set.

## **21.1.6 Block Diagram**

[Figure 21-2](#) is the top level block diagram of the PCounter.



**Figure 21-2. Block Diagram of PCounter**

PCounter is divided into four major sub-blocks based upon the functionality.

- **Register Space Block:** This sub-block contains IPS decode logic and all registers mentioned in [Section 21.3, “Register Definition”](#).
- **Input Filters:** The state input signals are filtered from noise and unwanted toggling to ensure a minimum width of pulses on state input.
- **State Machine Block:** This sub-block decodes the state input signals and accordingly updates counters (FCounter and RCounter).
- **Waveform Generation Block:** This sub-block generates edge- or centre-aligned PWM signal on its Channel1 and Channel2 outputs, which can be used to generate necessary signal timing to activate the sensor and subsequent sampling of sensor inputs.

## 21.2 External Signal Description

[Table 21-1](#) shows the user-accessible signals of PCounter available at the MCU pins.

**Table 21-1. External Signals of PCounter**

No.	Signal Name	Type	Direction	Functional Description
1	PCNT[2:0]	—	In	This input pin reflects the state of the rotatory device connect outside the chip. Sensor Input.
2	PCNTCH0	—	Out	PCounter PWM Channel1 output.
3	PCNTCH1	—	Out	PCounter PWM Channel2 output.

## 21.3 Register Definition

This section provides a detailed descriptions of all PCounter registers.

### 21.3.1 PCounter Status Register (PCNT\_STATUS)

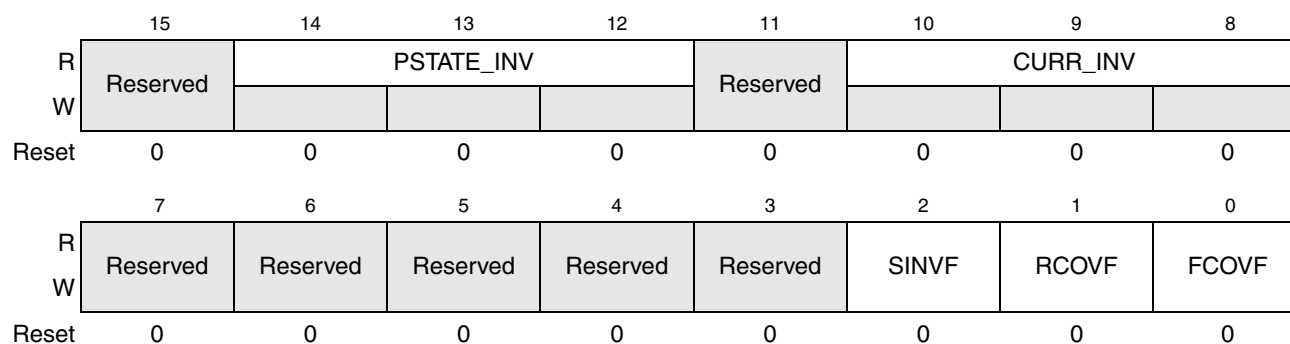


Figure 21-3. PCounter Status (PCNT\_STATUS) Register

Table 21-2. PCNT\_STATUS Field Descriptions

Field	Description
15	Reserved.
14:12 PSTATE_INV V	Sensor input pin state when a state invalid event occurs. PSTATE_INV is used in debugging to indicate the value on sensor inputs when a state invalid event occurs. If a multiple state invalid event occurs, PSTATE_INV indicates the state of the last state invalid event. This PSTATE_INV is valid only when SINVF is set. PSTATE_INV is read only.
11	Reserved.
10:8 CURR_INV	Current state of state machine when a state invalid event occurs. This CURR_INV is used in debugging to indicate the current state of the internal state machine when a state invalid event occurs. If a multiple state invalid event occurs, the CURR_INV indicates the current state on the latest state invalid event. This CURR_INV is valid only when SINVF is set. CURR_INV is read only.
7:3	Reserved.
2 SINVF	State invalid interrupt flag. This bit indicates whether the sensor inputs, with respect to the current state, is valid or not. Refer to <a href="#">Section 21.4, “Functional Description,”</a> for more details. Write 1 to clear this bit. 1 Invalid state detected at PCNT[2:0]. 0 No invalid state detected at PCNT[2:0].

**Table 21-2. PCNT\_STATUS Field Descriptions (continued)**

Field	Description
1 RCOVF	<p>RCounter overflow flag.</p> <p>When RCounter reaches the reverse counter modulus register's value (if this feature is enabled) or 16'FFFF (if this feature is disabled), it starts again from 16'h0000 and sets RCOVF bit. Write 1 to clear this bit.</p> <p>1 RCounter overflowed and started from 0000.</p> <p>0 No overflow occurred.</p>
0 FCOVF	<p>FCounter overflow flag.</p> <p>When FCounter reaches the forward counter modulus register's value (if modulus feature is enabled) or 16'FFFF (if modulus feature is disabled), it starts again from 16'h0000 and sets FCOVF bit. Write 1 to clear this bit.</p> <p>1 FCounter overflowed and started from 0000.</p> <p>0 No overflow occurred.</p>

## 21.3.2 PCounter Control Register (PCNT\_CTRL)

	15	14	13	12	11	10	9	8
R								
W	SINVIE	RCOVFIE	FCOVFIE	MODE		CHANNEL_SEL		
Reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R								
W	PCNT_EN	DIR	POL	CPWMS	FILTER VALUE			
Reset	0	0	0	0	0	0	0	0

**Figure 21-4. PCNT\_CTRL Register**

**Table 21-3. PCNT\_CTRL Field Descriptions**

Field	Description
15 SINVIE	<p>State invalid interrupt enable bit.</p> <p>When an invalid state is detected on PCounter's input, it sets the SINVIE bit in the PCNT_STATUS. If SINVIE bit is set, the invalid state event will generate an interrupt.</p> <p>1 State invalid event generates an interrupt.</p> <p>0 State invalid event does not generate an interrupt.</p>
14 RCOVFIE	<p>RCounter overflow interrupt enable bit.</p> <p>When an RCounter overflows, it sets the RCOVF bit in the PCNT_STATUS. If RCOVFIE bit is set, RCounter overflow event will generate an interrupt.</p> <p>1 RCounter overflow event generates an interrupt.</p> <p>0 RCounter overflow event does not generate an interrupt.</p>
13 FCOVFIE	<p>FCounter overflow interrupt enable bit.</p> <p>When an FCounter overflows, it sets the FCOVF bit in PCNT_STATUS. If FCOVFIE bit is set, FCounter overflow event will generate an interrupt.</p> <p>1 FCounter overflow event generates an interrupt.</p> <p>0 FCounter overflow event does not generate an interrupt.</p>

**Table 21-3. PCNT\_CTRL Field Descriptions (continued)**

Field	Description
12:11 MODE	This bit selects the gray mode, binary mode, 180 degree dual sensor mode or PWM mode of operation. 00 PCounter operates in binary mode. 01 PCounter operates in gray mode. 10 PCounter operates in 180 degree dual sensor mode. 11 PCounter operates only as PWM module ( Sensor input is not sampled ) User cannot update MODE bit once PCounter is enabled.
10:8 CHANNEL_SEL	Depending upon the pins of the rotatory disc to be interfaced outside, user enables or disables the individual input. For example, if user wants to operate PCounter in quadrature gray mode in which only two inputs are required, users must enable corresponding two bits of CHANNEL_SEL to be connected to the rotatory sensor. User cannot update CHANNEL_SEL bits once PCounter is enabled. Out of reset, as CHANNEL_SEL reset value is 3'b000.
7 PCNT_EN	Enable/Disable bit. This bit enables or disables PCounter. Once PCounter is enabled, all the IPS registers with write access are locked (except for the PCNT_STATUS and interrupt enabler bits). User needs to disable the PCounter to get IPS registers unlocked and change their value. 1 PCounter is enabled. 0 PCounter is disabled.
6 DIR	Direction bit. This bit indicates to the PCounter about the direction of the flow. If this bit is set, only forward flow direction is valid. If this bit is reset, both the forward and reverse flow direction are valid. User cannot update DIR bit once PCounter is enabled
5 POL	Polarity bit. This bit sets the polarity of both PWM channels of PCounter. 1 High-true pulses (clear output on compare-up) on Ch0 and Ch1. 0 Low-true pulses (set output on compare-up) on Ch0 and Ch1. User cannot update POL bit once PCounter is enabled.
4 CPWMS	CPWM selection. This bit selects whether the PCounter internal generates a centre- or edge-aligned PWM. 1 Ch0 and Ch1 operate as a center-aligned PWM. 0 Ch0 and Ch1 operate as a edge-aligned PWM. User cannot update CPWMS bit once PCounter is enabled.
3:0 FILTER VALUE	These bits indicate the number of the filter clocks the signal PCNT[2:0] must remain stable for being detected. These bits are used by the filtering block. 0 Filtering operation disabled. 1 to 15 Number of filter clocks to be counted when any change is detected in the input signal (either 0 to 1 or 1 to 0). Take caution when making the filter duration equal to 0 or 1 as any glitch on the PCNT[2:0] pins can cause a state change or set the invalid mode flag. User cannot update FILTER VALUE once PCounter is enabled.

CHANNEL\_SEL and MODE decide the PCounter operating mode. [Table 21-4](#) shows the operating modes of PCounter. See [Section 21.4, “Functional Description,”](#) for more details.



**Table 21-4. PCounter Operating Modes**

PCNT_EN	MODE	CHANNEL_SEL	OPERATING MODE
0	XX	XXX	PCounter Disabled
1	00	000	Invalid Configuration
1	00	001,010,100	Single Direct Counter Mode
1	00	011,101,110	Two Signals Binary Mode
1	00	111	Three Signals Binary Mode
1	01	000	Invalid Configuration
1	01	001,010,100	Single Direct Counter Mode
1	01	011,101,110	Two Signals Gray Mode
1	01	111	Three Signals Gray Mode
1	10	000	Invalid Configuration
1	10	001,010,100	Invalid Configuration
1	10	011,101,110	180 degree Dual Sensor Mode
1	10	111	Invalid Configuration
1	11	XXX	Atomic counter or PWM Mode (edge- or center-aligned)

PCounter is disabled in the invalid configuration.

### 21.3.3 PCounter Forward Counter Modulus Register (PCNT\_FCMOD)

The PCNT\_FCMOD modulo register contains the modulo value for the FCounter. After the FCounter reaches the modulo value, it resumes counting from 0x0000, and the overflow flag FCOVF of PCNT\_STATUS becomes set.

This modulo feature can be disabled by programming 0x0000 in PCNT\_FCMOD register. In this case, FCounter counts from 0x0000 to 0xFFFF and then rolls over. FCOVF of PCNT\_STATUS gets set when the FCounter is starting again from 0x0000.

PCNT\_FCMOD is updated by an IPS write only when PCounter is disabled (i.e. PCNT\_EN = 0). Once enabled, all the IPS writes to this register are ignored. IPS read on this registers has no such restrictions. It can be read at any time.

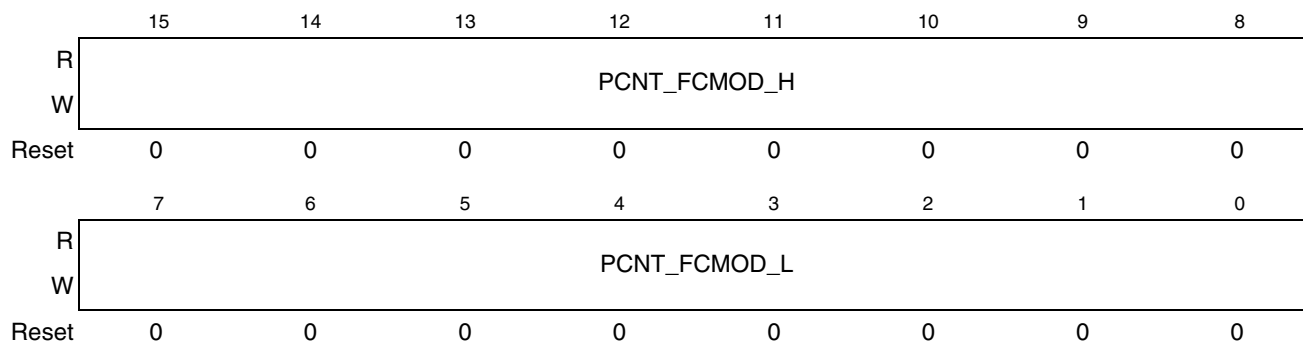


Figure 21-5. PCounter Forward Counter Modulus (PCNT\_FCMOD) Register

### 21.3.4 PCounter Forward Counter Register (PCNT\_FCNTNTR)

PCNT\_FCNTNTR gives the forward count value. This register is also called as the IPS\_FCounter register. IPS\_FCounter is just a mirror of FCounter which is actually inside of the state machine block.

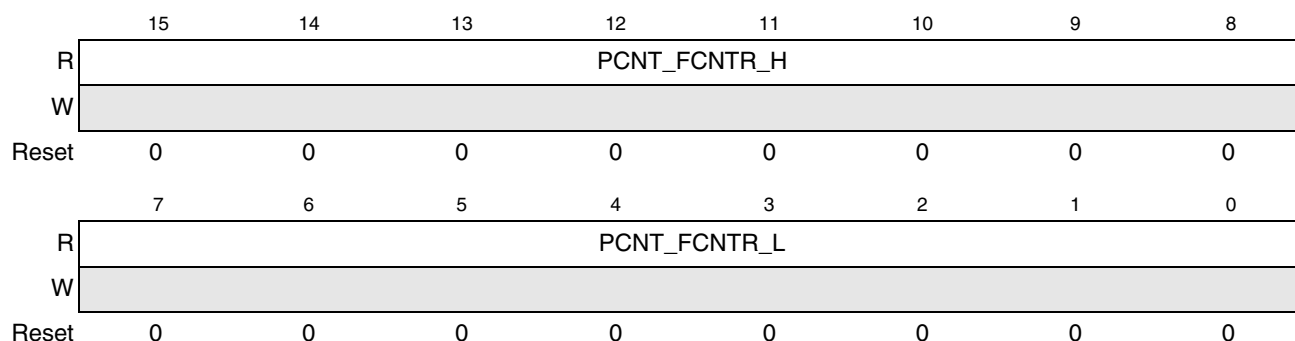


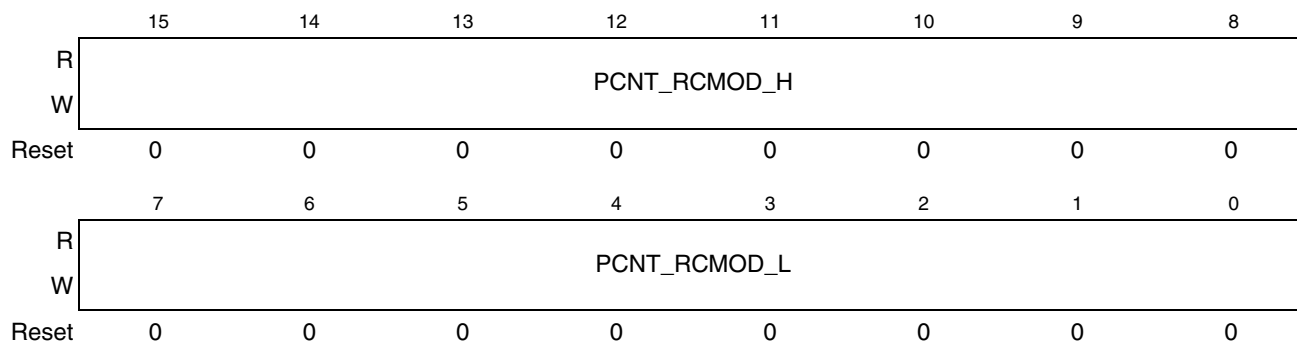
Figure 21-6. PCNT\_FCNTNTR

### 21.3.5 PCounter Reverse Counter Modulus Register (PCNT\_RCMOD)

The PCNT\_RCMOD modulo register contains the modulo value for the RCounter. After RCounter reaches the modulo value, it resumes counting from 0x0000, and the overflow flag RCOVF of PCNT\_STATUS becomes set.

This modulo feature can be disabled by programming 0x0000 in PCNT\_RCMOD register. In this case, RCounter counts from 0x0000 to 0xFFFF and then rolls over. RCOVF of PCNT\_STATUS gets set when the FCounter is starting again from 0x0000.

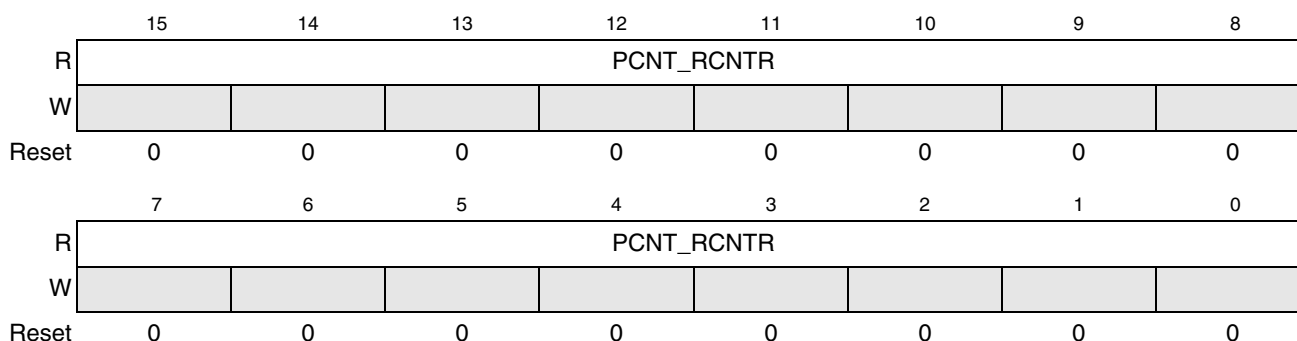
PCNT\_RCMOD can be updated by an IPS write only when PCounter is disabled (i.e. PCNT\_EN = 0). Once enabled, any IPS write to this register is ignored. IPS read on this registers has no such restrictions. it can be read at any time.



**Figure 21-7. PCounter Reverse Counter Modulus (PCNT\_RCMOD) Register**

### 21.3.6 PCounter Reverse Counter Register (PCNT\_RCNTNTR)

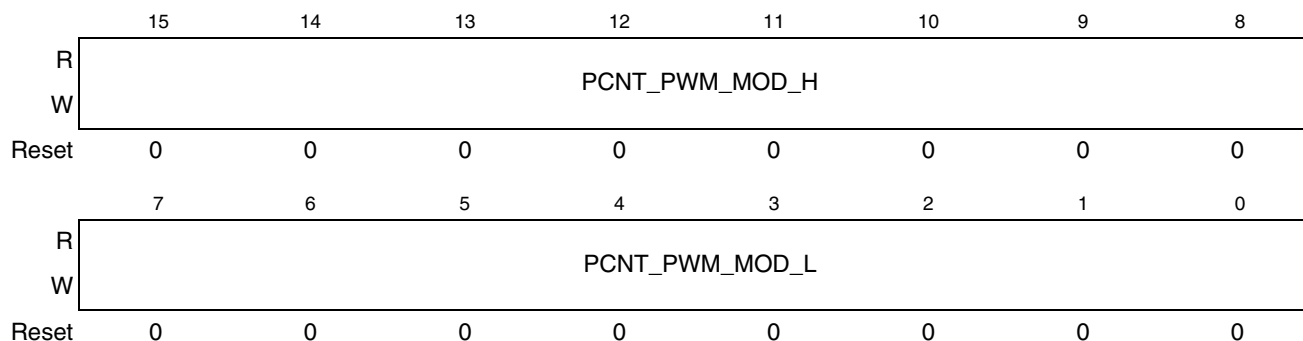
PCNT\_RCNTNTR gives the reverse count value. This register is also called the IPS\_RCounter register. IPS\_RCounter is just a mirror of RCounter which is actually inside of the state machine block.



**Figure 21-8. PCounter Reverse Counter (PCNT\_RCNTNTR) Register**

### 21.3.7 PCounter PWM Modulus Register (PCNT\_PWM\_MOD)

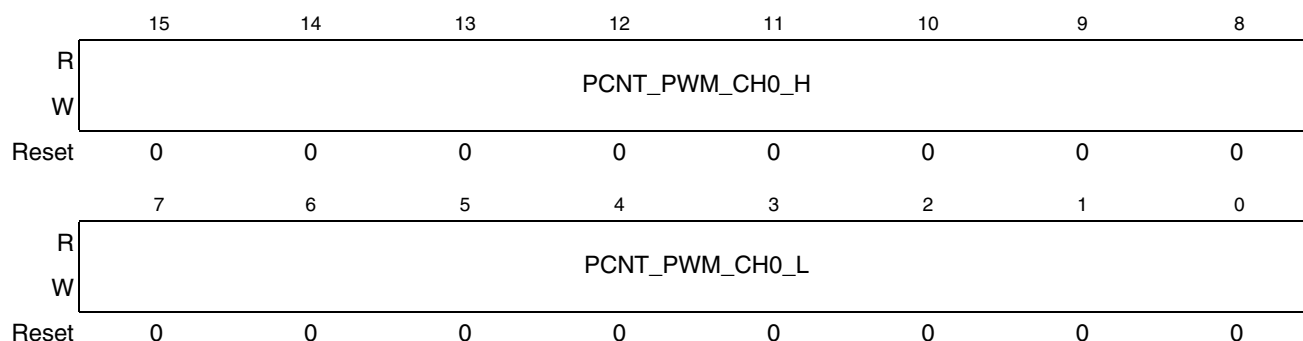
A 16 bit internal PWM counter is present inside of the waveform generation block that generates the required waveform (edge- or centre-aligned) on both of the channel outputs. The PWM modulus register contains the modulo value for the internal 16 bit internal PWM counter. If PWM is programmed to generate edge-aligned PWM, the PWM counter counts up to the modulo value and resumes counting from 0x0000 at the next clock edge. If PWM is programmed to generate centre-aligned PWM, PWM counter counts up to modulo value and then starts counting down to 0x0000 at the next clock edge.



**Figure 21-9. PCounter PWM Modulus (PCNT\_PWM\_MOD) Register**

### 21.3.8 PCounter PWM Channel 0 Value Register (PCNT\_PWM\_CH0\_VAL)

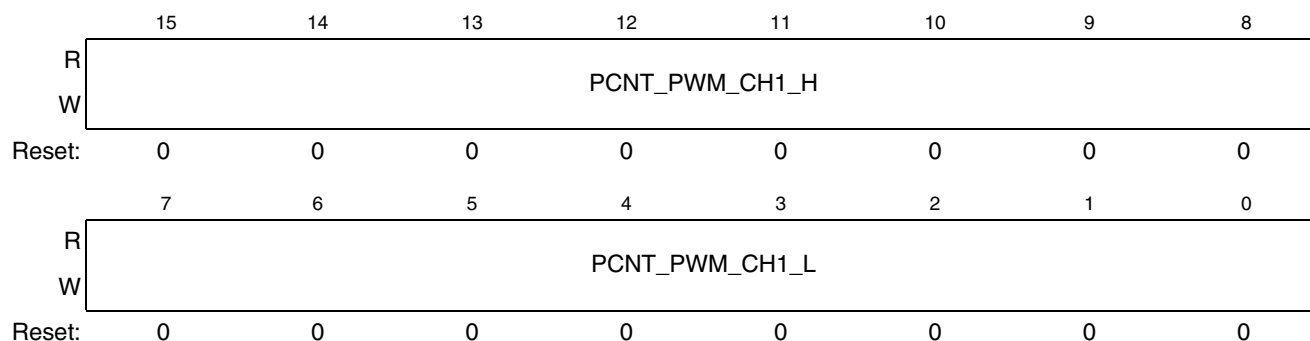
The PWM Channel 0 value register contains the value which is used to compare with internal PWM counter to generate the required waveform on Channel 0.



**Figure 21-10. PCounter PWM Channel 0 Value (PCNT\_PWM\_CH0\_VAL) Register**

### 21.3.9 PCounter PWM Channel 1 Value Register (PCNT\_PWM\_CH1\_VAL)

The PWM Channel 1 value register contains the value which is compared with internal PWM counter to generate the required waveform on Channel 1.



**Figure 21-11. PCounter PWM Channel 1 Value (PCNT\_PWM\_CH1\_VAL) Register**

## 21.3.10 PCounter State Register(PCNT\_STATE)

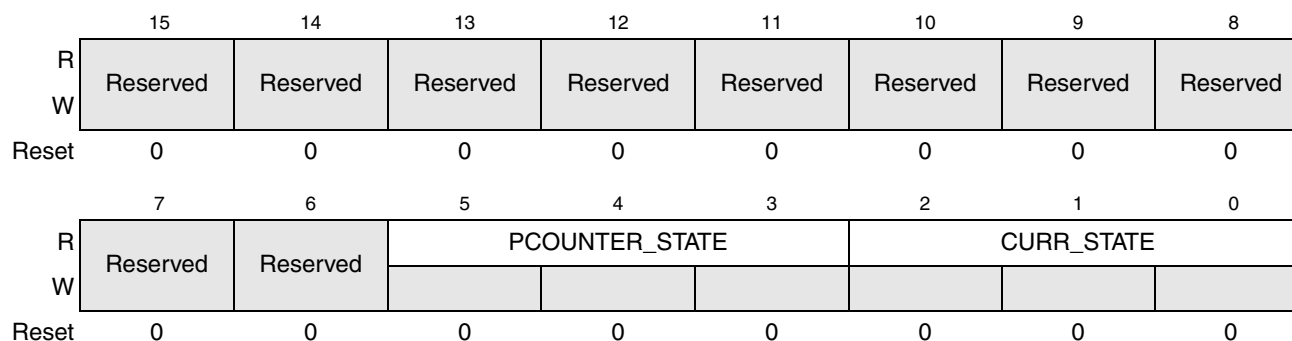


Figure 21-12. PCNT\_STATE

Table 21-5. PCNT\_STATE Field Descriptions

Field	Descriptions
15:6	Reserved
5:3 PCOUNTER_STATE	This is used for debugging purpose. These bits reflect value on sensor inputs which are coming into PCounter.
2:0 CURR_STATE	This is used for debugging purpose. These bits reflect the current state of the state machine.

## 21.4 Functional Description

### 21.4.1 Modes of Operation

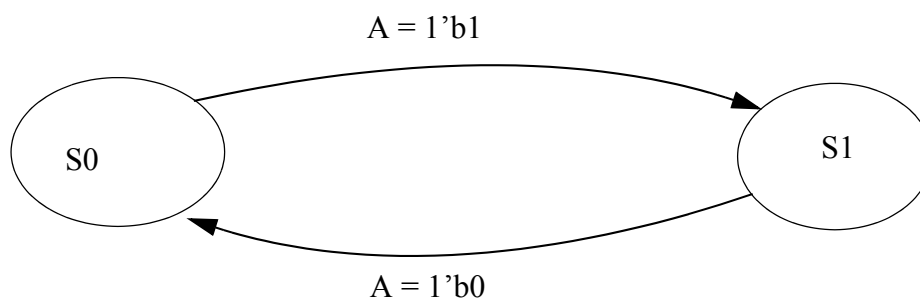
#### 21.4.1.1 Single Direct Counter Mode

PCounter must be programmed in this mode if it is connected to a rotary disc having a single sensor. In this mode, high and low reflect the state of the device connected. PCounter's FCounter is updated based upon the transition made by the input signal from 0 to 1 as well as from 1 to 0. Because we can not determine the direction in which rotary disc is moving, the FCounter is updated on each transition. RCounter is never updated.



Figure 21-13. Sensor Inputs for Rotary Disc Moving Forward or Backward

PCounter's internal state machine transition in this mode is shown in the [Figure 21-14](#).



**Figure 21-14. PCNT Internal State Machine Transition**

Table 21-6 shows the PCounter's internal state machine behaviour on various present states and sensor inputs.

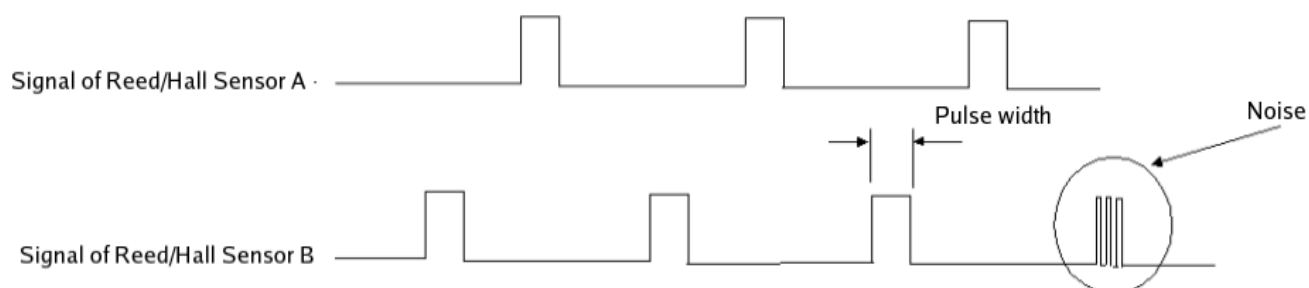
**Table 21-6. PCounter Internal State Machine Behavior**

Present State	Sensor Input	Next State	FCounter Update Pulse	RCounter Update Pulse	Invalid State Event
S0	1'b0	S0	No	No	No
S0	1'b1	S1	Yes	No	No
S1	1'b0	S0	Yes	No	No
S1	1'b1	S1	No	No	No

### 21.4.1.2 180 Degree Dual Sensor Mode

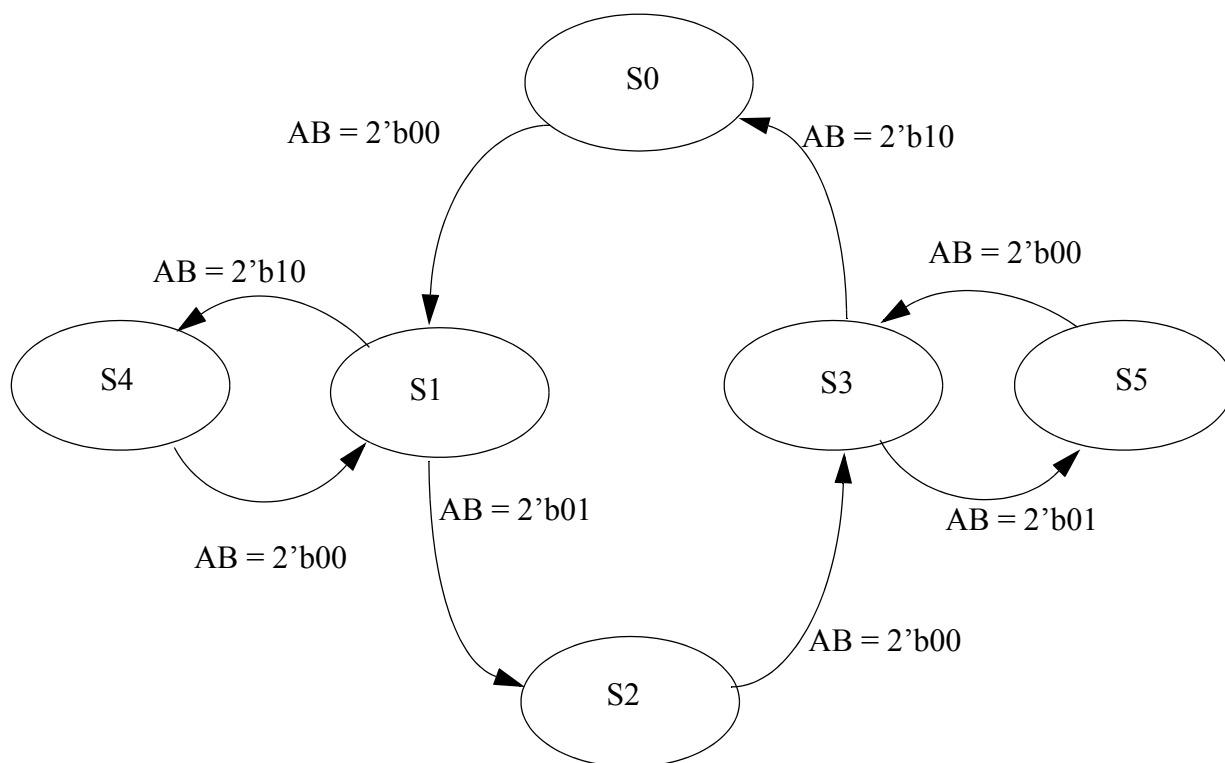
PCounter must be programmed in this mode if it connects to two sensors placed 180 degree apart.

Figure 21-15 shows the waveform which gets reflected on sensor inputs if disc is moving.



**Figure 21-15. Sensor inputs in 180 Degree Dual Sensor Mode**

Figure 21-16 shows the PCounter's internal state machine transition diagram in 180 degree dual sensor mode.



**Figure 21-16. State Transition Diagram in 180 Degree Dual Sensor Mode**

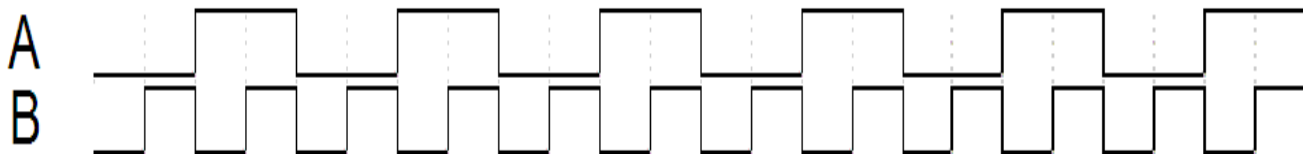
PCounter searches for a pattern 00-01-00-10 on the sensors inputs in this mode. Once the pattern is found, FCounter is increased by 1. Because the value “00” occurs on either side of the “01”, the direction can not be detected in this mode, and PCounter never increases the RCounter.

The glitches on the sensor inputs may cross the internal filters (As shown in [Figure 21-15](#), some glitches are coming on the sensor input B). These can be taken care by the internal state machine if the noise is coming only on one sensor input in a particular cycle. If it is coming on both sensor inputs, it may lead to counting errors, which is very rare.

### 21.4.1.3 Two-Signal Binary Mode

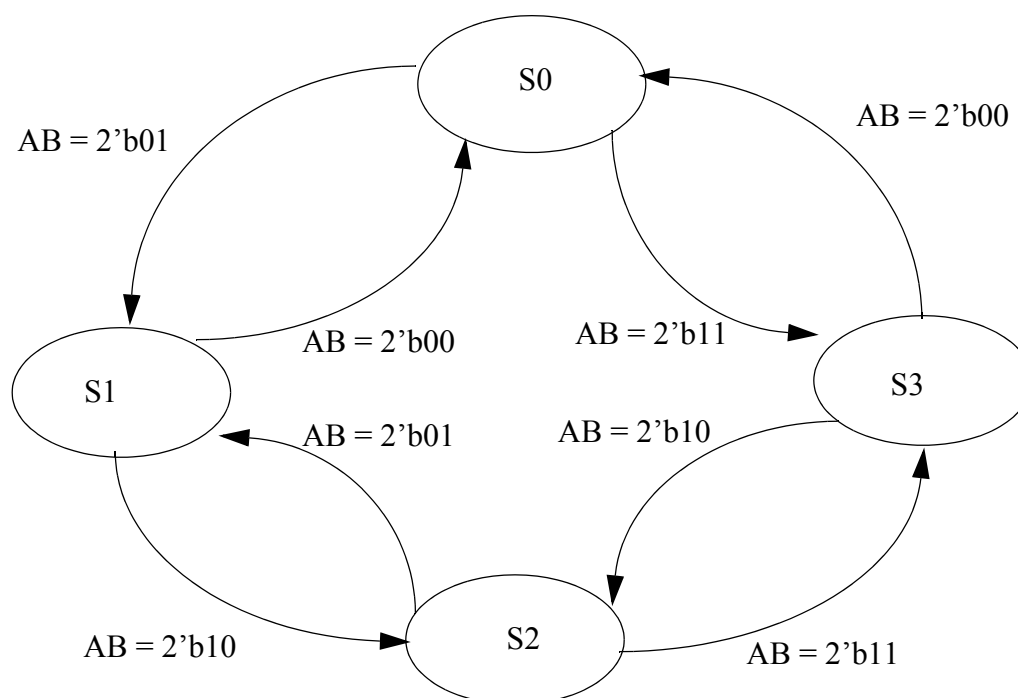
PCounter is programmed in this mode if it is connected to a rotary disc having two sensors and is binary encoded. In this two-signal binary mode, both signals cumulatively reflect the state of the device.

Contacts on the disc reflect the position (state) of the rotary disc. [Figure 21-17](#) shows the waveform of both contacts (A and B) when the disc is moving in the forward direction.



**Figure 21-17. Sensor Inputs for Rotary Disc Moving Forward**

PCounter gets the rotary disc's state by monitoring both contacts. PCounter keeps the state of the internal state machine the same as the rotary disc's position. When the rotary disc changes its state in rotating, internal state machine also makes transition to the same state. Figure 21-18 shows the PCounter's internal state machine transition diagram for two-signal binary mode.



**Figure 21-18. State Diagram for Two-Signal Binary Mode**

In the above state diagram, S0, S1, S2 and S3 are the states of PCounter's internal state machine for two contacts binary encoded disc. These states correspond to rotary disc's states or sensor inputs one to one. When the disc changes its state in rotating, PCounter's state machine also makes the same transition.

Depending on the present rotary disc's position (state), some states are valid while others are invalid. For example, suppose rotary disc's state is S1. If the disc moves forward, it will be in S2 state. If the disc moves backward, it will be in S0 state. So the valid states are S0 or S2 if the rotary disc is in S1 state. Disc can not move directly to S3 i.e.  $S1 \rightarrow S3$ . It has to follow  $S1 \rightarrow S2 \rightarrow S3$  or  $S1 \rightarrow S0 \rightarrow S3$  sequence. So the S3 is an invalid state. If S3 comes on the sensor inputs, it might be due to external unwanted factors.

In the above example, if the sensor inputs transit from  $S1 \rightarrow S2$  (i.e. disc moves forward), internal state machine will generate a FCounter update pulse to increase the FCounter by 1 and move into S2 state. If the sensor inputs transit from  $S1 \rightarrow S0$  (i.e. disc moves backward), internal state machine will generate a RCounter update pulse to increase the RCounter by 1 and move into S0 state. If the sensor inputs transit



from S1 -> S3, PCounter will generate an invalid state event and remain in the S1 state, which will make invalid state flag set and CPU is interrupted if the SINVIE of PCNT\_CTRL is set .

Table 21-7 shows the PCounter's internal state machine behaviour on various present state and sensor inputs.

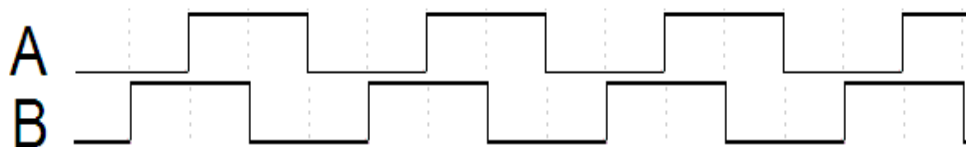
**Table 21-7. PCounter Internal State Machine**

Present State	Sensor Input	Next State	FCounter Update Pulse	RCounter Update Pulse	Invalid State Event
S0	2'b00	S0	No	No	No
S0	2'b01	S1	Yes	No	No
S0	2'b10	S0	No	No	Yes
S0	2'b11	S3	No	Yes	No
S1	2'b00	S0	No	Yes	No
S1	2'b01	S1	No	No	No
S1	2'b10	S2	Yes	No	No
S1	2'b11	S1	No	No	Yes
S2	2'b00	S2	No	No	Yes
S2	2'b01	S1	No	Yes	No
S2	2'b10	S2	No	No	No
S2	2'b11	S3	Yes	No	No
S3	2'b00	S0	Yes	No	No
S3	2'b01	S3	No	No	Yes
S3	2'b10	S0	No	Yes	No
S3	2'b11	S3	No	No	No

#### 21.4.1.4 Two-Signal Gray Mode

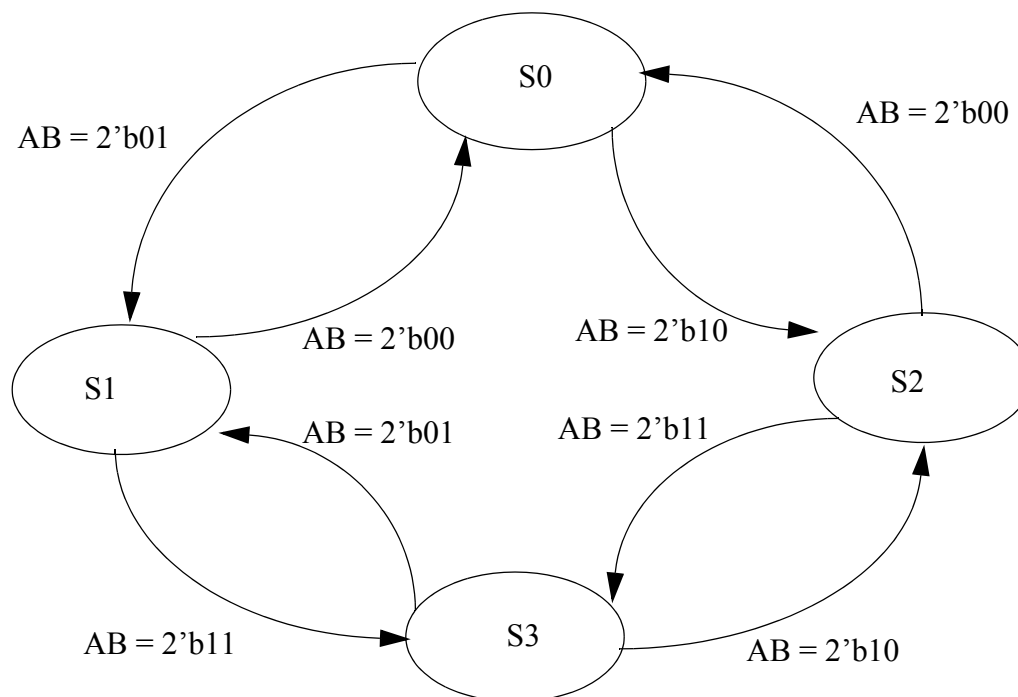
PCounter must be programmed in this mode if it is connected to a gray encoded rotary disc having two contacts or quadrature LC sensor. In this mode, both signals cumulatively reflect the states of the device. This mode is also called the quadrature mode.

Contacts on the disc reflect the position (state) of the rotary disc or sensor. Figure 21-19 shows the waveform of both contacts (A and B) when the disc is moving in the forward direction.



**Figure 21-19. Sensor Inputs for Rotary Disc Moving Forward**

PCounter gets the rotary disc's state by monitoring both contacts. It keeps its internal state machine's state the same as the rotary disc's position. When the rotary disc changes its state in rotating, internal state machine also makes transition to the same state. Figure 21-20 shows the PCounter's internal state machine transition diagram for two-signal gray mode.



**Figure 21-20. State Diagram for Two-Signal Gray Mode**

In the above state diagram, S0, S1, S2 and S3 are the states of PCounter's internal state machine for two contacts gray encoded disc or 90 degree apart sensor configuration. These states corresponds to the rotary disc's states or sensor inputs one to one. When the disc changes its state in rotating, PCounter's state machine also makes the same transition.

Depending on the present rotary disc's position (state), some states are valid while others are invalid. For example, suppose rotary disc's state is S1. If the disc moves forward, it will move to S3 state. If the disc moves backward, it will move to S0 state. So the valid states are S0 and S3 if the rotary disc is in S1 state. Disc can not move directly into S2 i.e. S1 --> S2. It has to follow the S1-> S3 -> S2 or S1-> S0 -> S2 sequence. So the S2 is an invalid state. If S2 comes on the sensor inputs, it might be due to external unwanted factors.

In the above example, if the sensor inputs transit from S1 -> S3 (i.e. disc moves forward), internal state machine will generate an FCounter update pulse to increase the FCounter by 1 and move into S3 state. If the sensor inputs transit from S1 -> S0 (i.e. disc moves backward), internal state machine will generate a RCounter update pulse to increase the RCounter by 1 and move into S0 state. If the sensor inputs transit from S1 -> S2, PCounter will generate an invalid state event and remain in the S1 state, which will make invalid state flag set and CPU is interrupted if the SINIE of PCNT\_CTRL is set .

Table 21-8 shows the PCounter's internal state machine behaviour on various present state and sensor inputs.

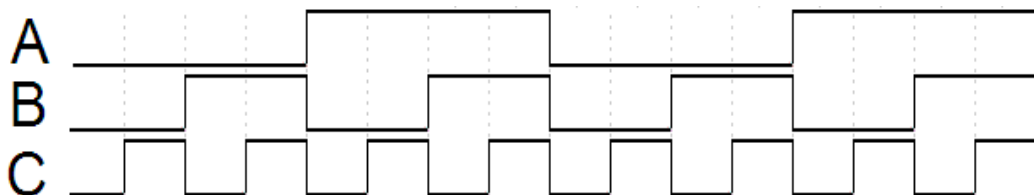
**Table 21-8. PCounter Internal State Machine**

Present State	Sensor Input	Next State	FCounter Update Pulse	RCounter Update Pulse	Invalid State Event
S0	2'b00	S0	No	No	No
S0	2'b01	S1	Yes	No	No
S0	2'b10	S2	No	Yes	No
S0	2'b11	S0	No	No	Yes
S1	2'b00	S0	No	Yes	No
S1	2'b01	S1	No	No	No
S1	2'b10	S1	No	No	Yes
S1	2'b11	S3	Yes	No	No
S2	2'b00	S0	Yes	No	No
S2	2'b01	S2	No	No	Yes
S2	2'b10	S2	No	No	No
S2	2'b11	S3	No	Yes	No
S3	2'b00	S3	No	No	Yes
S3	2'b01	S1	No	Yes	No
S3	2'b10	S2	Yes	No	No
S3	2'b11	S3	No	No	No

### 21.4.1.5 Three-Signal Binary Mode

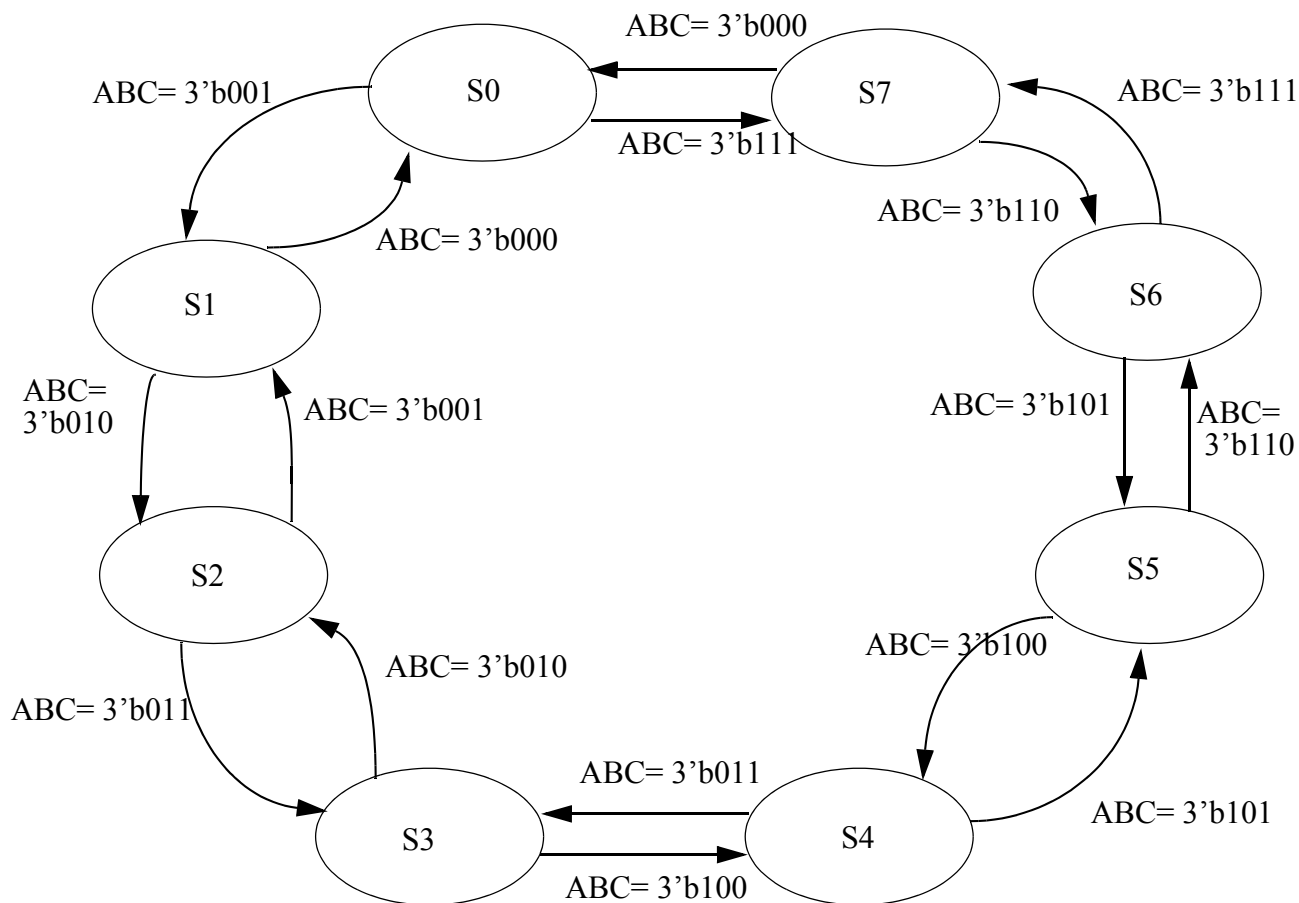
PCounter must be programmed in this mode if it is connected to a rotary disc having three contacts and is binary encoded. In this three-signal binary mode, all the three signals cumulatively reflect the state of the device. Below is a typical example.

Contacts on the disc reflect the position (state) of the rotary disc. [Figure 21-21](#) shows the waveform of all the three contacts (A, B and C) when the disc is moving forward.



**Figure 21-21. Sensor Inputs for Rotary Disc Moving Forward**

PCounter gets the rotary disc's state by monitoring all the three contacts. It keeps its internal state machine's state the same as the rotary disc's position. When the rotary disc changes its state in rotating, internal state machine also makes transition to the same state. [Figure 21-22](#) shows the PCounter's internal state machine transition diagram for three-signal binary mode.



**Figure 21-22. PCounter Internal State Machine Transition Diagram For Three Signal Binary Mode**

In the above state diagram, S0, S1, S2, S3, S4, S5, S6, and S7 are the possible states of PCounter's internal state machine for three contacts binary encoded disc. These states correspond to rotary disc's states one to one. When the disc changes its state in rotating, PCounter's state machine also makes the same transition.

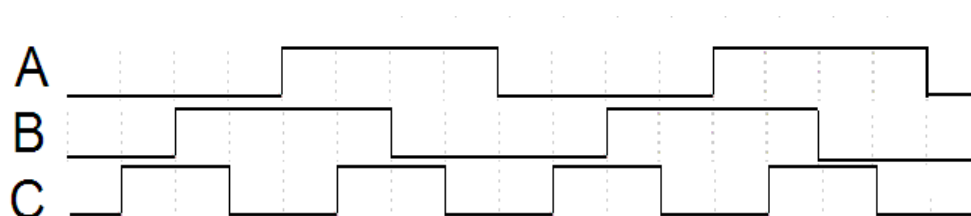
Depending on the present rotary disc's position (state), some states are valid while others are invalid. For example, suppose the current rotary disc's state is S1. If the disc moves forward, disc will move to S2 state. If disc moves backward, then disc will move to S0 state. So the valid states are S0 and S2, if the rotary disc is in S1 state. Disc can not move directly to S3, S4, S5, S6, or S7. So S3, S4, S5, S6, and S7 are invalid states if the disc is currently in S1 state. If invalid state comes on the sensor inputs, it might be due to external unwanted factors.

In the above example, if the sensor inputs transit from S1 → S2 (i.e. disc moves forward), internal state machine will generate a FCounter update pulse to increase the FCounter by 1 and move to S2 state. If the sensor inputs transit from S1 → S0 (i.e. disc moves backward), internal state machine will generate a RCounter update pulse to increase the RCounter by 1 and move to S0 state. If the sensor inputs transit from S1 to any invalid state, PCounter will generate an invalid state event and remain in the S1 state.

### 21.4.1.6 Three-Signal Gray Mode

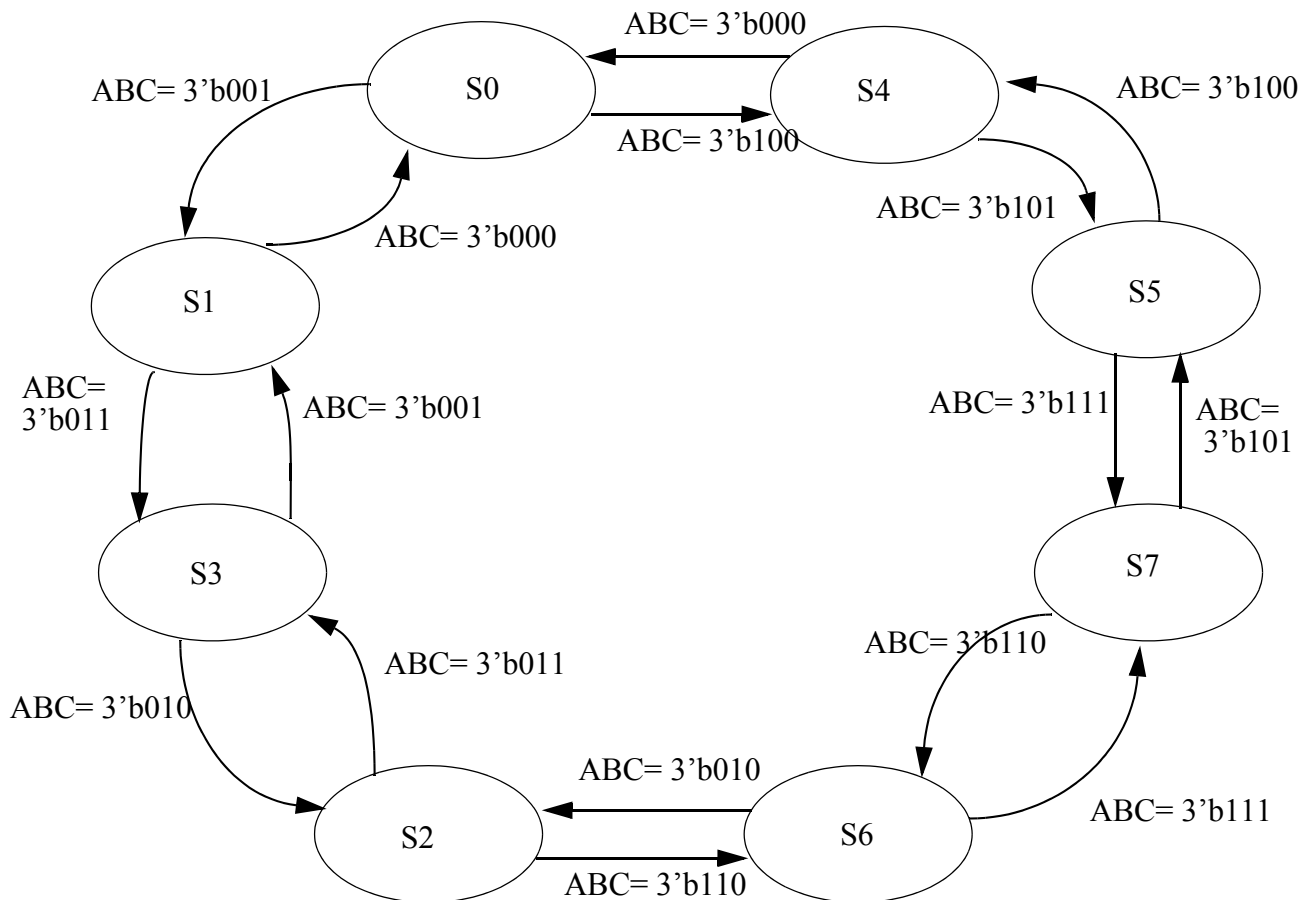
PCounter must be programmed in this mode if it is connected to a rotary disc having three contacts and is gray encoded. In this three-signal gray mode, all the three signals cumulatively reflect the state of the device. Below is a typical example.

Contacts on the disc reflect the position (state) of the rotary disc. [Figure 21-23](#) shows the waveform of all the three contacts (A, B and C) when the disc is moving forward.



**Figure 21-23. Sensor Inputs for Rotary Disc Moving Forward**

PCounter gets the rotary disc's state by monitoring all the three contacts. It keeps its internal state machine's state the same as the rotary disc's position. When the rotary disc changes its state in rotating, internal state machine also makes transition to the same state. [Figure 21-24](#) shows the PCounter's internal state machine transition diagram for three-signal gray mode.



**Figure 21-24. PCounter Internal State Machine Transition Diagram for Three-Signal Gray Mode**

In the above state diagram, S0, S1, S2, S3, S4, S5, S6, and S7 are the states of PCounter's internal state machine for three contacts gray encoded disc. These states correspond to rotary disc's states one to one. When the disc changes its state in rotating, PCounter's state machine also makes the same transition.

Depending on the present rotary disc's position (state), some states are valid while others invalid. For example, suppose rotary disc's state is S1. If the disc moves forward, disc will move to S3 state. If disc moves backward, disc will move to S0 state. So the valid states are S0 and S3, if the current rotary disc is in S1 state. Disc can not move directly to S2, S4, S5, S6, or S7. So S2, S4, S5, S6, and S7 are invalid states if the disc is currently at S1 state. If invalid state comes on the sensor inputs then it might be due to external unwanted factors.

In the above example, if the sensor inputs transit from S1 -> S3 (i.e. disc moves forward), internal state machine will generate a FCounter update pulse to increase the FCounter by 1 and move into S3 state. If the sensor inputs transit from S1 -> S0 (i.e. disc moves backward), then internal state machine will generate a RCounter update pulse to increase the RCounter by 1 and move to S0 state. If the sensor inputs transit from S1 to any invalid state, PCounter will generate an invalid state event and remain in the S1 state.

### 21.4.1.7 Atomic Counter/PWM Mode (Edge- or Centre-Aligned)

Waveform generation block, a sub-block of PCounter, generates edge- or centre-aligned PWM on its channels. CPWMS bit of the PCNT\_CTRL register selects the centre- or edge-aligned mode of the operation.

Although waveform generation block is enabled in all the other modes to generate required waveform to control the sensor, but in this mode all the other sub-blocks of PCounter (except this for one and register space block) are disabled. PCounter in this mode behaves as a normal PWM module which can generate edge- or centre-aligned waveform. See [Section 21.4.5, “PCounter Internal PWM,”](#) for details.

PCounter can be used as an atomic counter in this mode. FCounter register keeps incrementing on writing to PCNT\_FCNTL and reading this register returns the latest count value. This register is used in metering applications to store the energy consumed over a period of time. This counter overflows when FCounter transits from 16'hFFFF to 16'h0000.

#### NOTE

User must keep CHANNEL\_SEL register to 3'b000 when operating in the Atomic Counter/PWM mode.

## 21.4.2 Noise Filtering

A digital filter is implemented for noise on state inputs. It runs on filter\_clk. The filter value can be set by the core in the PCNT\_CTRL register. The FILTER\_VALUE is the number of filter clocks to be counted in detecting changes in the input signal (either 0 to 1 or 1 to 0). If the signal is found at the same level throughout the counting up to FILTER\_VALUE, that value is allowed to go to state machine. Otherwise, it will not be reflected at the filter output. Take caution in making the filter duration equal to 0 or 1 because any glitch on the PCNT[2:0] pins can cause a state change or set the invalid mode flag. When the Pcounter is enabled, the filter value can not be changed on the fly. User needs to disable PCounter to change the filter value.

## 21.4.3 Synchronous and Asynchronous Interrupt of PCounter

### 21.4.3.1 Synchronous Interrupt

A single synchronous interrupt (ipi\_int) is output from the module which is an ORed version of all the below events.

- FCounter overflow flag (if FCOVFIE is set)
- RCounter overflow flag (if RCOVFIE is set)
- State invalid event flag (if SINIE is set)

CPU reads the status register in the interrupt service routine to determine which event has occurred. This interrupt will not be generated when MCU is in stop mode.

### 21.4.3.2 Asynchronous Interrupt

A single asynchronous interrupt (ipi\_int\_async) is output from the module which is an ORed version of all the below events.

- FCounter overflow event (if FCOVFIE is set)
- RCounter overflow event (if RCOVFIE is set)
- State invalid event (if SINVIE is set)

This asynchronous interrupt can be generated even in stop mode and used to wake the device from low power modes. CPU reads the status register in the interrupt service routine to determine which event occurred.

### 21.4.4 Sampling vs Level Sensing

The state of the rotary disc is read through the sensors. In light sensing, sampling method is often used because permanently powered light sensing consumes huge current.

In a sampling method, the LED is energized for a period of time prior to reading the sensors, because it takes time for the light level of the LED to be stable.

Figure 21-25 shows the typical waveform required in sampling method. The sensor activation signal controls the light sensor connected outside. It is made ON for a small duration in one sampling period (which is defined by the max rpm of rotary disc).

It takes the light sensor some time to be stable, so the PCounter start the internal sampling after the time.

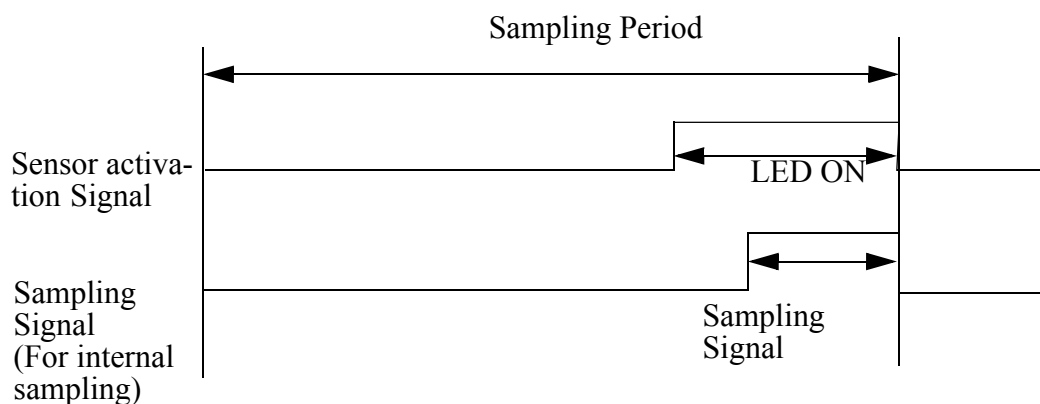


Figure 21-25. Waveform

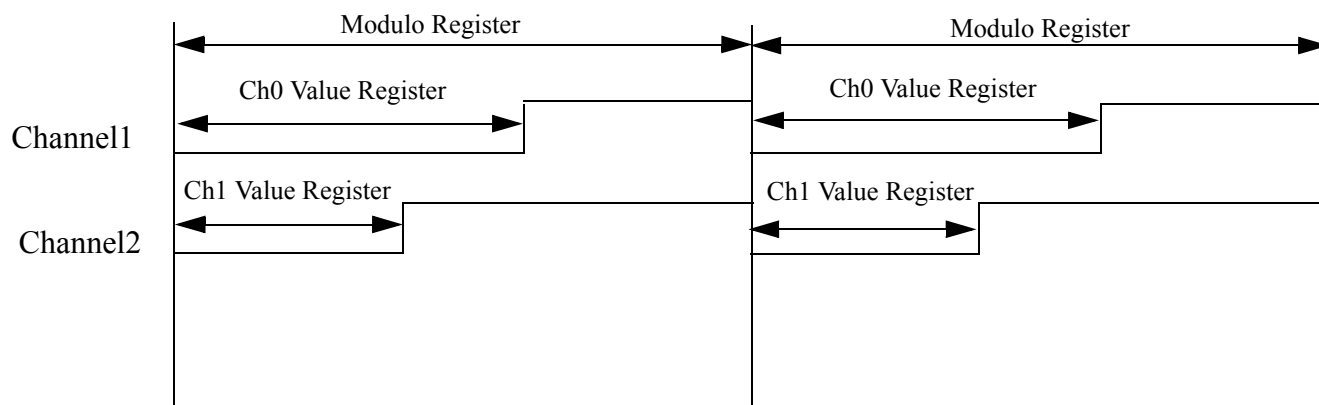
### 21.4.5 PCounter Internal PWM

Waveform generation block, a sub-block of PCounter, generates PWM waveform on its two channels. This sub-block supports below two modes of operation.



### 21.4.5.1 Edge-Aligned PWM

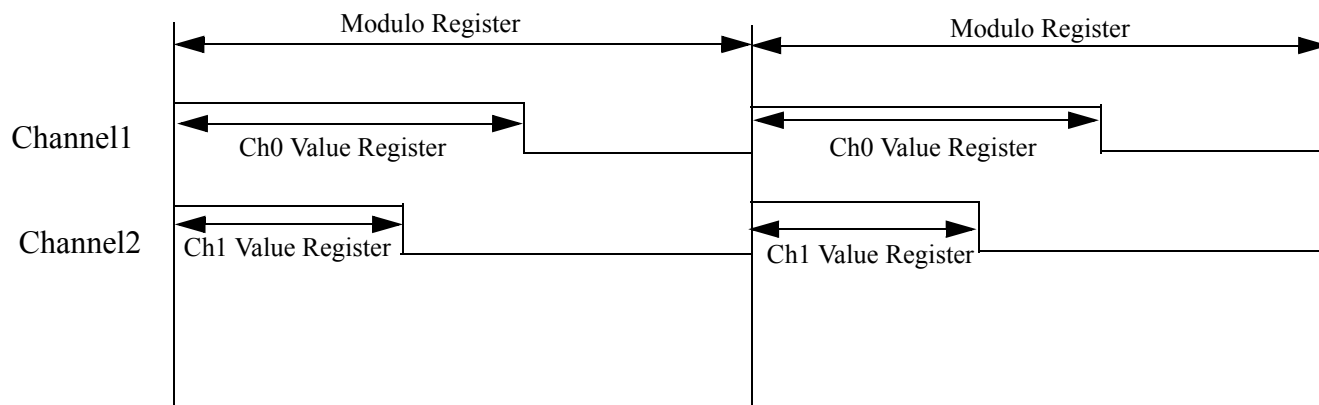
The value of a 16-bit modulo register sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. Figure 21-26 shows the waveform of an edge-aligned PWM channel. This type of PWM signal is called edge-aligned PWM because the leading edges of all PWM signals are aligned with the beginning of the period that is the same for all channels. Polarity of the PWM can be selected by the POL bit of PCNT\_CTRL register. Figure 21-26 and Figure 21-27 show the waveform of an edge-aligned PWM with POL = 1'b0 and POL = 1'b1 respectively.



**Figure 21-26. An Edge-Aligned PWM Waveform ( POL = 1'b0 )**

Duty cycle for PWM\_CH0 (when POL is 1'b0 ) in edge-aligned mode can be calculated by below equation.

$$\text{Duty Cycle} = ((\text{PCNT\_PWM\_MOD} - \text{PCNT\_PWM\_CH0\_VAL} + 1) / (\text{PCNT\_PWM\_MOD} + 1)) \times 100 \quad \text{Eqn. 21-1}$$



**Figure 21-27. An Edge-Aligned PWM Waveform ( POL = 1'b1 )**

Duty cycle for PWM\_CH0 (when POL is 1'b1 ) in edge-aligned mode can be calculated by below equation.

$$\text{Duty Cycle} = 100 - (((\text{PCNT\_PWM\_MOD} - \text{PCNT\_PWM\_CH0\_VAL} + 1) / (\text{PCNT\_PWM\_MOD} + 1)) \times 100) \quad \text{Eqn. 21-2}$$

### 21.4.5.2 Centre-Aligned PWM

Twice the value of a 16-bit modulo register sets the period of the PWM output. The internal counter counts up until it reaches the modulo value and then counts down until it reaches zero. When the count matches the channel value register while counting up, the PWM output becomes active. When the count matches the channel value register while counting down, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. Polarity of the PWM can be selected by the POL bit of PCNT\_CTRL register. Figure 21-28 and Figure 21-28 shows the waveform of an edge-aligned PWM channel with POL = 1'b0 and POL = 1'b1 respectively.

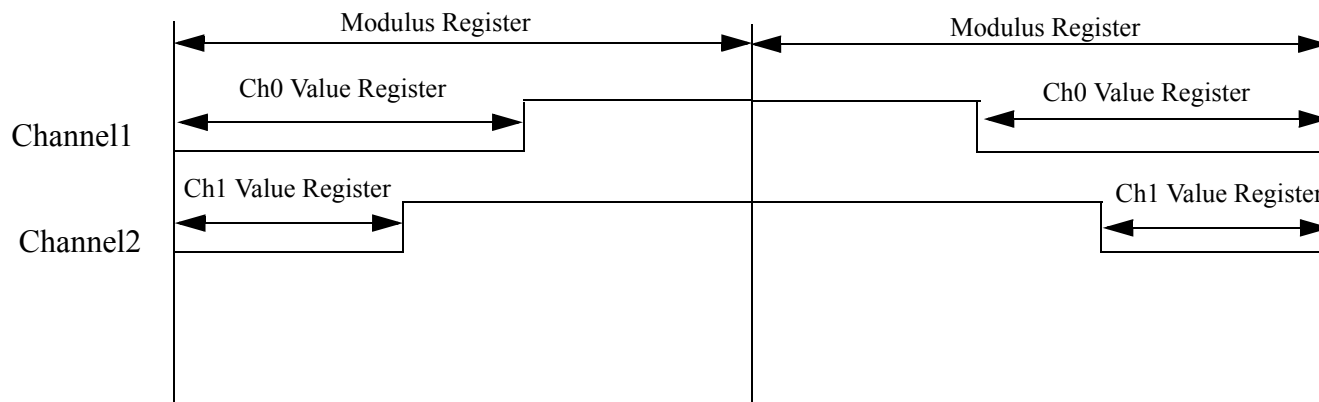


Figure 21-28. A Centre-Aligned PWM Waveform (POL = 1'b0)

Duty cycle for the PWM\_CH0 (when POL is 1'b0 ) in centre-aligned mode can be calculated by below equation.

$$\text{Duty Cycle} = ((\text{PCNT\_PWM\_MOD} - \text{PCNT\_PWM\_CH0\_VAL}) / (\text{PCNT\_PWM\_MOD})) \times 100 \quad \text{Eqn. 21-3}$$

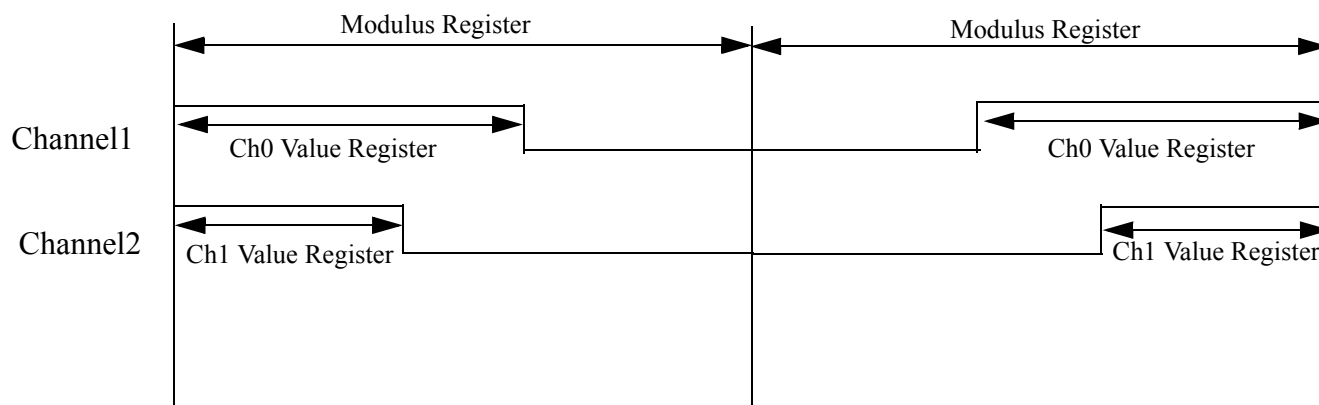


Figure 21-29. A Centre-Aligned PWM Waveform (POL = 1'b1)

Duty cycle for PWM\_CH0 (when POL is 1'b1 ) in centre-aligned mode can be calculated by below equation.

$$\text{Duty Cycle} = 100 - (((\text{PCNT\_PWM\_MOD} - \text{PCNT\_PWM\_CH0\_VAL}) / (\text{PCNT\_PWM\_MOD})) \times 100) \quad \text{Eqn. 21-4}$$

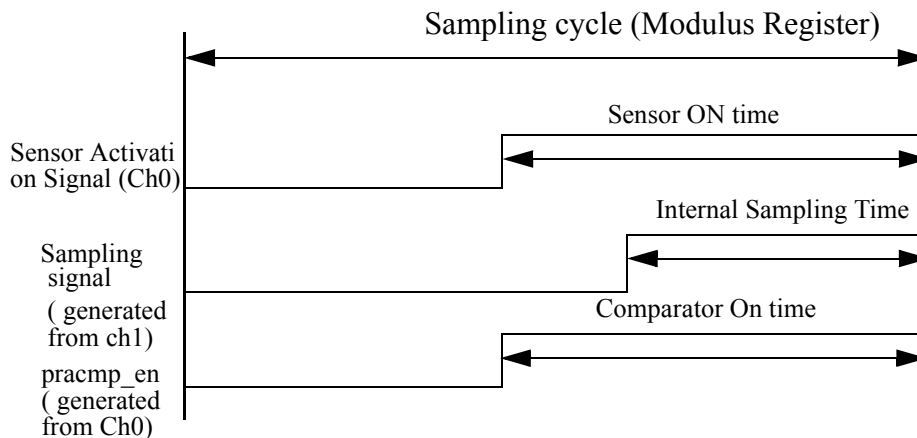
Table 21-9 shows the minimum and maximum period of the PWM cycle, which we can get by PCounter (assuming the max value of PCNT\_PWM\_MOD\_WIDTH is 16 ).

**Table 21-9. Min & Max Period of PMW Cycle on filter\_clk frequency**

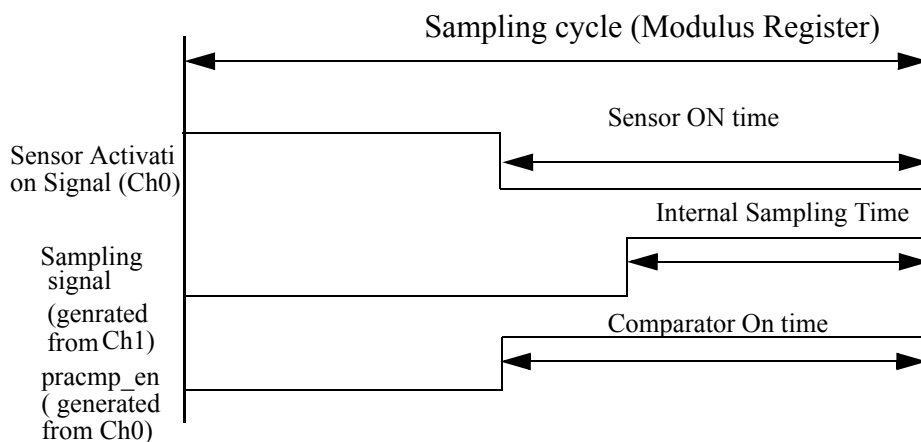
Name of	Filter Clk ( 32 kHz)		Filter Clk ( 512 Hz)		Filter Clk ( 20 MHz )	
	Min	Max	Min	Max	Max	Max
PWM cycle (edge)	61.50 $\mu$ s	~2 sec	3.9 ms	~125 sec	0.1 $\mu$ s	~.0032 sec
PWM cycle (centre)	156.25 $\mu$ s	~4 sec	9.8 ms	~250 sec	0.25 $\mu$ s	~.0064 sec

## 21.4.6 Sensor Activation Signal & Internal Sampling Signal Generation

As described in Section 21.4.4, the sampling method is an efficient method for rotary disc reading. PCounter’s internal PWM can be used to generate the required sensor activation signal to control the sensor and the sampling signal for subsequent sampling. User may use either PWM mode (edge- or centre-aligned) to generate the activation signal and sampling signal . Figure 21-30 shows the typical waveform generated by internal PWM in the edge-aligned mode. See Section 21.4.5, “PCounter Internal PWM,” for details. The sensor activation signal comes out of channel 0, while the sampling enable signal comes out of Channel 0. Additionally, a comparator signal is output (which is the same as channel 0 but without polarity ) to enable/disable comparator on the MCU.



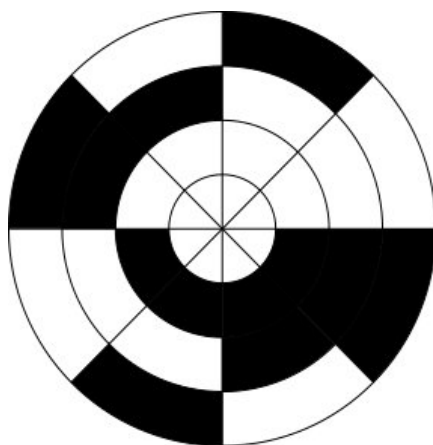
**Figure 21-30. Timing Diagram for Sensor with ACTIVE HIGH Activation Signal ( Pol = 1'b0 )**



**Figure 21-31. Timing Diagram for Sensor with ACTIVE HIGH Activation Signal ( Pol = 1'b1 )**

### Sampling Cycle:

Rotatory disc is divided into many sectors to increase the granularity of the counting. [Figure 21-32](#) shows an example of binary encoded disc divided into eight sectors. In this case, we are able to detect if the disc moves by 45 degree and increases the FCounter by 1 if the movement is in forward direction (RCounter in case movement is in backward direction).



**Figure 21-32. Binary Encoded Disc Divided in Eight Sectors**

Sampling cycle time is the time rotatory disc takes to cross one sector. In actual, disc does not run at a fixed RPM. In that case, the max RPM determines the sampling cycle time. Below is the formula to calculate the sampling cycle time. Let the max rpm of the rotatory disc be S.

$$\text{Total no of sector crossed per minute} = \text{MaxRPM} \times \text{Numbers of sectors on the disc} \quad \text{Eqn. 21-5}$$

$$\text{Total no of sector crossed per Second} = (\text{MaxRPM} \times \text{Numbers of sectors on the disc}) / 60 \quad \text{Eqn. 21-6}$$

$$\text{Sampling cycle time} = 60 / (2 \times \text{MaxRPM} \times \text{Numbers of sectors on the disc}) \quad \text{Eqn. 21-7}$$

Sampling cycle time is counted by an internal counter inside the waveform generation block if the filter clock frequency is 32 kHz. For an power efficient system, count value is up to the internal counter used and calculated as below

$$\text{Counter final count value} = (60 / (2 \times \text{MaxRPM} \times \text{Numbers of sectors on the disc})) / (1 / 32K) \quad \text{Eqn. 21-8}$$

In [Figure 21-30](#), sampling cycle time is the value of the PWM modulus register (PCNT\_PWM\_MOD). The internal counter width is given by the parameter PCNT\_PWM\_MOD\_WIDTH. If PCNT\_PWM\_MOD\_WIDTH is 16, we can count up to 65536. If the frequency of the filter clock is 32 kHz, we can have maximum sampling cycle time around two seconds.

Sensor on time is the time the sensor is active in one sampling cycle. Its duration is different from the PWM Modulus Register and PWM Channel1 Value Register.

Sampling signal high time is the time the sensor inputs are valid. When the sampling gets high, the PCounter starts processing the state inputs. This signal is used internally. Its duration is different from the PWM Modulus Register and PWM Channel2 Value Register.

## 21.4.7 Initialization Sequence

Initialize the PCounter with the following steps:

1. Out of reset, disable the PCounter.
2. Program PCNT\_PWM\_MOD, PCNT\_PWM\_CH0\_VAL and PCNT\_PWM\_CH1\_VAL registers for proper generation of sensor activation signal and sampling signal/ PWM signals.
3. Program the PCNT\_CTRL register with the required filter value, interrupt enablers, mode, PWM mode and channel selection with PCNT\_EN 1'b1.

At the top level, PCounter state inputs are of three bits as PCNT[2:0]. Depending upon the rotary disc interfaced outside, user must enable or disable the individual input. Before entering the filters, PCNT[2:0] gets remapped as shown in [Table 21-10](#).

**Table 21-10. Bit Remapping Based on CHANNEL\_SEL Register**

CHANNEL_SEL	Sensor Input Signals Selected (Entering PCounter State Machine)	Comment
000	3'b000	No channel selected
001	{2'b00,PCNT[0]}	Only one channel is ON. PCNT[0] is taken as LSB.
010	{2'b00,PCNT[1]}	Only one channel is ON. PCNT[1] is taken as LSB.
011	{1'b0,PCNT[1],PCNT[0]}	Two channels are ON. PCNT[0] is taken as LSB. PCNT[1] is taken as MSB.
100	{1'b0,1'b0,PCNT[2]}	Only one channel is ON. PCNT[2] is taken as LSB.

**Table 21-10. Bit Remapping Based on CHANNEL\_SEL Register (continued)**

<b>CHANNEL_SEL</b>	<b>Sensor Input Signals Selected (Entering PCounter State Machine)</b>	<b>Comment</b>
101	{1'b0, PCNT[2], PCNT[0]}	Two channels are ON. PCNT[0] is taken as LSB. PCNT[2] is taken as MSB.
110	{1'b0, PCNT[2], PCNT[1]}	Two channels are ON. PCNT[1] is taken as LSB. PCNT[2] is taken as MSB.
111	{PCNT[2], PCNT[1], PCNT[0]}	All the three channels are ON. PCNT[2] is MSB, and PCNT[0] as LSB.

4. PCNT\_PWM\_MOD, PCNT\_PWM\_CH0 and PCNT\_PWM\_CH1 get locked when the PCNT\_EN bit of PCNT\_CTRL is asserted. Users are not able to update these registers till PCNT\_EN = 1'b1.

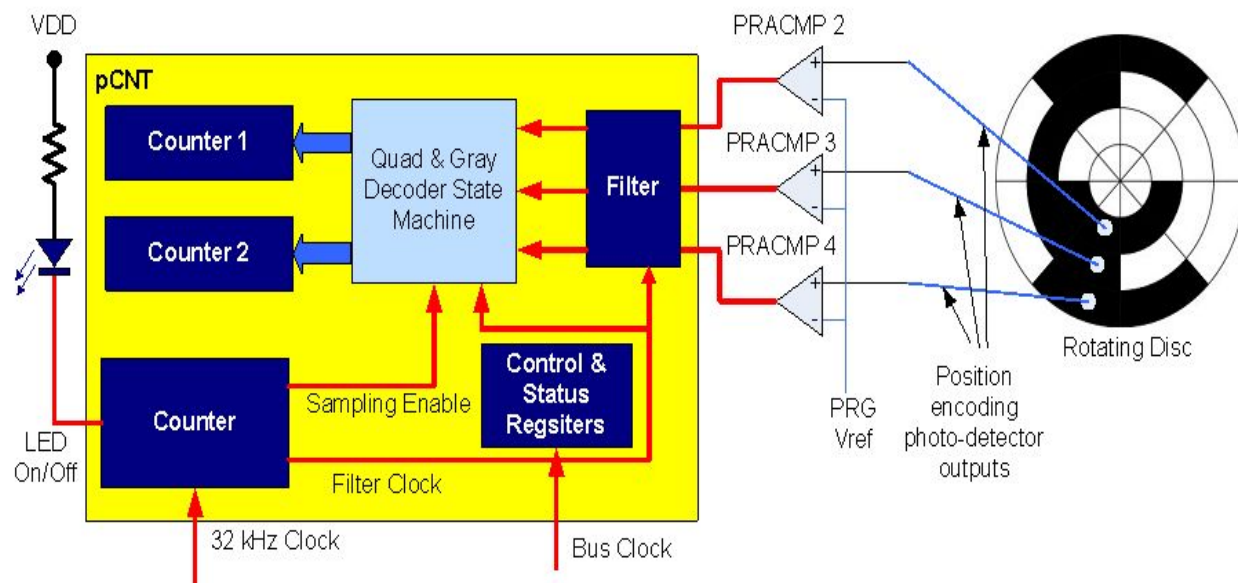
Most bits of PCNT\_CTRL also gets locked if the PCNT\_EN is high except for the interrupt enabler bits.

If user wants to update above locked registers, he/she has to disable the PCounter by making an IPS write with PCNT\_EN as low (other bits of the ips\_wdata are ignored), then update the required registers if the register to be updated is other than PCNT\_CTRL. And update the PCNT\_CTRL with new value of control bits and PCNT\_EN bit 1'b1.

## 21.5 Typical Use Case

Figure 21-33 illustrates how the PCounter is interfaced. In the diagram, a gray encoded rotary device with three contacts is interfaced to the PCounter. States of rotary device, which are read through sensor, are analog in nature so they get converted into digital signals by a PRACMP (Programmable Analog comparator). Outputs of those comparators are connected to the PCounter's input.

In Figure 21-33, a sensor interface is on the left side. In case of light sensing, LEDs are used to read the states. These LEDs are energized for a small duration in a sampling cycle. PCounter samples in that period. This saves the power of the complete sensing system.



**Figure 21-33. PCounter Interfacing**





## Chapter 22

# Programmable Delay Block (S08PDBV1)

### 22.1 Introduction

The primary function of the programmable delay block is simply to provide controllable delays from the either an external trigger, or a programmable interval tick, to the sample trigger input of one or more ADCs.

#### NOTE

In this chapter, there are 2 trigger signals for each channel (TriggerA and TriggerB). Actually only TriggerA is available in this MCU. TriggerB is not active. Please ignore the TriggerB description in this chapter.

There are only two channels in this MCU.

All accesses to the registers must be 16-bit width. Hence two 8-bit accesses are necessary to complete a single transaction on PDB and the two addresses should differ only in the bit 0. In case one 8-bit access is done and a new address is provided, PDB gasket will flush the previous access and continue a new with the latest access.

Registers in PDB will be updated when both bytes are provided to the gasket.

Interleaved 8-bit read and write accesses are supported. In case the interleaved accesses are to the same address then the read access will return the old value and not the new written value. A fresh read access will give the new written value. For example, if 8-bit access are done as WR -> RD -> WR -> RD to the same address then read would give the value of register that was present before the current write value.

In case of two 8-bit reads to a register (e.g. 0x00 and 0x01) are spaced quite far in time and the register value changes in between, then the second read would still return the previous value as that would have been latched in the PDB gasket.

## 22.1.1 PDB Clock Gating

The bus clock to the PDB module can be gated on and off using the PDB bit in SCGC4. This bit is cleared after any reset, which disables the bus clock to this module. To conserve power, the PDB bit can be cleared to disable the clock to this module. See [Section 5.7, “Peripheral Clock Gating,”](#) for details.

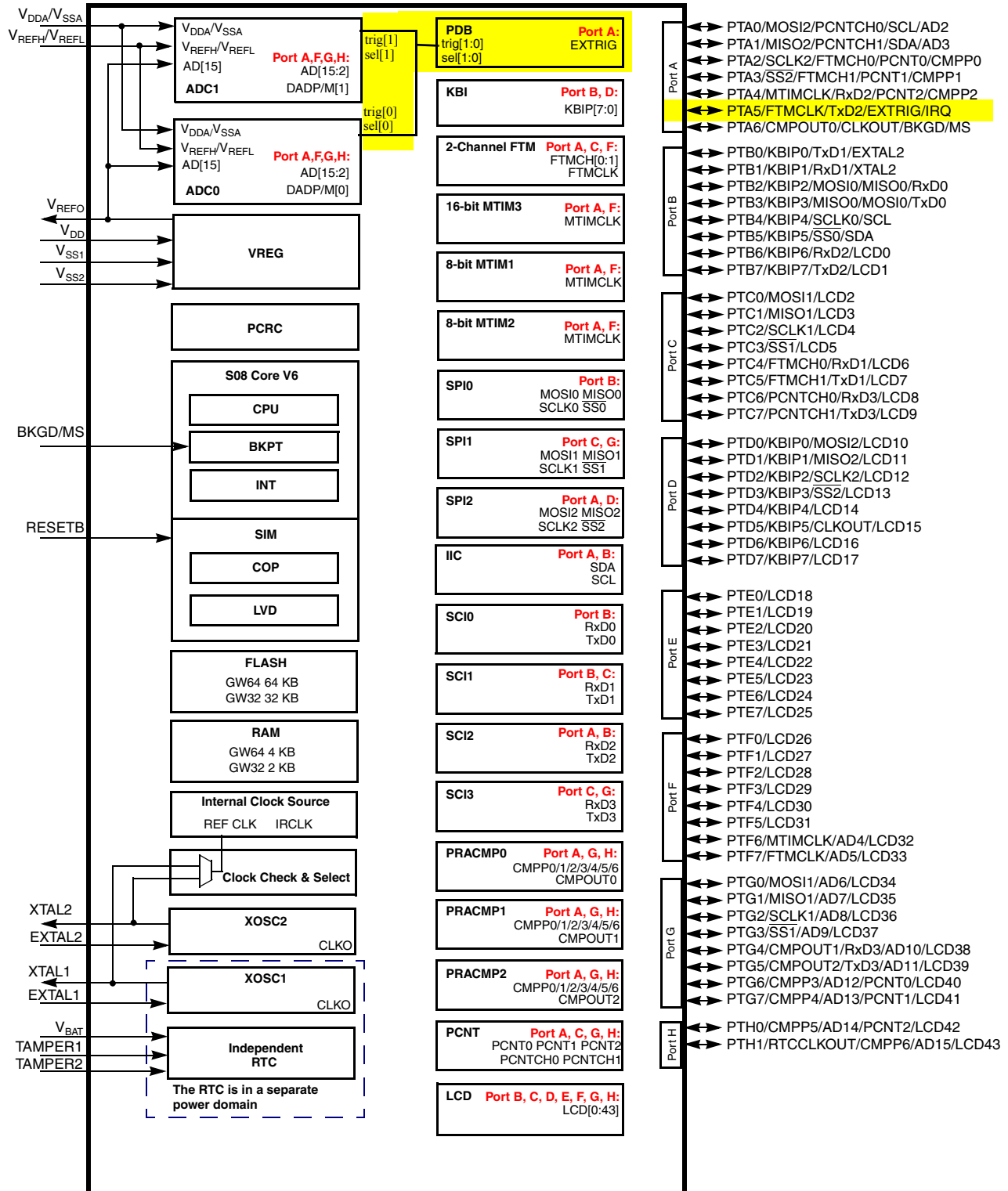


Figure 22-1. MC9S08GW64 Series Block Diagram Highlighting PDB Module and Pins

## 22.1.2 Features

- Positive transition of trigger\_in will initiate the counter
- Up to four configurable channels
  - The number of channels implemented is equivalent to the number of ADCs.
  - Each channel can supply two pre-trigger events to two ADC trigger select inputs.
  - Each trigger output is individually controlled.
- Continuous trigger or single shot mode supported
- Bypass mode supported
- Each trigger output can be independently enabled.
- One programmable delay interrupt
- One sequence error interrupt

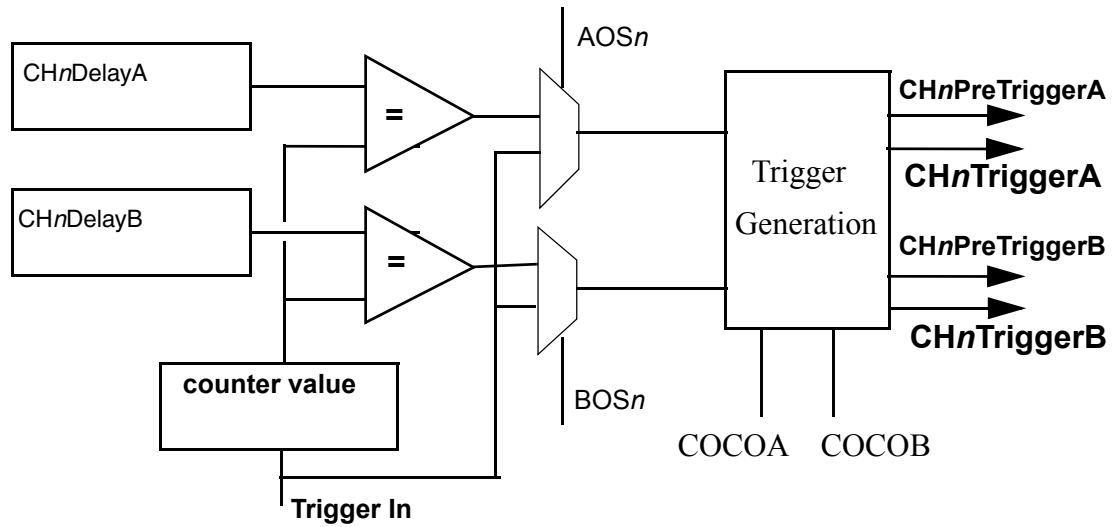
## 22.1.3 Modes of Operation

Modes of operation include:

- Disabled: Counter is off and all pre-trigger and trigger outputs are low.
- Enabled OneShot: Counter is enabled & restarted at count zero upon receiving a positive edge on the trigger input. Each TriggerA and TriggerB will see only one output trigger per input trigger.
- Enabled Continuous: Counter is enabled & restarted at count zero. The counter will be rolled over to zero again when the count reaches the value specified in the PDBMOD 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 only one of the trigger output; therefore this mode can be used in conjunction with any of the above.

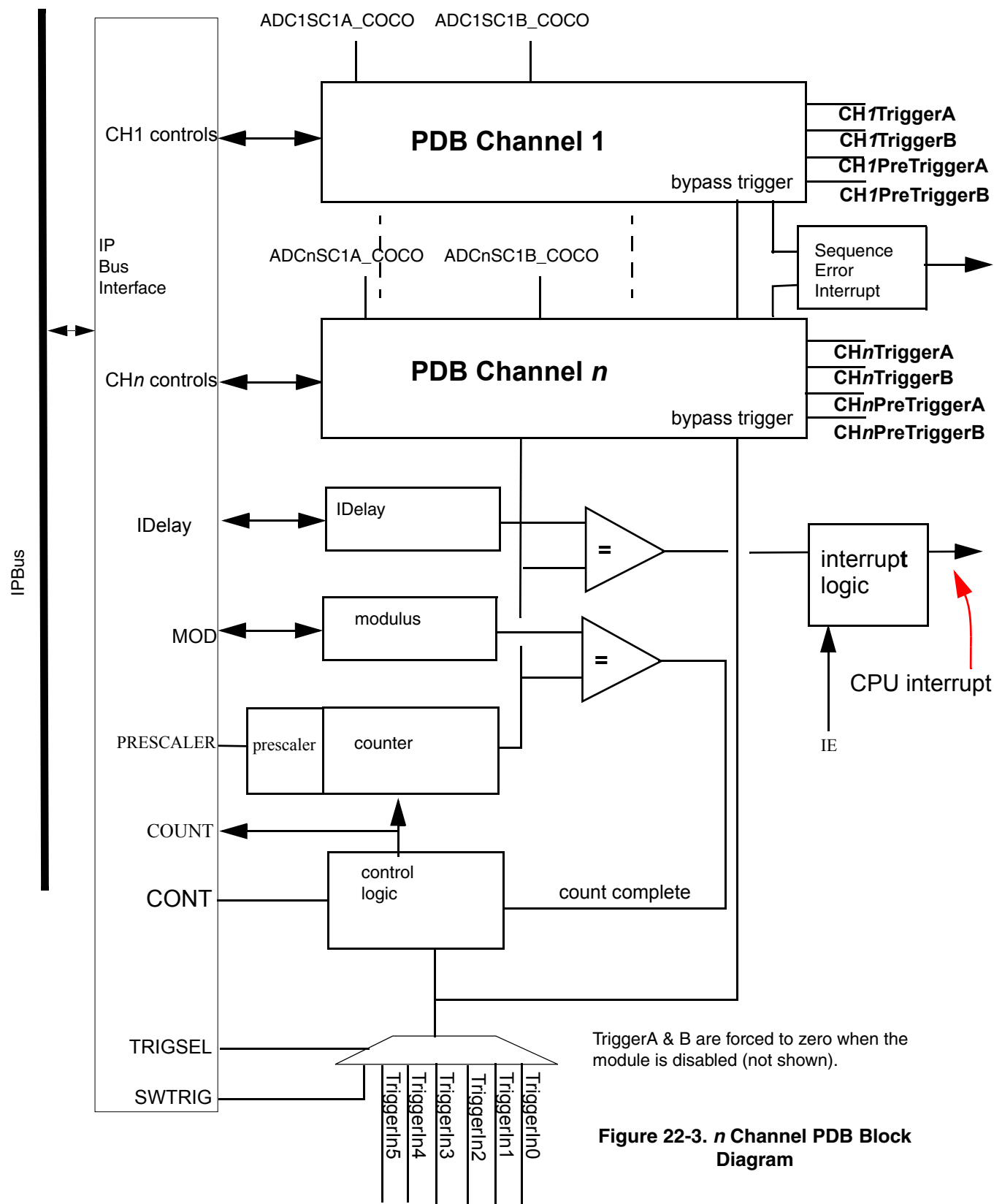
## 22.1.4 Block Diagram

[Figure 22-2](#) and [Figure 22-3](#) illustrate the basic structure of the PDB block.



Not shown are the ENA and ENB controls for each channel. These can be used to gate off the channel outputs.

**Figure 22-2. PDB Channel Block Diagram**



## 22.2 Memory Map and Registers

### 22.2.1 Memory Map

The memory map of PDB will expand as the number of channels increases above 1. The number of PDB channels implemented in this MCU is equivalent to the number of ADC blocks. In Table 22-1,  $n$  is the channel number.

Offset (bytes)	Register	Description
0x00	PDBSC	PDB Status & Control Register
0x02	PDBMOD	PDB Counter Modulus Register
0x04	PDBCNT	PDB Counter Value (READ ONLY)
0x06	PDBIDLY	PDB Comparison Value for Interrupt Timer
0x08 + ( $n \times 0x08$ )	PDBCH $n$ CR	PDB Channel $n$ Control Register
0x08 + ( $n \times 0x08$ ) + 2	PDBCH $n$ DLYA	PDB Channel $n$ Delay A Register
0x08 + ( $n \times 0x08$ ) + 4	PDBCH $n$ DLYB	PDB Channel $n$ Delay B Register
0x08 + ( $n \times 0x08$ ) + 6	Reserved	Reserved

Table 22-1. PDB Memory Map

Note that the memory map extends four extra registers per PDB channel.

### 22.2.2 Registers Descriptions

#### 22.2.2.1 PDB Status and Control Register (PDBSC)

This register contains status and control bits for the Programmable Delay Block. The counter is enabled if EN has been set to one.

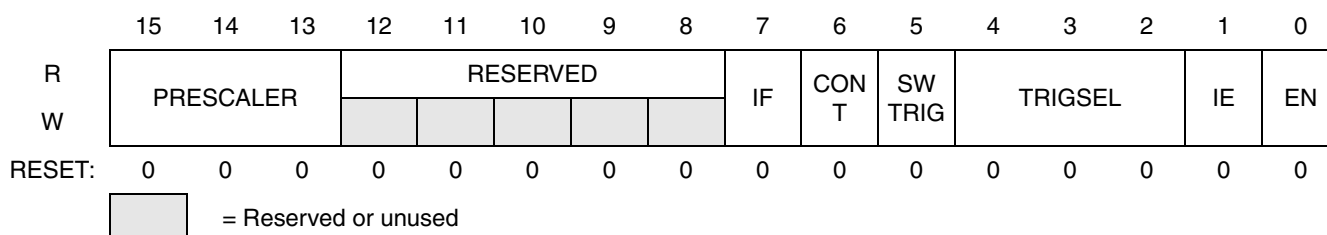


Figure 22-4. PDB Status and Control Register (PDBSC)

Table 22-2. PDBSC Register Field Descriptions

Field	Description
15:13 PRESCALE R	<b>Clock Prescaler Select</b> 000 = timer uses peripheral clock 001 = timer uses peripheral clock / 2 010 = timer uses peripheral clock / 4 011 = timer uses peripheral clock / 8 100 = timer uses peripheral clock / 16 101 = timer uses peripheral clock / 32 110 = timer uses peripheral clock / 64 111 = timer uses peripheral clock / 128
12:8 RESERVED	<b>RESERVED.</b> Write these register bits as zero.
7 IF	<b>Interrupt Flag</b> 0 = A comparison event has not been detected (The count value has equaled IDELAY at some point) 1 = A comparison event has been detected (In this, or a previous PDB cycle, the count has equaled IDELAY) If the module is enabled (EN=1) and the interrupt enable (IE) bit has been set, an interrupt will be issued when the IF bit is set. This bit can be cleared by writing a one to it. The IF bit will be set (when COUNT=IDELAY) regardless of whether or not the interrupt has been enabled.
6 CONT	<b>Continuous Mode Enable</b> 0 = Module is in OneShot mode 1 = Module is in continuous mode
5 SWTRIG	<b>Software Trigger</b> — When TRIGSEL=3'b111 and the module is enabled, writing a one to this field will trigger a reset and restart of the counter. Alternately, if TriggerA or TriggerB are bypassed, it will propagate there immediately. This bit always reads as zero.
4:2 TRIGSEL	<b>Input Trigger Select</b> 000 = Comparator 1 output 001 = Comparator 2 output 100 = EXTRIG 111 = SWTRIG All other values are reserved.
1 IE	<b>IDelay match Interrupt Enable</b> 0 = IDelay Interrupt is not enabled 1 = IDelay Interrupt is enabled Note: This bit does not affect the sequence error interrupt. The sequence error interrupt is always enabled, when the PDB is enabled.
0 EN	<b>PDB Enable</b> 0 = Module Counter is Disabled 1 = Module Counter is Enabled Setting EN=0 will set the count register to zero.



### 22.2.2.2 PDB Modulus Register (PDBMOD)

This register specifies the period of the counter in terms of bus clock cycles. When the counter reaches this value, it will be reset back to all zeros. If PDBSC\_CONT is set to one, the count will begin anew.

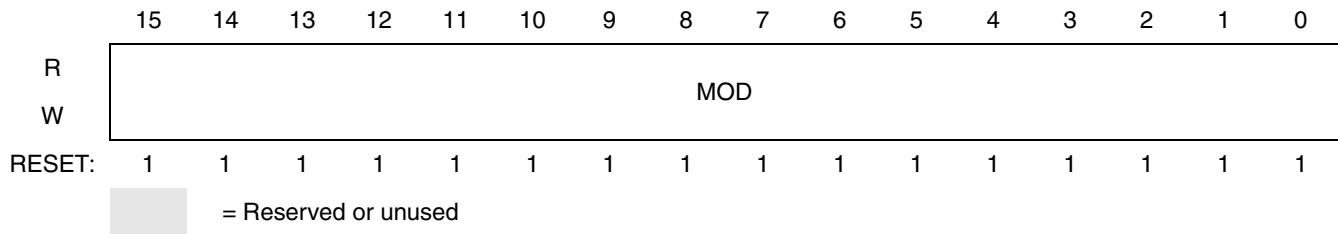


Figure 22-5. PDB Modulus Register (PDBMOD)

### 22.2.2.3 PDB Counter Register (PDBCNT)

This register can be used to read the current value of the counter. It is READ ONLY.

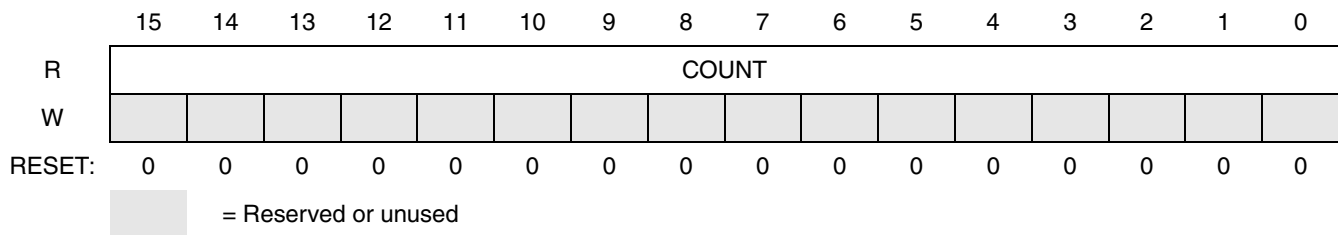


Figure 22-6. PDB Counter Register (PDBCNT)

This register is set to zero when PDBSC\_EN = 0. The count does not start until a hardware or software trigger event occurs.

### 22.2.2.4 PDB Interrupt Delay Register (PDBIDLY)

This register can be used to read and write the value used to schedule the PDB IDelay interrupt. This feature can be used to schedule an independent interrupt at some point in the PDB cycle.

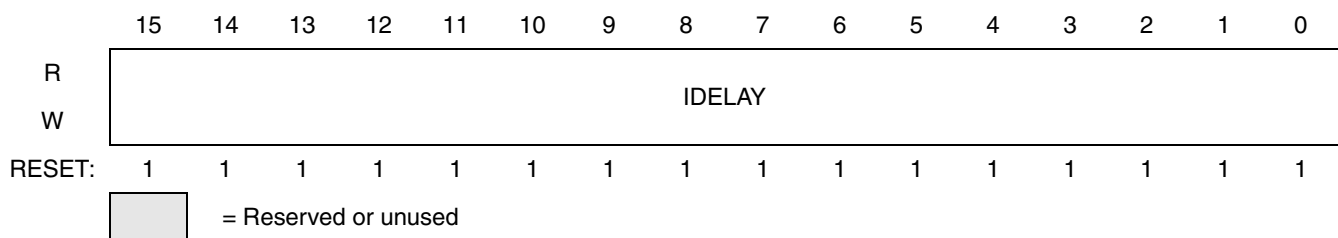


Figure 22-7. PDB Interrupt Delay Register (PDBIDLY)

PDBSC\_IE must be set in order for an interrupt to be issued as a result of the count value equaling IDELAY. However, PDBSC\_IF will be set whenever COUNT=IDELAY.

## 22.2.2.5 PDB Channel $n$ Control Register (PDBCH $n$ CR)

This register contains status and control bits for the channel  $n$  of the Programmable Delay Block. The channel outputs are enabled if either ENA or ENB have been set to one.

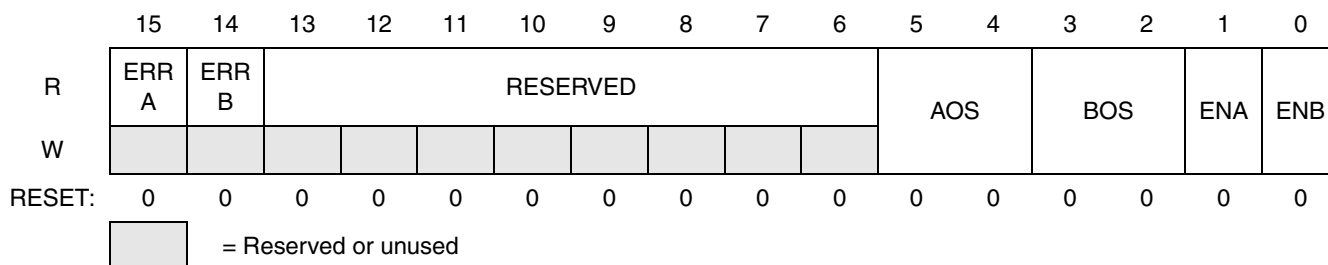


Figure 22-8. Programmable Delay Block Channel  $n$  Control Register (PDBCH $n$ CR)

Table 22-3. PDBCH $n$ CR Register Field Descriptions

Field	Description
15 ERRA	<b>Sequence error on TriggerA</b> 0 = No sequence error on TriggerA 1 = Sequence error is detected on TriggerA. The Delay A is timed out before the previous ADC conversion, which is triggered by TriggerB, is done. A PDB sequence error interrupt is generated when this bit is set. Write 1 clear this bit.
14 ERRB	<b>Sequence error on TriggerB</b> 0 = No sequence error on TriggerB 1 = Sequence error is detected on TriggerB. The Delay B is timed out before the previous ADC conversion, which is triggered by TriggerA, is done. A PDB sequence error interrupt is generated when this bit is set. Write 1 clear this bit.
13:6 RESERVED	<b>RESERVED.</b> Write these register bits as zero.
5:4 AOS	<b>Channel <math>n</math> TriggerA Output Select</b> 00 = Counter delay is bypassed 01 = TriggerA is function of Delay A 10 = Reserved 11 = Reserved
3:2 BOS	<b>Channel <math>n</math> TriggerB Output Select</b> 00 = Counter delay is bypassed 01 = TriggerB is function of Delay B 10 = Reserved 11 = Reserved
1 ENA	<b>TriggerA Enable</b> 0 = PreTriggerA and TriggerA outputs are disabled (and forced to zero) 1 = PreTriggerA and TriggerA outputs are enabled
0 ENB	<b>TriggerB Enable</b> 0 = PreTriggerB and TriggerB outputs are disabled (and forced to zero) 1 = PreTriggerB and TriggerB outputs are enabled

### 22.2.2.6 PDB Channel $n$ Delay A & Delay B Registers (PDBCH $n$ DLYA & PDBCH $n$ DLYB)

These registers are used to specify the delay from assertion of TriggerIn to assertion of PreTriggerA and PreTriggerB out for a given channel. These registers are repeated for each channel present on a given instance of the PDB. These delays are only applicable if the module is enabled and the output trigger in question has not been bypassed. In each case, the delay is in terms of bus clock cycles.

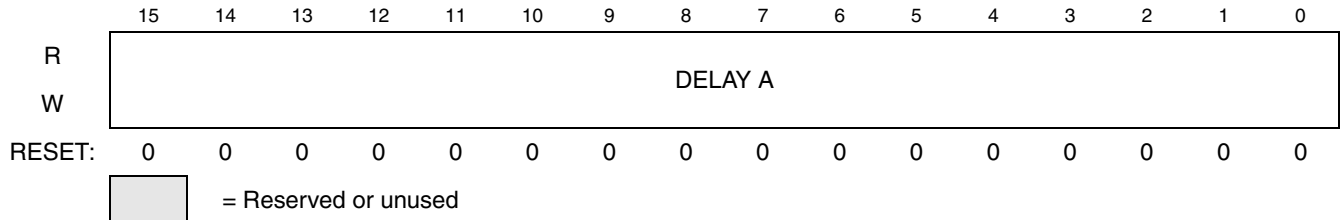


Figure 22-9. PDB Channel  $n$  Delay A Register (PDBCH $n$ DLYA)

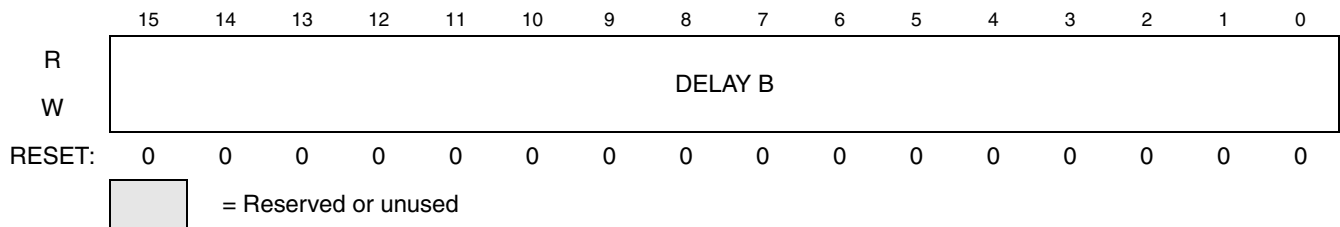


Figure 22-10. PDB Channel  $n$  Delay B Register (PDBCH $n$ DLYB)

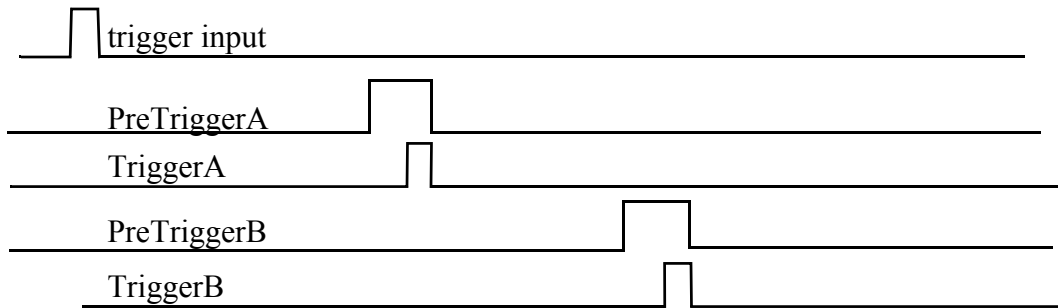
## 22.2.3 Functional Description

The PDB contains a single counter whose output is compared against a minimum of three different digital values. For each channel, 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 X DelayA) + 2 bus clock cycles
- trigger input to Pre-TriggerB = (prescaler X DelayB) + 2 bus clock cycles
- Add one additional bus clock cycle to determine the time at which the trigger outputs change.

Pre-TriggerA and Pre-TriggerB are used to precondition the ADC blocks two bus clock periods prior to the actual measurement trigger. The ADC16V1 on this MCU contains two sets of control and result registers, allowing 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 will be sampled next. When PreTriggerA and TriggerA is asserted, the ADC conversion is triggered with set A of the control and results registers. When PreTriggerB and TriggerB is asserted, the ADC conversion is triggered with set B of the control and results registers.

The signals shown in Figure 22-11 would be used to operate on any one of the ADCs. The trigger delays for the ADCs are independently set via the DELAYA and DELAYB parameters.



**Figure 22-11. Decoupled A & B Trigger Generation**

The value, modulus, is used to reset the counter back to zero at the end of the count. If PDBSC\_CONT is set, the counter will then resume a new count. Otherwise, the timer operation will cease until the next trigger input event occurs.

The PDB Channel  $n$  PreTriggerA and PreTriggerB is connected to ADC $n$  Trigger Select Events ADHWTSA and ADHWTB correspondingly. Either TriggerA or TriggerB can trigger the ADC conversion. When TriggerA triggers the ADC conversion, control and results register set A is used; when TriggerB triggers the ADC conversion, control and results register set B is used.

The Delay A and Delay B registers should be configured to make the next trigger asserted after the previous ADC conversion is finished.

When one conversion, triggered by TriggerA is in progress, the TriggerB output is suppressed, until the ADC $n$ SC1A\_COCO bit is set. If Delay B is timed out during the ADC conversion triggered by TriggerA, the Sequence Error bit PDBCH $n$ SC\_ERRB will be set.

When one conversion, triggered by TriggerB is in progress, the TriggerA output is suppressed, until the ADC $n$ SC1B\_COCO bit is set. If Delay A is timed out during the ADC conversion triggered by TriggerB, the Sequence Error bit PDBCH $n$ SC\_ERRA will be set.

A PDB sequence error interrupt will be generated, if any of the PDBCH $n$ SC\_ERRA or PDBCH $n$ SC\_ERRB on any of the PDB channels is set. The sequence error interrupt cannot be disabled in PDB module.

## Chapter 23

# Voltage Reference (VREFV1)

### 23.1 Introduction

The Voltage Reference (VREF) is intended to supply an accurate voltage output that is trimmable by an 8-bit register in 0.5 mV steps. This reference can be used to provide a reference voltage to external devices or internal analog peripherals such as the ADC, DAC, or ACMP. The voltage reference has two operation modes that provide different levels of load regulation and power consumption. Figure 23-1 is a block diagram of the Voltage Reference.

#### 23.1.1 Overview

The Voltage Reference optimizes the existing bandgap and bandgap buffer system. Unity gain amplifiers ease the design and keep power consumption low. The 8-bit trim register is user accessible so that the voltage reference can be trimmed in application.

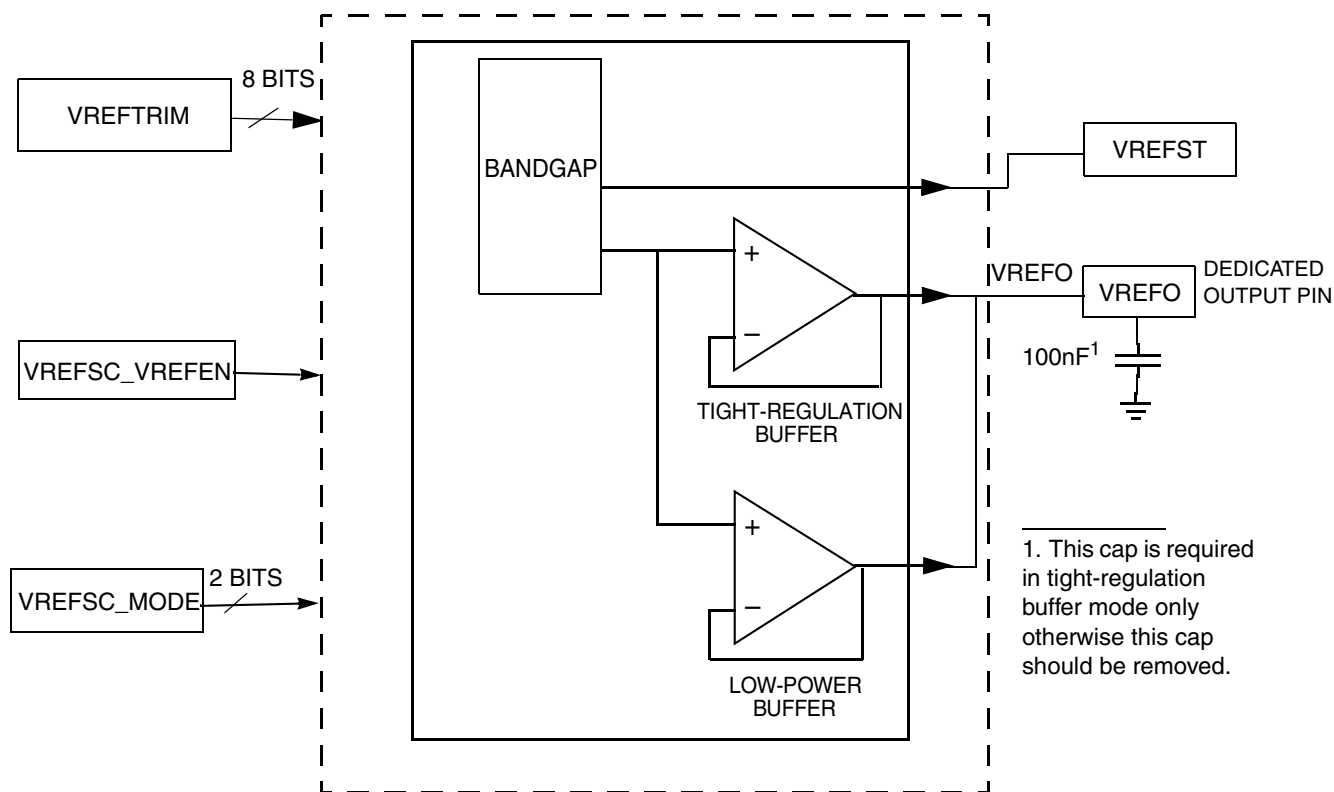


Figure 23-1. Voltage Reference Block Diagram

The voltage reference contains trim resolution of 0.5mV per step. The buffer has modes with improved high-current swing output. In addition, a low-power buffer mode is available for use with ADCs or DACs. The voltage reference can be output on a dedicated output pin<sup>1</sup>.

23.1.2 Features

The Voltage Reference module has the following features:

- Programmable trim register with 0.5mV steps, automatically loaded with factory trimmed value upon reset
- Programmable mode selection:
  - Off
  - Bandgap out (or stabilization delay)
  - Low-power buffer mode
  - Tight-regulation buffer mode
- 1.2 V output at room temperature, 40 ppm/C
- Dedicated output pin VREFO
- Load regulation in tight-regulation mode of 100 uV/mA max
- PSR of 0±0.1 mV DC and -60dB AC

23.1.3 Modes of Operation

The Voltage Reference continues normal operation in Run, Wait, LPRun, and LPWait modes. The Voltage Reference module can be enabled to operate in STOP3 mode.

23.1.4 External Signal Description

Table 23-1 shows the Voltage Reference signals properties.

Table 23-1. Signal Properties

Name	Function	I/O	Reset	Pull Up
VREFO	Internally generated voltage reference output	O	—	—

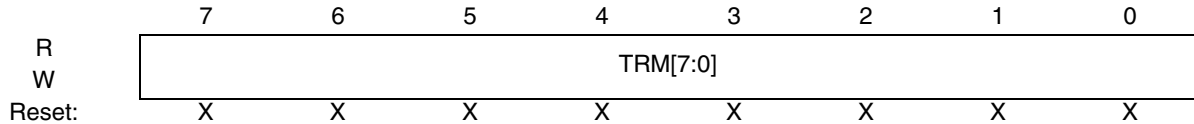
1. In Tight-Regulation buffer mode, a 100nF capacitor is required to be connected between the VREFO pad and the ground.

## 23.2 Memory Map and Register Definition

### 23.2.1 VREF Trim Register (VREFTRM)

The VREFTRM register contains eight bits that contain the trim data for the Voltage Reference as described in [Table 23-2](#).

**Register address: Base + 0x00**



**Figure 23-2. VREF Trim Register (VREFTRM)**

**Table 23-2. VREFTRM Register Field Descriptions**

Field	Description
7:0 TRM[7:0]	Trim Bits 7:0 These bits change the resulting VREF output by $\pm 0.5\text{mV}$ for each step.

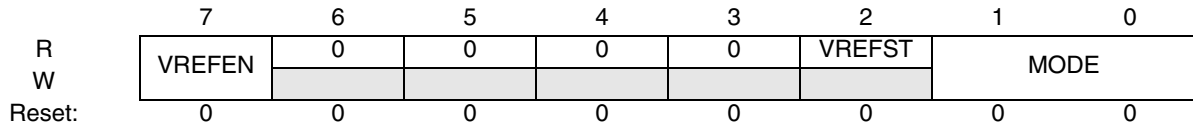
**Table 23-3. VREFTRM Register Settings**

TRM7	TRM6	TRM5	TRM4	TRM3	TRM2	TRM1	TRM0	VREF (mV)
1	0	0	0	0	0	0	0	max
1	0	0	0	0	0	0	1	max-0.5
1	0	0	0	0	0	1	0	max-1.0
1	0	0	0	0	0	1	1	max-1.5
1	.....							.....
1	1	1	1	1	1	0	0	mid+1.5
1	1	1	1	1	1	0	1	mid+1.0
1	1	1	1	1	1	1	0	mid+0.5
1	1	1	1	1	1	1	1	mid
0	0	0	0	0	0	0	0	mid-0.5
0	0	0	0	0	0	0	1	mid-1.0
0	0	0	0	0	0	1	0	mid-1.5
0	0	0	0	0	0	1	1	mid-2.0
0	.....							.....
0	1	1	1	1	1	0	0	min+1.5
0	1	1	1	1	1	0	1	min+1.0
0	1	1	1	1	1	1	0	min+0.5
0	1	1	1	1	1	1	1	min

### 23.2.2 VREF Status and Control Register (VREFSC)

The VREF status and control register contains the control bits used to enable the internal voltage reference and select the buffer mode to be used.

**Register address: Base + 0x01**



**Figure 23-3. Voltage Reference Control Register (VREFSC)**

**Table 23-4. VREFSC Register Field Descriptions**

Field	Description
7 VREFEN	<b>Internal Voltage Reference Enable</b> — This bit is used to enable the Voltage Reference module. 0 the Voltage Reference module is disabled 1 the Voltage Reference module is enabled
2 VREFST	<b>Internal Voltage Reference Stable</b> — This bit indicates that the Voltage Reference module has completed its startup and stabilization. 0 Voltage Reference module is disabled or not stable 1 Voltage Reference module is stable
1:0 MODE[1:0]	<b>Mode selection</b> — These bits are used to select the modes of operation for the Voltage Reference module. 00 Bandgap on only, for stabilization and startup 01 Low-power buffer enabled 10 Tight-regulation buffer enabled 11 RESERVED

## 23.3 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The Voltage Reference can be used in two main cases. It can be used as a reference to analog peripherals such as an ADC channel or analog comparator input. For this case, the low-power buffer can be used. When the tight-regulation buffer is enabled, VREFO can be used both internally and externally.

Table 23-5 shows all possible functional configurations of the Voltage Reference.

**Table 23-5. Voltage Reference Functional Configurations**

VREFEN	MODE[1:0]	Configuration	Functionality
0	X	Voltage Reference Disabled	Off
1	00	Voltage Reference Enabled, Bandgap on only	Startup and Standby
1	01	Voltage Reference Enabled, Low-Power buffer on	Can be used for internal peripherals only and VREFO pin should not be loaded
1	10	Voltage Reference Enabled, Tight-Regulation buffer on	Can be used both internally and externally. A 100nF capacitor is required on VREFO and 10mA max drive strength is allowed.
1	11	Voltage Reference Disabled	RESERVED



### 23.3.1 Voltage Reference Disabled, VREFEN=0

When VREFEN=0, the Voltage Reference is disabled, all bandgap and buffers are disabled. The Voltage Reference is in off mode.

### 23.3.2 Voltage Reference Enabled, VREFEN=1

When VREFEN=1, the Voltage Reference is enabled, and different modes should be set by the Mode[1:0] bits.

#### 23.3.2.1 Mode[1:0]=00

The internal bandgap is on to generate an accurate voltage output that can be trimmed by the bits TRM[7:0] in 0.5 mV steps. The bandgap requires some time for startup and stabilization. The VREFST bit can be monitored to determine if the stabilization and startup is complete.

Both low-power buffer and tight-regulation buffer are disabled in this mode, and there is no buffered voltage output. The Voltage Reference is in standby mode.

#### 23.3.2.2 Mode[1:0]=01

The internal bandgap is on.

The low-power buffer is enabled to generate a buffered internal voltage. It can be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

#### 23.3.2.3 Mode[1:0]=10

The internal bandgap is on.

The tight-regulation buffer is enabled to generate a buffered voltage to VREFO with load regulation less than 100  $\mu\text{V}/\text{mA}$ . A 100nF capacitor is required on VREFO pin and it is allowed to drive 10 mA maximum current. VREFO can be used internally and/or externally.

#### 23.3.2.4 Mode[1:0]=11

RESERVED

## 23.4 Initialization Information

The Voltage Reference requires some time for startup and stabilization. Once the VREFEN bit is set, the VREFST bit can be monitored to determine if the stabilization and startup is complete.

When the Voltage Reference is already enabled and stabilized, changing the Mode selection bits (MODE[1:0]) will not clear the VREFST bit, but there will be some startup time before the output voltage is stabilized when the low-power buffer or tight-regulation buffer is enabled, and there will be some setting time when a step change of the load current occurs.



## Chapter 24

# Development Support

### 24.1 Introduction

This chapter describes the single-wire background debug mode (BDM), which uses the on-chip background debug controller (BDC) module, and the independent on-chip real-time in-circuit emulation (ICE) system, which uses the on-chip debug (DBG) module.

#### 24.1.1 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

### 24.2 Background Debug Controller (BDC)

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

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

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

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

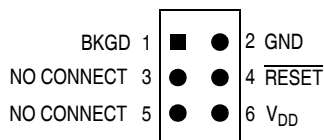


Figure 24-1. BDM Tool Connector

## 24.2.1 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 24.2.2, "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 24.2.2, "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.

## 24.2.2 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.

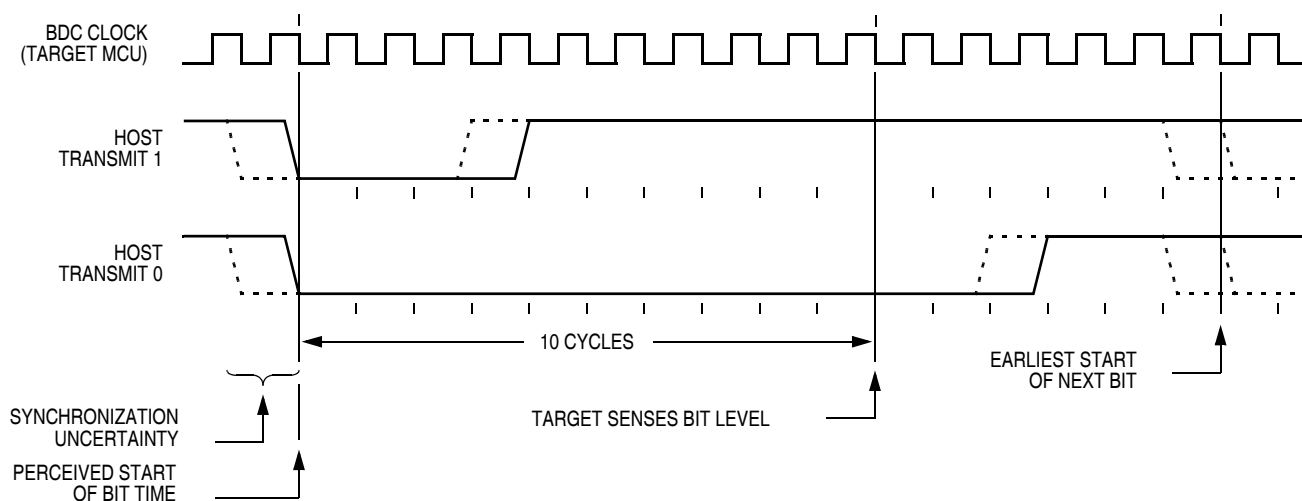
BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress 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.

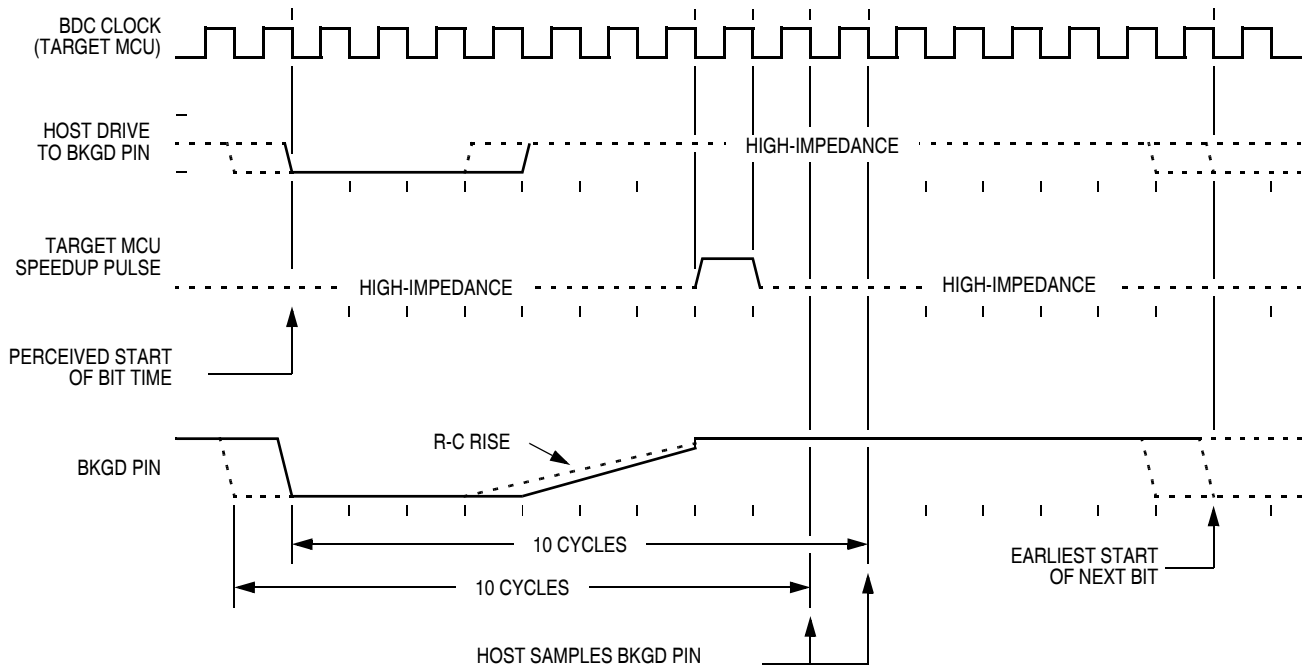
The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting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

Figure 24-2 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 24-2. BDC Host-to-Target Serial Bit Timing**

Figure 24-3 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 24-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 24-4 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.

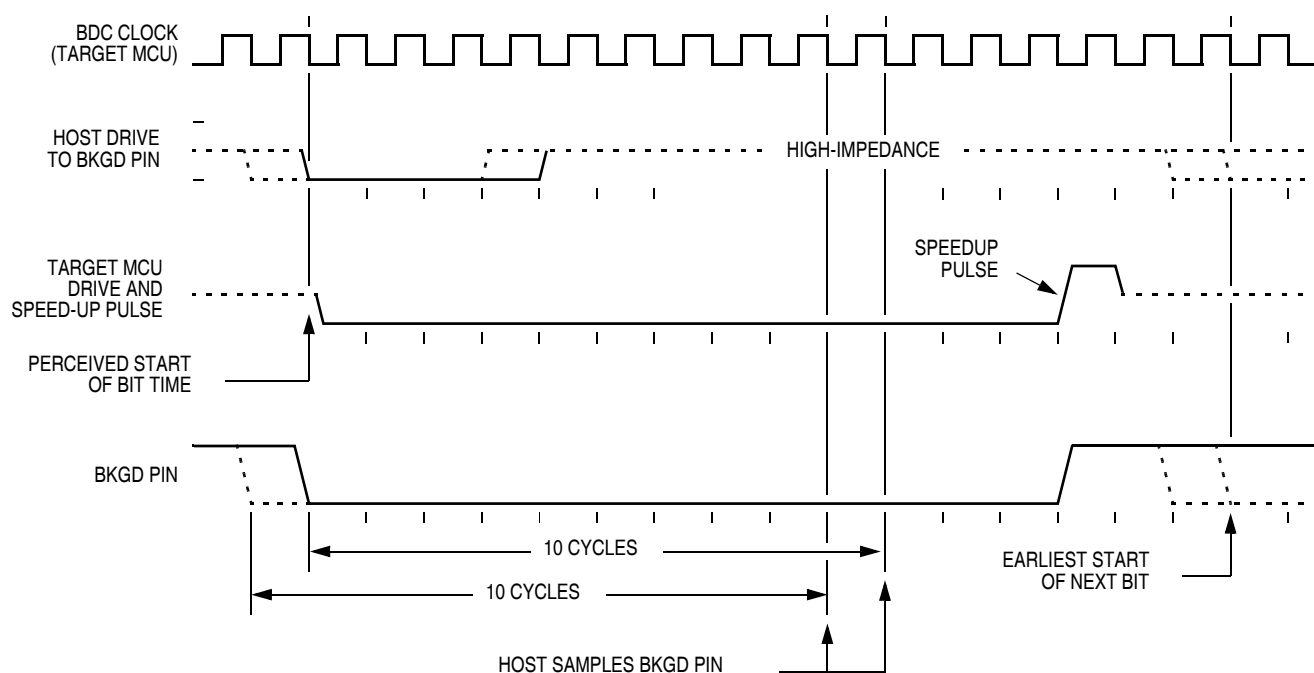


Figure 24-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

### 24.2.3 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 24-1 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 24-1 to describe the coding structure of the BDC commands.



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

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

**Table 24-1. BDC Command Summary**

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
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

Table 24-1. BDC Command Summary (continued)

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
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.

## 24.2.4 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.

## 24.3 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.

### 24.3.1 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.

### 24.3.1.1 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 24-5. BDC Status and Control Register (BDCSCR)

Table 24-2. 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

Table 24-2. BDCSCR Register Field Descriptions (continued)

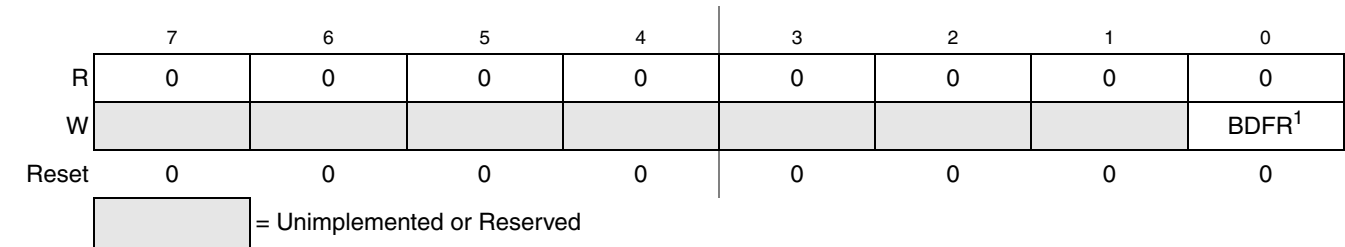
Field	Description
4 FTS	<b>Force/Tag Select</b> — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches 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
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 MC9S08GW64 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

### 24.3.1.2 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 24.2.4, “BDC Hardware Breakpoint.”](#)

### 24.3.2 System Background Debug Force Reset Register (SBD FR)

This register contains a single write-only control bit. A serial background mode command such as WRITE\_BYTE must be used to write to SBD FR. 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 24-6. System Background Debug Force Reset Register (SBD FR)

Table 24-3. SBD FR 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 25

# Break Point Unit (BKPT) (64K)

### 25.1 Introduction

The BKPT module extends breakpoint capacity intended for non-intrusive debug of application software. It implements up to 3 additional breakpoints to the single breakpoint existing in the BDC module. The BKPT module provides basic on-chip ICE (in-circuit emulation) functions that allow address bus and data bus matching with flexible activation capability. The BKPT module supports the HCS08L 8-bit architecture with memory space up to 64K or 128K. To handle 64K of memory space, BKPT can be used alone or in combination with the Memory Management Unit (MMU). To handle 128K of memory space, BKPT must be used together with MMU.

#### 25.1.1 Features

The BKPT module includes these distinctive features:

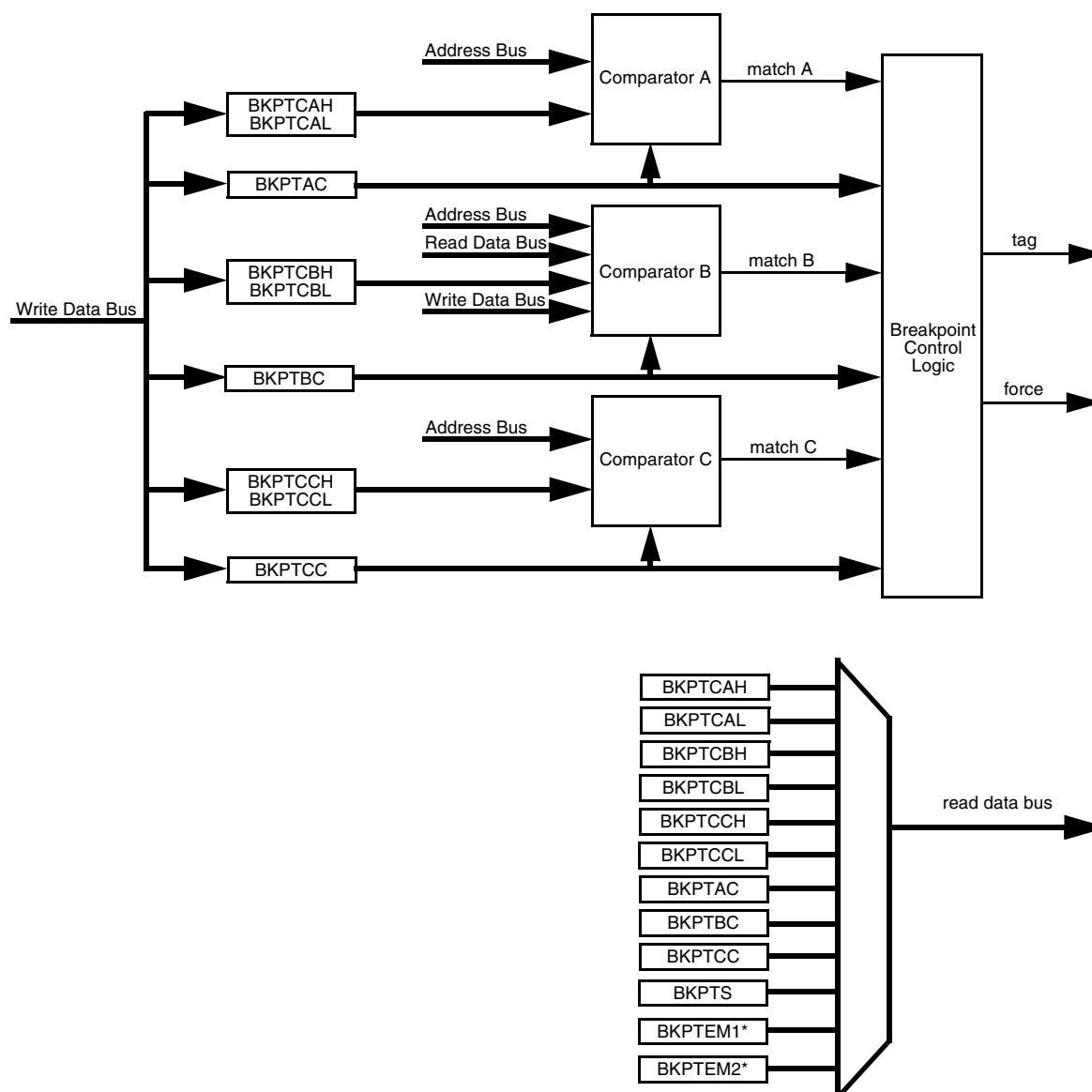
- Three comparators (A, B, and C) with ability to match addresses in 64Kspace
  - Each comparator can be used as hardware breakpoint
  - Full mode, Comparator A compares address and Comparator B compares data
- Tag and Force type breakpoints configured individually for each comparator.

#### 25.1.2 Modes of Operation

The BKPT module comparators A, B and C are disabled if the MCU is in secure mode. The BKPT module comparators A, B and C are disabled when executing a Background Debug Mode (BDM) command.

#### 25.1.3 Block Diagram

Figure 25-1 shows the block diagram of the BKPT module Block Diagram.



\* In 64K versions of this module there are no BKPTM1 and BKPTM2 registers

**Figure 25-1. BKPT Block Diagram**

## 25.2 Signal Description

The BKPT module contains no external signals.



## 25.3 Memory Map and Registers

This section provides a detailed description of all BKPT registers accessible to the end user.

### 25.3.1 Module Memory Map

Table 25-1 shows the registers contained in the BKPT module.

**Table 25-1. Module Memory Map**

Address	Use	Access
Base + \$0000	Breakpoint Comparator A High Register (BKPTCAH)	Read/write
Base + \$0001	Breakpoint Comparator A Low Register (BKPTCAL)	Read/write
Base + \$0002	Breakpoint Comparator B High Register (BKPTCBH)	Read/write
Base + \$0003	Breakpoint Comparator B Low Register (BKPTCBL)	Read/write
Base + \$0004	Breakpoint Comparator C High Register (BKPTCCH)	Read/write
Base + \$0005	Breakpoint Comparator C Low Register (BKPTCCL)	Read/write
Base + \$0006	Breakpoint Comparator A Control Register (BKPTAC)	Read/write
Base + \$0007	Breakpoint Comparator B Control Register (BKPTBC)	Read/write
Base + \$0008	Breakpoint Comparator C Control Register (BKPTCC)	Read/write
Base + \$0009	Breakpoint Status Register (BKPTS)	Read

## 25.3.2 Register Bit Summary-

Table 25-2. Register Bit Summary

	7	6	5	4	3	2	1	0
<b>BKPTCAH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>BKPTCAL</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>BKPTCBH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>BKPTCBL</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>BKPTCCH</b>	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
<b>BKPTCCL</b>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>BKPTAC</b>	BKPTAEN	RWAEN	RWA	TAGA	PAGESELA	0	FMEN	FMDC
<b>BKPTBC</b>	BKPTBEN	RWBEN	RWB	TAGB	PAGESELB	0	0	0
<b>BKPTCC</b>	BKPTCEN	RWCEN	RWC	TAGC	PAGESELC	0	0	0
<b>BKPTS</b>	0	0	0	0	0	AF	BF	CF

## 25.3.3 Register Descriptions

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

### NOTE

For all registers below, consider: U = Unchanged, bit maintains its value after reset. POR = Power-On Reset.

### 25.3.3.1 Breakpoint Comparator A High Register (BKPTCAH)

Module Base + 0x0000

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

Figure 25-2. Breakpoint Comparator A High Register (BKPTCAH)

Table 25-3. BKPTCAH 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

### 25.3.3.2 Breakpoint Comparator A Low Register (BKPTCAL)

Module Base + 0x0001

	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	1	1	1	1	1	1	1	0
Reset	U	U	U	U	U	U	U	U

Figure 25-3. Breakpoint Comparator A Low Register (BKPTCAL)

Table 25-4. BKPTCAL 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

### 25.3.3.3 Breakpoint Comparator B High Register (BKPTCBH)

Module Base + 0x0002

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

Figure 25-4. Breakpoint Comparator B High Register (BKPTCBH)

Table 25-5. BKPTCBH 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

### 25.3.3.4 Breakpoint Comparator B Low Register (BKPTCBL)

#### NOTE

BKPTCBL register is used to compare the data bus (8-bit comparison) when full mode is enabled, FMEN bit set in BKPTAC register. In full mode comparator B high, BKPTCBH, has no functionality.

Module Base + 0x0003

	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	0	0	0	0	0	0	0	0
Reset	U	U	U	U	U	U	U	U

Figure 25-5. Breakpoint Comparator B Low Register (BKPTCBL)

Table 25-6. BKPTCBL 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

### 25.3.3.5 Breakpoint Comparator C High Register (BKPTCCH)

Module Base + 0x0004

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

Figure 25-6. Breakpoint Comparator C High Register (BKPTCCH)

Table 25-7. BKPTCCH 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

### 25.3.3.6 Breakpoint Comparator C Low Register (BKPTCCL)

Module Base + 0x0005

	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	0	0	0	0	0	0	0	0
Reset	U	U	U	U	U	U	U	U

Figure 25-7. Breakpoint Comparator C Low Register (BKPTCCL)

Table 25-8. BKPTCCL 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

### 25.3.3.7 Breakpoint Comparator A Control Register (BKPTAC)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	BKPTAEN	RWAEN	RWA	TAGA	PAGESELA	0	FMEN	FMDC
W								
POR	0	0	0	0	0	0	0	0
Reset	U	U	U	U	U	0	U	U

 = Unimplemented or Reserved

Figure 25-8. Breakpoint Comparator A Control Register (BKPTAC)

Table 25-9. BKPTAC Field Descriptions

Field	Description
7 BKPTAEN	<b>BKPT A Enable Bit</b> — The BKPTAEN bit enables the comparator A to generate a breakpoint. The BKPTAEN bit is forced to zero and cannot be set if the MCU is secure. 0 Comparator A disabled to generate breakpoint 1 Comparator A enabled to generate breakpoint
6 RWAEN	<b>Read/Write Comparator A Enable Bit</b> — The RWAEN bit controls whether read or write comparison is enabled for Comparator A. In full mode, RWAEN=1 and RWA=0 select write comparisons and the write data bus. Otherwise, read comparisons and the read data bus will be selected. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
5 RWA	<b>Read/Write Comparator A Value Bit</b> — The RWA bit controls whether read or write is used for comparison in Comparator A. The RWA bit has no effect if RWAEN = 0. In full mode, RWAEN=1 and RWA=0 select write comparisons and the write data bus. Otherwise, read comparisons and the read data bus will be selected. 0 Write cycle will be matched 1 Read cycle will be matched
4 TAGA	<b>Tag or Force Bit</b> — The TAGA bit controls whether comparator A breakpoint will be requested as a tag or force breakpoint to the CPU. 0 Force breakpoint request selected 1 Tag breakpoint request selected
3 PAGESELA	<b>Comparator A Page Select Bit</b> — This PAGESELA bit controls whether Comparator A will perform an address comparison using an extended memory access through the PPAGE mechanism. 0 Address comparison is performed using the core address bits [15:0] that correspond either to a 16-bit CPU address, or a 16-bit linear address pointer 1 Address comparison is performed using the core address bits [15:0] that correspond to flash memory address made up of PPAGE[1:0]:addr[13:0]
1 FMEN	<b>Full Mode Enable Bit</b> - The FMEN bit enables the Full Mode where Comparator A compares address and Comparator B compares data. When FMEN = 1 the BKPTBC register has no effect in comparator B. 0 Full Mode not enabled 1 Full Mode enabled
0 FMDC	<b>Full Mode Data Condition</b> - The FMDC bit controls whether a breakpoint is generated when Comparator B match is equal or when Comparator B match is not equal. The FMDC bit has no effect if FMEN=0. 0 Condition (A and B) generates breakpoint 1 Condition (A and (not B)) generates breakpoint

**NOTE**

For any access in the paging window PAGESELA should be 1 for breakpoint to happen.

**25.3.3.8 Breakpoint Comparator B Control Register (BKPTBC)****NOTE**

BKPTBC does not control the comparator B if the Full mode is enabled, FMEN bit is set in BKPTAC register.

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	BKPTBEN	RWBEN	RWB	TAGB	PAGESELB	0	0	0
W								
POR	0	0	0	0	0	0	0	0
Reset	U	U	U	U	U	0	0	0


 = Unimplemented or Reserved

Figure 25-9. Breakpoint Comparator B Control Register (BKPTBC)

Table 25-10. BKPTBC Field Descriptions

Field	Description
7 BKPTBEN	<b>BKPT B Enable Bit</b> — The BKPTBEN bit enables the comparator B to generate a breakpoint. In full modes, BKPTAEN is used to enable the comparator B and BKPTBEN is ignored. The BKPTBEN bit is forced to zero and cannot be set if the MCU is secure. 0 Comparator B disabled to generate breakpoints 1 Comparator B enabled to generate breakpoints
6 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
5 RWB	<b>Read/Write Comparator B Value Bit</b> — The RWB bit controls whether read or write is used for comparison in Comparator B. The RWB bit has no effect if RWBEN = 0. In full mode, 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
4 TAGB	<b>Tag or Force Bit</b> — The TAGB bit controls whether comparator B breakpoint will be requested as a tag or force breakpoint to the CPU. In full modes, TAGA is used to control tag or force requests and TAGB is ignored. 0 Force breakpoint request selected 1 Tag breakpoint request selected
3 PAGESELB	<b>Comparator B Page Select Bit</b> — This PAGESELB bit controls whether Comparator B will perform an address comparison using an extended memory access through the PPAGE mechanism. 0 Address comparison is performed using the core address bits [15:0] that correspond either to a 16-bit CPU address, or a 16-bit linear address pointer 1 Address comparison is performed using the core address bits [15:0] that correspond to flash memory address made up of PPAGE[1:0]:addr[13:0]

**NOTE**

For any access in the paging window PAGESELB should be 1 for breakpoint to happen.



### 25.3.3.9 Breakpoint Comparator C Control Register (BKPTCC)

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R	BKPTCEN	RWCEN	RWC	TAGC	PAGESELC	0	0	0
W								
POR	0	0	0	0	0	0	0	0
Reset	U	U	U	U	U	0	0	0

= Unimplemented or Reserved

Figure 25-10. Breakpoint Comparator C Control Register (BKPTCC)

Table 25-11. BKPTCC Field Descriptions

Field	Description
7 BKPTCEN	<b>BKPT C Enable Bit</b> — The BKPTCEN bit enables the comparator C to generate a breakpoint. The BKPTCEN bit is forced to zero and cannot be set if the MCU is secure. 0 Comparator C disable to generate breakpoints 1 Comparator C enabled to generate breakpoints
6 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
5 RWC	<b>Read/Write Comparator C Value Bit</b> — The RWC bit controls whether read or write is used for comparison in Comparator C. The RWC bit has no effect if RWCEN = 0. 0 Write cycle will be matched 1 Read cycle will be matched
4 TAGC	<b>Tag or Force Bit</b> — The TAGC bit controls whether comparator C breakpoint will be requested as a tag or force breakpoint to the CPU. 0 Force breakpoint request selected 1 Tag breakpoint request selected
3 PAGESELC	<b>Comparator C Page Select Bit</b> — This PAGESELC bit controls whether Comparator C will perform an address comparison using an extended memory access through the PPAGE mechanism. 0 Address comparison is performed using the core address bits [15:0] that correspond either to a 16-bit CPU address, or a 16-bit linear address pointer 1 Address comparison is performed using the core address bits [15:0] that correspond to flash memory address made up of PPAGE[1:0]:addr[13:0]

#### NOTE

For any access in the paging window PAGESELC should be 1 for breakpoint to happen.

### 25.3.3.10 Breakpoint Status Register (BKPTS)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	AF	BF	CF
W								
POR	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	U	U	U


 = Unimplemented or Reserved

Figure 25-11. Breakpoint Status Register (BKPTS)

Table 25-12. BKPTS Field Descriptions

Field	Description
2 AF	<b>Comparator A Match Bit</b> — The AF bit is cleared when the core goes back to foreground mode. In full mode, AF is set only if the condition “A And B” or “A And Not B” is met, see <a href="#">Section 25.4.2.2.1</a> and <a href="#">Section 25.4.2.2.2</a> for more information. When not in full mode, the AF bit indicates if Comparator A match condition was met. 0 Comparator A did not match 1 Comparator A match
1 BF	<b>Comparator B Match Bit</b> — The BF bit is cleared when the core goes back to foreground mode. In full mode, BF is set only if the condition “A And B” or “A And Not B” is met, see <a href="#">Section 25.4.2.2.1</a> and <a href="#">Section 25.4.2.2.2</a> for more information. When not in full mode, the BF bit indicates if Comparator B match condition was met. 0 Comparator B did not match 1 Comparator B match
0 CF	<b>Comparator C Match Bit</b> — The CF bit is cleared when the core goes back to foreground mode. The CF bit indicates if Comparator C match condition was met . 0 Comparator C did not match 1 Comparator C match

#### NOTE

Bits AF, BF and CF do not reflect whether a tagged instruction is executed or not.

## 25.4 Functional Description

This section provides a complete functional description of the Breakpoint Unit. The BKPT module can set up to 3 breakpoints. Each breakpoint is enabled individually by the respective BKPTxEN bit in the BKPTxC register. The BKPT module is made up of two main blocks, the Comparators and Breakpoint Logic Control.

### 25.4.1 Comparator

The BKPT module contains three comparators, A, B, and C. Comparator A compares the core address bus with the address stored in the BKPTCAH and BKPTCAL registers. Comparator B compares the core address bus with the address stored in the BKPTCBH and BKPTCBL registers, except in full mode with force-type breakpoint, where it compares the data bus to the data stored in the BKPTCBL register. Comparator C compares the core address bus with the address stored in the BKPTCCH and BKPTCCL registers. Matches on Comparators A, B, and C are signaled to the Breakpoint Control Logic block.

#### 25.4.1.1 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.

### 25.4.2 Breakpoint Control Logic (BCL)

The BCL is the logic controller for the BKPT module. The outputs of the comparator block are sent to the breakpoint control logic where the BKPTxEN bit (in BKPTxC register) determines whether a CPU break is requested. The TAGx control bit (in BKPTxC register) determines whether the CPU break will be a tag-type or force-type breakpoint.

#### 25.4.2.1 Breakpoint types

A breakpoint request to the CPU can be created by each comparator if the respective BKPTxEN bit in the BKPTxC register is set.

There are two types of breakpoint requests supported by the BKPT 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 force breakpoint request remains active until the CPU enters in BDM, and is not cleared by resets, except POR. The TAGx bit in the BKPTxC register determines whether CPU breakpoint requests will be a tag-type or force-type breakpoints. When TAGx = 0, a force-type breakpoint is requested and it will take effect at the next instruction boundary after the request. When TAGx = 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. When setting a tag breakpoint, the status bits AF, BF and CF in register BKPTS should be ignored since they are meaningless.

### 25.4.2.2 Full Mode

The BCL is also responsible for the Full Mode comparison functionality. The full mode allows the generation of a breakpoint to the CPU upon an address and data match. Note that comparator B performs a data comparison using the data contained in the address that comparator A uses to perform an address comparison. Full mode is enabled by the FMEN bit whereas the condition that generates a breakpoint in full mode is determined by the FMDC bit. Both bits are defined in the BKPTAC register. The following sections explain the full mode functionality.

#### 25.4.2.2.1 A And B (Full Mode)

In the A And B full mode, Comparator A compares to the address bus and Comparator B compares to the data bus. In the A And B full mode, if the match condition for A and B happens, both the AF and BF flags in the BKPTS register are set. If a match condition on only A or only B happens, no flags are set. For Tag Breakpoint, only matches from Comparator A will be used to determine if the Breakpoint conditions are met and Comparator B matches will be ignored. For Force Breakpoint, both matches from Comparator A and Comparator B will be used together to determine if the Breakpoint conditions are met.

#### 25.4.2.2.2 A And Not B (Full Mode)

In the A And Not B full mode, comparator A compares to the address bus and comparator B compares to the data bus. In the A And Not B full mode, if the match condition for A and Not B happen, both the AF and BF flags in the BKPTS register are set. If a match condition on only A or only Not B occurs, no flags are set. For Tag Breakpoint, only matches from Comparator A will be used to determine if the Breakpoint conditions are met and Comparator B matches will be ignored. For Force Breakpoint, both matches from Comparator A and Comparator B will be used together to determine if the Breakpoint conditions are met.

## 25.5 Resets

The BKPT module cannot cause an MCU reset.

## 25.6 Interrupts

The BKPT contains no interrupt source.

## ***How to Reach Us:***

### **Home Page:**

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

### **Web Support:**

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

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

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

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### **Japan:**

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

### **Asia/Pacific:**

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

### **For Literature Requests Only:**

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

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

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



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. 2009-2010. All rights reserved.