

# MC13211/212/213

ZigBee™ - Compliant Platform  
2.4 GHz Low Power Transceiver for the  
IEEE® 802.15.4 Standard plus Microcontroller  
Reference Manual

Document Number: MC1321xRM  
Rev. 1.6  
05/2010

**How to Reach Us:****Home Page:**

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

**E-mail:**

[support@freescale.com](mailto:support@freescale.com)

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

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
+1-800-521-6274 or +1-480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

**Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[support@freescale.com](mailto:support@freescale.com)

**Japan:**

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

**Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

**For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-521-6274 or 303-675-2140  
Fax: 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. 2005, 2006, 2007, 2008, 2009, 2010. All rights reserved.

# Contents

---

## About This Book

Audience . . . . .	xv
Organization . . . . .	xv
Revision History . . . . .	xvi
References . . . . .	xviii

## Chapter 1 MC1321x Introduction

1.1	Ordering Information . . . . .	1-3
1.2	General Platform Features . . . . .	1-3
1.3	Microcontroller Features. . . . .	1-4
1.4	RF Modem Features . . . . .	1-5
1.5	Software Solutions . . . . .	1-5
1.5.1	Simple Media Access Controller (SMAC) . . . . .	1-5
1.5.2	IEEE 802.15.4 2006 Standard-Compliant MAC . . . . .	1-5
1.5.3	SynkroRF Platform . . . . .	1-6
1.5.4	BeeStack Consumer . . . . .	1-6
1.5.5	ZigBee-Compliant Network Stack . . . . .	1-7
1.6	System Block Diagram . . . . .	1-8
1.7	802.15.4 Modem Overview . . . . .	1-9
1.7.1	Modem Block Diagram . . . . .	1-9
1.7.2	Modem Data Transfer Modes . . . . .	1-9
1.7.3	Modem Packet Structure. . . . .	1-10
1.7.4	Modem Receive Path Description . . . . .	1-10
1.7.5	Modem Transmit Path Description. . . . .	1-11
1.7.6	Radio Usage . . . . .	1-12
1.8	MCU Overview. . . . .	1-12
1.8.1	MCU Block Diagram . . . . .	1-13
1.8.2	HCS08 Central Processing Unit (CPU) . . . . .	1-14
1.8.3	MCU Peripheral Modules. . . . .	1-14
1.8.4	MCU Internal Clock Distribution. . . . .	1-15
1.9	System Clock Configuration. . . . .	1-16

## Chapter 2 MC1321x Pins and Connections

2.1	Device Pin Assignment. . . . .	2-1
2.2	Pin Definitions . . . . .	2-2

## Chapter 3 System Considerations

3.1	Introduction. . . . .	3-1
3.2	Power Connections . . . . .	3-1

3.3	Test Pin SM	3-3
3.4	Internal Functional Interconnects and System Reset	3-3
3.4.1	System Reset	3-4
3.4.2	Modem Interrupt Request to MCU	3-5
3.4.3	SPI Command Channel	3-5
3.4.4	Modem Control Signals	3-6
3.4.5	Modem Status Signals	3-6
3.5	Clock Sources	3-6
3.5.1	Modem Oscillator	3-7
3.5.2	Modem CLKO Clock Source Output	3-7
3.5.3	MCU Clock Sources	3-8
3.5.4	System Clock Configurations	3-9
3.6	MC1321x Transceiver Initialization	3-11
3.7	MCU Background/Mode Select (PTG0/BKGD/MS)	3-11
3.8	MC1321x GPIO (Mixed I/O from Modem and MCU)	3-12
3.8.1	MCU GPIO Characteristics	3-12
3.8.2	Modem GPIO Characteristics	3-14
3.9	MC1321x Digital Signal Properties Summary	3-15
3.9.1	Signal Properties Summary	3-15
3.10	Transceiver RF Configurations and External Connections	3-18
3.10.1	RF Interface Pins	3-18
3.10.2	Controlling RF Modes of Operation	3-21
3.10.3	RF Control Output CT_Bias	3-22
3.11	MC1321x Timer Resources	3-23
3.11.1	MCU TPM Modules	3-23
3.11.2	Modem Event Timer Block with Four Timer Compare Registers	3-24
3.11.3	MCU Real-Time Interrupt (RTI)	3-24
3.11.4	MCU Computer Operating Properly (COP) Watchdog	3-25
3.12	Low Power Considerations	3-25
3.12.1	Modem Low Power States	3-25
3.12.2	Special Consideration Where Modem Doze Current Is Higher Than Specified	3-26
3.12.3	Modem Low Power Exit Using ATTN	3-27
3.12.4	MCU Low Power States	3-28
3.12.5	Recovery Times from Low Power Modes	3-29
3.12.6	Run Time Current	3-31
3.12.7	Configuration of Interconnected GPIO for Low Power Operation	3-34
3.12.8	General System Considerations for Low Power	3-35

## Chapter 4

### MC1321x Serial Peripheral Interface (SPI)

4.1	SiP Level SPI Pin Connections	4-1
4.2	Features	4-2
4.3	SPI System Block Diagram	4-2
4.4	Modem SPI Overview	4-3

4.5	Modem SPI Basic Operation	4-3
4.5.1	Modem SPI Pin Definition	4-3
4.5.2	Modem SPI Byte Burst Operation	4-5
4.6	MCU SPI Block Diagrams	4-6
4.6.1	MCU SPI Module Block Diagram	4-6
4.6.2	MCU SPI Baud Rate Generation	4-7
4.7	MCU SPI Functional Description	4-7
4.7.1	SPI Clock Formats	4-8
4.7.2	MCU SPI Pin Controls	4-9
4.7.3	MCU SPI Interrupts	4-9
4.7.4	Mode Fault Detection	4-10
4.8	MCU SPI Registers and Control Bits	4-10
4.8.1	SPI Control Register 1 (SPI1C1)	4-11
4.8.2	SPI Control Register 2 (SPI1C2)	4-12
4.8.3	SPI Baud Rate Register (SPI1BR)	4-13
4.8.4	SPI Status Register (SPI1S)	4-14
4.8.5	SPI Data Register (SPI1D)	4-15
4.9	Modem SPI Singular Transactions	4-15
4.9.1	SPI Singular Transaction Signalling	4-15
4.9.2	SPI Singular Transaction Protocol	4-16
4.10	Modem Symbol/Data Format	4-17
4.11	Modem SPI Recursive Transactions	4-18
4.11.1	Recursive SPI Register Read	4-18
4.11.2	Recursive SPI Register Write	4-18
4.11.3	Special Case - Packet RAM Access	4-19
4.12	Modem Program Reset (Writing Address 0x00)	4-22
4.13	Configuring MCU Registers for Proper SPI Operation	4-22
4.13.1	Set SPI Module Mode	4-22
4.13.2	SPI Baud Rate Control	4-23

## Chapter 5 Modem SPI Register Descriptions

5.1	Overview	5-1
5.2	Register Model and Description Details	5-2
5.3	Reset - Register 00	5-5
5.4	RX_Pkt_RAM - Register 01	5-5
5.5	TX_Pkt_RAM - Register 02	5-6
5.6	TX_Pkt_Ctl - Register 03	5-7
5.7	CCA_Thresh - Register 04	5-7
5.8	IRQ_Mask - Register 05	5-8
5.9	Control_A - Register 06	5-10
5.10	Control_B - Register 07	5-11
5.11	PA_Enable - Register 08	5-13
5.12	Control_C - Register 09	5-14

5.13	CLKO_Ctl - Register 0A	5-15
5.14	GPIO_Dir - Register 0B	5-16
5.15	GPIO_Data_Out - Register 0C	5-17
5.16	LO1_Int_Div - Register 0F	5-18
5.17	LO1_Num - Register 10	5-19
5.18	PA_Lvl - Register 12	5-20
5.19	Tmr_Cmp1_A - Register 1B	5-20
5.20	Tmr_Cmp1_B - Register 1C	5-21
5.21	Tmr_Cmp2_A - Register 1D	5-22
5.22	Tmr_Cmp2_B - Register 1E	5-23
5.23	Tmr_Cmp3_A - Register 1F	5-23
5.24	Tmr_Cmp3_B - Register 20	5-24
5.25	Tmr_Cmp4_A - Register 21	5-24
5.26	Tmr_Cmp4_B - Register 22	5-25
5.27	TC2_Prime - Register 23	5-26
5.28	IRQ_Status - Register 24	5-26
5.29	RST_Ind - Register 25	5-29
5.30	Current_Time_A - Register 26	5-30
5.31	Current_Time_B - Register 27	5-30
5.32	GPIO_Data_In - Register 28	5-31
5.33	Chip_ID - Register 2C	5-32
5.34	RX_Status - Register 2D	5-32
5.35	Timestamp_A - Register 2E	5-33
5.36	Timestamp_B - Register 2F	5-34
5.37	BER_Enable - Register 30	5-34
5.38	PSM_Mode - Register 31	5-35
5.39	Reserved - Register 34	5-36

## Chapter 6 Modem Timer Information

6.1	Event Timer Block	6-1
6.2	Event Timer Time Base	6-2
6.3	Setting Current Time	6-2
6.4	Reading Current Time	6-3
6.5	Latching the Timestamp	6-3
6.6	Event Timer Comparators	6-3
6.6.1	Timer Compare Fields	6-3
6.6.2	Timer Disable Bits	6-4
6.6.3	Timer Status Flags	6-4
6.6.4	Timer Interrupt Masks	6-4
6.6.5	Setting Compare Values	6-4
6.7	Intended Event Timer Usage	6-5
6.7.1	Generating Time-Based Interrupts	6-5
6.7.2	Using tmr_cmp2[23:0] to Exit Doze Mode	6-6

6.7.3	Timer-Triggered Transceiver Events .....	6-6
-------	--	-----

## Chapter 7 Modem Modes of Operation

7.1	Modem Operational Modes Summary .....	7-1
7.2	Low Power Modes .....	7-4
7.2.1	Off Mode .....	7-4
7.2.2	Hibernate Mode .....	7-4
7.2.3	Doze Mode .....	7-4
7.3	Active Modes .....	7-5
7.3.1	Idle Mode .....	7-5
7.3.2	Controlling Transition to Other Active Modes from Idle .....	7-6
7.3.3	Packet Mode Data Transfer TX and RX Operation .....	7-6
7.3.4	Stream Mode Data Transfer TX and RX Operation .....	7-9
7.3.5	Clear Channel Assessment (CCA) Modes (including Link Quality Indication) .....	7-13
7.4	Frequency of Operation .....	7-18
7.5	Transmit Power Adjustment .....	7-19
7.6	2.4GHz PLL Out-of-Lock Interrupt .....	7-20

## Chapter 8 Modem Interrupt Description

8.1	Modem Interrupts .....	8-1
8.1.1	Modem Interrupt Sources .....	8-1
8.1.2	Output Pin IRQ .....	8-3
8.2	PLL_lock_irq Status Bit and Operation .....	8-3
8.3	Attn_irq Status Bit and Interrupt Operation .....	8-4
8.4	Interrupts from Exiting Low Power Modes .....	8-4
8.4.1	Exiting Off Mode (Reset) .....	8-4
8.4.2	Exiting Hibernate Mode .....	8-4
8.4.3	Exiting Doze Mode(s) .....	8-5

## Chapter 9 Modem Miscellaneous Functions

9.1	Reset Function .....	9-1
9.1.1	Input Pin M_RSTB .....	9-1
9.1.2	Software Reset (Writing to Register 00) .....	9-1
9.1.3	Reset Indicator Bit (RST_Ind Register 25, Bit 7) .....	9-1
9.2	General Purpose Input/Output .....	9-2
9.2.1	Configuring GPIO Direction .....	9-2
9.2.2	Setting GPIO Output Drive Strength .....	9-2
9.2.3	Programming GPIO Output Value .....	9-2
9.2.4	Reading GPIO Input State .....	9-2
9.2.5	GPIO1 and GPIO2 as Stream Mode Status Indicators .....	9-3

9.2.6	GPIO in Off, Hibernate, and Doze Modes . . . . .	9-3
9.3	Crystal Oscillator . . . . .	9-3
9.3.1	Crystal Requirements . . . . .	9-4
9.3.2	Crystal Trim Operation . . . . .	9-4
9.4	Output Clock Pin CLKO. . . . .	9-5
9.4.1	Enable CLKO (clko_en, Control_C Register 09, Bit 5) . . . . .	9-5
9.4.2	Setting CLKO frequency (clko_rate[2:0], CLKO_Ctl Register 0A, Bits 2-0) . . . . .	9-5
9.4.3	Enable CLKO During Doze Mode (clko_doze_en, Control_B Register 07, Bit 9) . . . . .	9-6
9.4.4	Setting CLKO Output Drive Strength (clko_drv[1:0], GPIO_Data_Out Register 0C, Bits 11-10) . . . . .	9-6
9.5	Input Pin ATTN . . . . .	9-6

## Chapter 10 MCU Modes of Operation

10.1	Introduction. . . . .	10-1
10.2	Features. . . . .	10-1
10.3	Run Mode . . . . .	10-1
10.4	Active Background Mode. . . . .	10-1
10.5	Wait Mode . . . . .	10-2
10.6	Stop Modes . . . . .	10-3
10.6.1	Stop1 Mode. . . . .	10-3
10.6.2	Stop2 Mode. . . . .	10-4
10.6.3	Stop3 Mode. . . . .	10-4
10.6.4	Active BDM Enabled in Stop Mode. . . . .	10-5
10.6.5	LVD Enabled in Stop Mode . . . . .	10-5
10.6.6	On-Chip Peripheral Modules in Stop Modes . . . . .	10-6

## Chapter 11 MCU Memory

11.1	HCS08 Memory Map . . . . .	11-1
11.1.1	Reset and Interrupt Vector Assignments . . . . .	11-2
11.2	Register Addresses and Bit Assignments . . . . .	11-3
11.3	RAM . . . . .	11-8
11.4	FLASH . . . . .	11-9
11.4.1	Features. . . . .	11-9
11.4.2	Program and Erase Times . . . . .	11-9
11.4.3	Program and Erase Command Execution . . . . .	11-10
11.4.4	Burst Program Execution . . . . .	11-11
11.4.5	Access Errors . . . . .	11-12
11.4.6	FLASH Block Protection . . . . .	11-13
11.4.7	Vector Redirection . . . . .	11-14
11.5	Security . . . . .	11-14
11.6	FLASH Registers and Control Bits . . . . .	11-15
11.6.1	FLASH Clock Divider Register (FCDIV) . . . . .	11-15



11.6.2	FLASH Options Register (FOPT and NVOPT) . . . . .	11-17
11.6.3	FLASH Configuration Register (FCNFG) . . . . .	11-18
11.6.4	FLASH Protection Register (FPROT and NVPROT) . . . . .	11-18
11.6.5	FLASH Status Register (FSTAT) . . . . .	11-19
11.6.6	FLASH Command Register (FCMD) . . . . .	11-20

## Chapter 12 MCU Resets, Interrupts, and System Configuration

12.1	Introduction . . . . .	12-1
12.2	Features . . . . .	12-1
12.3	MCU Reset . . . . .	12-2
12.4	Computer Operating Properly (COP) Watchdog . . . . .	12-2
12.5	Interrupts . . . . .	12-3
12.5.1	Interrupt Stack Frame . . . . .	12-4
12.5.2	External Interrupt Request (IRQ) Pin . . . . .	12-4
12.6	Low-Voltage Detect (LVD) System . . . . .	12-6
12.6.1	Power-On Reset Operation . . . . .	12-7
12.6.2	LVD Reset Operation . . . . .	12-7
12.6.3	LVD Interrupt Operation . . . . .	12-7
12.6.4	Low-Voltage Warning (LVW) . . . . .	12-7
12.7	Real-Time Interrupt (RTI) . . . . .	12-7
12.7.1	Interrupt Pin Request Status and Control Register (IRQSC) . . . . .	12-8
12.7.2	System Reset Status Register (SRS) . . . . .	12-9
12.7.3	System Background Debug Force Reset Register (SBDFR) . . . . .	12-11
12.7.4	System Options Register (SOPT) . . . . .	12-11
12.7.5	System Device Identification Register (SDIDH, SDIDL) . . . . .	12-12
12.7.6	System Real-Time Interrupt Status and Control Register (SRTISC) . . . . .	12-13
12.7.7	System Power Management Status and Control 1 Register (SPMSC1) . . . . .	12-14
12.7.8	System Power Management Status and Control 2 Register (SPMSC2) . . . . .	12-15

## Chapter 13 MCU Parallel Input/Output

13.1	Introduction . . . . .	13-1
13.2	Features . . . . .	13-3
13.3	Pin Descriptions . . . . .	13-3
13.3.1	Port A and Keyboard Interrupts . . . . .	13-3
13.3.2	Port B and Analog to Digital Converter Inputs . . . . .	13-4
13.3.3	Port C and SCI2, IIC, and High-Current Drivers . . . . .	13-4
13.3.4	Port D, TPM1 and TPM2 . . . . .	13-5
13.3.5	Port E, SCI1, and SPI . . . . .	13-5
13.3.6	Port F and High-Current Drivers . . . . .	13-6
13.3.7	Port G, BKGD/MS, and Oscillator . . . . .	13-6
13.4	Parallel I/O Controls . . . . .	13-7
13.4.1	Data Direction Control . . . . .	13-7

13.4.2	Internal Pullup Control . . . . .	13-7
13.4.3	Slew Rate Control . . . . .	13-7
13.5	Stop Modes . . . . .	13-8
13.6	Parallel I/O Registers and Control Bits . . . . .	13-8
13.6.1	Port A Registers (PTAD, PTAPE, PTASE, and PTADD) . . . . .	13-8
13.6.2	Port B Registers (PTBD, PTBPE, PTBSE, and PTBDD) . . . . .	13-10
13.6.3	Port C Registers (PTCD, PTCPE, PTCSE, and PTCDD) . . . . .	13-11
13.6.4	Port D Registers (PTDD, PTDPE, PTDSE, and PTDDD) . . . . .	13-13
13.6.5	Port E Registers (PTED, PTEPE, PTESE, and PTEDD) . . . . .	13-14
13.6.6	Port F Registers (PTFD, PTFPE, PTFSE, and PTFDD) . . . . .	13-16
13.6.7	Port G Registers (PTGD, PTGPE, PTGSE, and PTGDD) . . . . .	13-17

## Chapter 14 MCU Internal Clock Generator (ICG)

14.1	Introduction . . . . .	14-1
14.1.1	Features . . . . .	14-2
14.1.2	Modes of Operation . . . . .	14-3
14.2	Oscillator Pins . . . . .	14-3
14.2.1	EXTAL— External Reference Clock / Oscillator Input . . . . .	14-3
14.2.2	XTAL— Oscillator Output . . . . .	14-4
14.2.3	External Clock Connections . . . . .	14-4
14.2.4	External Crystal/Resonator Connections . . . . .	14-4
14.3	Functional Description . . . . .	14-5
14.3.1	Off Mode (Off) . . . . .	14-5
14.3.2	Self-Clocked Mode (SCM) . . . . .	14-5
14.3.3	FLL Engaged, Internal Clock (FEI) Mode . . . . .	14-7
14.3.4	FLL Bypassed, External Clock (FBE) Mode . . . . .	14-7
14.3.5	FLL Engaged, External Clock (FEE) Mode . . . . .	14-8
14.3.6	FLL Lock and Loss-of-Lock Detection . . . . .	14-8
14.3.7	FLL Loss-of-Clock Detection . . . . .	14-9
14.3.8	Clock Mode Requirements . . . . .	14-10
14.3.9	Fixed Frequency Clock . . . . .	14-11
14.3.10	High Gain Oscillator . . . . .	14-11
14.4	Initialization/Application Information . . . . .	14-12
14.4.1	Introduction . . . . .	14-12
14.4.2	Example #1: External Crystal = 32 kHz, Bus Frequency = 4.19 MHz . . . . .	14-13
14.4.3	Example #2: External Crystal = 4 MHz, Bus Frequency = 20 MHz . . . . .	14-15
14.4.4	Example #3: No External Crystal Connection, 5.4 MHz Bus Frequency . . . . .	14-16
14.4.5	Example #4: Internal Clock Generator Trim . . . . .	14-18
14.5	ICG Registers and Control Bits . . . . .	14-19
14.5.1	Control Register 1 (C1) . . . . .	14-19
14.5.2	Control Register 2 (C2) . . . . .	14-20
14.5.3	Status Register 1 (S1) . . . . .	14-21
14.5.4	Status Register 2 (S2) . . . . .	14-22

14.5.5	Filter Registers (FLTU, ICGFLTL) .....	14-23
14.5.6	Trim Register (TRM) .....	14-24

## Chapter 15 MCU Central Processor Unit (CPU)

15.1	Introduction .....	15-1
15.2	Features .....	15-1
15.3	Programmer's Model and CPU Registers .....	15-2
15.3.1	Accumulator (A) .....	15-2
15.3.2	Index Register (H:X) .....	15-2
15.3.3	Stack Pointer (SP) .....	15-3
15.3.4	Program Counter (PC) .....	15-3
15.3.5	Condition Code Register (CCR) .....	15-3
15.4	Addressing Modes .....	15-4
15.4.1	Inherent Addressing Mode (INH) .....	15-5
15.4.2	Relative Addressing Mode (REL) .....	15-5
15.4.3	Immediate Addressing Mode (IMM) .....	15-5
15.4.4	Direct Addressing Mode (DIR) .....	15-5
15.4.5	Extended Addressing Mode (EXT) .....	15-5
15.4.6	Indexed Addressing Mode .....	15-5
15.5	Special Operations .....	15-6
15.5.1	Reset Sequence .....	15-7
15.5.2	Interrupt Sequence .....	15-7
15.5.3	Wait Mode Operation .....	15-8
15.5.4	Stop Mode Operation .....	15-8
15.5.5	BGND Instruction .....	15-8
15.6	HCS08 Instruction Set Summary .....	15-9

## Chapter 16 MCU Keyboard Interrupt (KBI)

16.1	Introduction .....	16-1
16.1.1	KBI Block Diagram .....	16-1
16.2	Register Definitions .....	16-1
16.2.1	KBI Status and Control Register (KBISC) .....	16-2
16.2.2	KBI Pin Enable Register (KBIPE) .....	16-3
16.3	Functional Description .....	16-3
16.3.1	Pin Enables .....	16-3
16.3.2	Edge and Level Sensitivity .....	16-3
16.3.3	KBI Interrupt Controls .....	16-4

## Chapter 17 MCU Timer/PWM (TPM Module)

17.1	Introduction .....	17-1
------	--------------------	------

17.2	TPM Block Diagram	17-1
17.3	Pin Descriptions	17-3
17.3.1	External TPM Clock Sources	17-3
17.3.2	TPM1CHn — TPM1 Channel n I/O Pins	17-3
17.4	Functional Description	17-3
17.4.1	Counter	17-4
17.4.2	Channel Mode Selection	17-5
17.4.3	Center-Aligned PWM Mode	17-6
17.5	TPM Interrupts	17-7
17.5.1	Clearing Timer Interrupt Flags	17-8
17.5.2	Timer Overflow Interrupt Description	17-8
17.5.3	Channel Event Interrupt Description	17-8
17.5.4	PWM End-of-Duty-Cycle Events	17-8
17.6	TPM Registers and Control Bits	17-9
17.6.1	Timer x Status and Control Register (TPM1SC)	17-9
17.6.2	Timer x Counter Registers (TPM1CNTH:TPM1CNTL)	17-11
17.6.3	Timer x Counter Modulo Registers (TPM1MODH:TPM1MODL)	17-12
17.6.4	Timer x Channel n Status and Control Register (TPM1CnSC)	17-12
17.6.5	Timer x Channel Value Registers (TPM1CnVH:TPM1CnVL)	17-14

## Chapter 18 MCU Serial Communications Interface (SCI)

18.1	Features	18-1
18.2	SCI System Description	18-1
18.3	Baud Rate Generation	18-1
18.4	Transmitter Functional Description	18-2
18.4.1	Transmitter Block Diagram	18-2
18.4.2	Send Break and Queued Idle	18-3
18.5	Receiver Functional Description	18-3
18.5.1	Receiver Block Diagram	18-4
18.5.2	Data Sampling Technique	18-5
18.5.3	Receiver Wakeup Operation	18-6
18.6	Interrupts and Status Flags	18-6
18.7	Additional SCI Functions	18-7
18.7.1	8-Bit and 9-Bit Data Modes	18-7
18.8	Stop Mode Operation	18-8
18.8.1	Loop Mode	18-8
18.8.2	Single-Wire Operation	18-8
18.9	SCI Registers and Control Bits	18-8
18.9.1	SCI1 Baud Rate Registers (SCIxBDH, SCIxBDL)	18-9
18.9.2	SCI1 Control Register 1 (SCIxC1)	18-10
18.9.3	SCI1 Control Register 2 (SCIxC2)	18-11
18.9.4	SCI1 Status Register 1 (SCIxS1)	18-13
18.9.5	SCI1 Status Register 2 (SCIxS2)	18-15

18.9.6	SCI1 Control Register 3 (SCIxC3)	18-15
18.9.7	SCI1 Data Register (SCIxD)	18-16

## Chapter 19 Inter-Integrated Circuit (IIC)

19.1	Introduction	19-1
19.1.1	Features	19-1
19.1.2	Modes of Operation	19-1
19.1.3	Block Diagram	19-2
19.2	Signal Description	19-2
19.3	External Signal Description	19-3
19.3.1	SCL — Serial Clock Line	19-3
19.3.2	SDA — Serial Data Line	19-3
19.4	Register Definitions	19-3
19.4.1	Module Memory Map	19-3
19.4.2	Register Descriptions	19-4
19.4.3	IIC Address Register (IIC1A)	19-4
19.4.4	IIC Frequency Divider Register (IIC1F)	19-4
19.4.5	IIC Control Register (IIC1C)	19-7
19.4.6	IIC Status Register (IIC1S)	19-8
19.4.7	IIC Data I/O Register (IIC1D)	19-9
19.5	Functional Description	19-9
19.5.1	IIC Protocol	19-9
19.6	Resets	19-13
19.7	Interrupts	19-13
19.7.1	Byte Transfer Interrupt	19-13
19.7.2	Address Detect Interrupt	19-13
19.7.3	Arbitration Lost Interrupt	19-13

## Chapter 20 Analog to Digital (ATD) Module

20.1	Introduction	20-1
20.1.1	Features	20-1
20.1.2	Modes of Operation	20-1
20.1.3	Block Diagram	20-2
20.2	Signal Description	20-3
20.2.1	Overview	20-3
20.3	Functional Description	20-3
20.3.1	Mode Control	20-4
20.3.2	Sample and Hold	20-4
20.3.3	Analog Input Multiplexer	20-6
20.3.4	ATD Module Accuracy Definitions	20-6
20.4	Resets	20-9
20.5	Interrupts	20-9

20.6	ATD Registers and Control Bits . . . . .	20-9
20.6.1	ATD Control (ATDC) . . . . .	20-10
20.6.2	ATD Status and Control (ATD1SC) . . . . .	20-12
20.6.3	ATD Result Data (ATD1RH, ATD1RL) . . . . .	20-13
20.6.4	ATD Pin Enable (ATD1PE) . . . . .	20-14

## Chapter 21 Development Support

21.1	Features . . . . .	21-1
21.2	Background Debug Controller (BDC) . . . . .	21-2
21.2.1	BKGD Pin Description . . . . .	21-3
21.2.2	Communication Details . . . . .	21-3
21.2.3	BDC Commands . . . . .	21-5
21.2.4	Coding Structure Nomenclature . . . . .	21-5
21.2.5	BDC Hardware Breakpoint . . . . .	21-8
21.3	On-Chip Debug System (DBG) . . . . .	21-8
21.3.1	Comparators A and B . . . . .	21-8
21.3.2	Bus Capture Information and FIFO Operation . . . . .	21-9
21.3.3	Change-of-Flow Information . . . . .	21-10
21.3.4	Tag vs. Force Breakpoints and Triggers . . . . .	21-10
21.3.5	Trigger Modes . . . . .	21-10
21.3.6	Hardware Breakpoints . . . . .	21-12
21.4	Register Definition . . . . .	21-12
21.4.1	BDC Registers and Control Bits . . . . .	21-12
21.4.2	System Background Debug Force Reset Register (SBDFR) . . . . .	21-15
21.4.3	DBG Registers and Control Bits . . . . .	21-15

## About This Book

This manual describes Freescale's second-generation ZigBee™ platform (the MC1321x family) which incorporates a low power 2.4 GHz radio frequency transceiver and an 8-bit microcontroller into a single 9x9x1 mm 71-pin LGA package. The MC1321x solution can be used for wireless applications from simple proprietary point-to-point connectivity, to a complete ZigBee mesh network. The combination of the radio and a microcontroller in a small footprint package allows for a cost-effective solution.

## Audience

This manual is intended for system designers.

## Organization

This document is organized into 21 chapters.

- |            |  |
|------------|--|
| Chapter 1  | <b>Introduction</b> — Briefly introduces the MC1321x. The MC1321x family is Freescale's second-generation ZigBee™ platform. This platform incorporates a low power 2.4 GHz radio frequency transceiver and an 8-bit microcontroller into a single 9x9x1 mm 71-pin LGA package. |
| Chapter 2  | <b>MC1321x Pins and Connections</b> — Describes device pinout and functionality.   |
| Chapter 3  | <b>System Considerations</b> — Describes system level considerations of the MC1321 modem and MCU.  |
| Chapter 4  | <b>MC1321x Serial Peripheral Interface (SPI)</b> — Shows how the MC1321x modem and CPU communicate primarily through the onboard SPI command channel.  |
| Chapter 5  | <b>Modem SPI Register Descriptions</b> — Details how all control, reading of status, writing of data, and reading of data is done through the MC1321x SPI port   |
| Chapter 6  | <b>Modem Timer Information</b> — Describes how the MC1321x uses its internal Event Timer block to manage system timing.  |
| Chapter 7  | <b>Modem Modes of Operation</b> — Describes the numerous MC1321x passive operational modes that allow for low-current operation as well as modes where the transceiver is active.  |
| Chapter 8  | <b>Modem Interrupt Description</b> — Shows how interrupts provide a way for the MC1321x to inform the host microcontroller (MCU) of onboard events without requiring the MCU to constantly query Mc1321x status.   |
| Chapter 9  | <b>Modem Miscellaneous Functions</b> — Describes how the MC1321x can be placed in one of two reset conditions either through hardware input M_RSTB or by writing to Reset Register 00.   |
| Chapter 10 | <b>MCU Modes of Operation</b> — Describes HCS08 operating modes and describes entry into each mode, exit from each mode, and details the functionality while in each of the modes.   |

Chapter 11	<b>MCU Memory</b> — Describes on-chip memory in the HCS08 series of MCUs and shows that it consists of RAM, FLASH program memory for non-volatile data storage, plus I/O and control/status registers.
Chapter 12	<b>MCU Resets, Interrupts, and System Configuration</b> — This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the HCS08.
Chapter 13	<b>MCU Parallel Input/Output</b> — This section explains software controls related to parallel input/output (I/O).
Chapter 14	<b>MCU Internal Clock Generator (ICG)</b> — Shows the functional organization of the internal clock generation (ICG) module and includes a general description and features list.
Chapter 15	<b>MCU Central Processor Unit (CPU)</b> — This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family.
Chapter 16	<b>MCU Keyboard Interrupt (KBI)</b> — Describes how the KBI module allows up to eight pins to act as additional interrupt sources.
Chapter 17	<b>MCU Timer/PWM (TPM Module)</b> — Details how the TPM uses one input/output (I/O) pin per channel and how the TPM shares its I/O pins with general-purpose I/O port pins.
Chapter 18	<b>MCU Serial Communications Interface (SCI)</b> — This chapter describes the SCI which allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs.
Chapter 19	<b>IIC Module</b> — Describes how the IIC bus standard compatible IIC module functions the same in normal and monitor modes. A brief description of the IIC in the various MCU modes is provided in this chapter.
Chapter 20	<b>Analog to Digital (ATD) Module</b> — Details the ATD module which is an analog-to-digital converter with a successive approximation register (SAR) architecture with sample and hold.
Chapter 21	<b>Development Support</b> — Describes the features of the background debug controller (BDC).

## Revision History

The following table summarizes revisions to this document since the previous release (Rev 1.5).

**Revision History**

Location	Revision
Chapter 5	Corrected error in Section 5.38.



## Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document.

ACK	Acknowledgement Frame
API	Application Programming Interface
BB	Baseband
CCA	Clear Channel Assessment
CRC	Cyclical Redundancy Check
DCD	Differential Chip Decoding
DME	Device Management Entity
FCS	Frame Check Sequence
FFD	Full Function Device
FFD-C	Full Function Device Coordinator
FLI	Frame Length Indicator
GTS	Guaranteed Time Slot
HW	Hardware
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LO	Local Oscillator
MAC	Medium Access Control
MCPS	MAC Common Part Sublayer
MCU	Microcontroller Unit
MLME	MAC Sublayer Management Entity
MSDU	MAC Service Data Unit
NWK	Network
PA	Power Amplifier
PAN	Personal Area Network
PANID	PAN Identification
PHY	PHYSical Layer
PIB	PAN Information Base
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
RF	Radio Frequency
RFD	Reduced Function Device
SAP	Service Access Point
SFD	Start of Frame Delimiter

SPI	Serial Peripheral Interface
SSCS	Service Specific Convergence Layer
SW	Software
VCO	Voltage Controlled Oscillator

## References

The following sources were referenced to produce this book:

- [1] IEEE 802.15.4 Standard
- [2] Freescale MC1319x Data Sheet
- [3] Freescale MC9S08GB/GT60 Data Sheet
- [4] Freescale MC1321x Data Sheet

# Chapter 1

## MC1321x Introduction

The MC1321x family is Freescale's second-generation ZigBee™ platform which incorporates a low power 2.4 GHz radio frequency transceiver and an 8-bit microcontroller into a single 9x9x1 mm 71-pin LGA package. The MC1321x solution can be used for wireless applications from simple proprietary point-to-point connectivity to a complete ZigBee mesh network. The combination of the radio and a microcontroller in a small footprint package allows for a cost-effective solution.

The MC1321x contains an RF transceiver which is an 802.15.4 Standard-compliant radio that operates in the 2.4 GHz ISM frequency band. The transceiver includes a low noise amplifier, 1mW nominal output power, PA with internal voltage controlled oscillator (VCO), integrated transmit/receive switch, on-board power supply regulation, and full spread-spectrum encoding and decoding.

The MC1321x also contains a microcontroller of the HCS08 Family of Microcontroller Units (MCU) (same die as MC9S08GBA) and can provide up to 60KB of flash memory and 4KB of RAM. The onboard MCU allows the communications stack and also the application to reside on the same system-in-package (SiP). The MC1321x family is organized as follows:

- The MC13211 has 16KB of flash and 1KB of RAM and is an ideal solution for low cost, proprietary applications that require wireless point-to-point or star network connectivity. The MC13211 combined with the Freescale Simple MAC (SMAC) provides the foundation for proprietary applications by supplying the necessary source code and application examples to get users started on implementing wireless connectivity
- The MC13212 contains 32K of flash and 2KB of RAM and is intended for use with the Freescale fully compliant 802.15.4 MAC. Custom networks based on the 802.15.4 Standard MAC can be implemented to fit user needs. The 802.15.4 Standard supports star, mesh and cluster tree topologies as well as beacons networks
- The MC13213 contains 60K of flash and 4KB of RAM and is also intended for use with the Freescale fully compliant 802.15.4 MAC and the fully ZigBee compliant Freescale BeeStack.

### NOTE

- The MC1321x now uses an updated version of the 689S08A 8-bit microprocessor to correct errata associated with the onboard FLL and reset pin. Refer to the associated errata for this new device, *Document Number MSE9S08GB60A\_4L11Y*, on the Freescale web site.
- The MC1321x also now uses an updated version of the transceiver device that is functionally fully compliant with earlier versions of the transceiver. See MC1321x errata on the Freescale web site.

However, for proper performance of the radio the following modem registers must be over-programmed:

Register 0x31 to 0xA0C0

Register 0x34 to 0xFEC6

These registers must be over-programmed for MC1321x devices in which the modem Chip\_ID Register 0x2C reads 0x6800.

Applications include, but are not limited to, the following:

- Residential and commercial automation
  - Lighting control
  - Security
  - Access control
  - Heating, ventilation, air-conditioning (HVAC)
  - Automated meter reading
- Industrial Control
  - Asset tracking and monitoring
  - Homeland security
  - Process management
  - Environmental monitoring and control
  - HVAC
  - Automated meter reading (AMR)
- Health Care
  - Patient monitoring
  - Fitness monitoring
- Consumer
  - Human interface devices (keyboard, mice, etc.)
  - Remote control
  - Wireless toys

## 1.1 Ordering Information

Table 1-1 lists additional devices in the MC1321x family.

### NOTE

The device marking for silicon revision 1.1 and newer is different than version 1.0 and older. For more details about the 71-pin LGA package used for the MC1321x family, see the *802.15.4/ZigBee Hardware Design Considerations Reference Manual (ZHDCRM)*.

**Table 1-1. Devices in the MC1321x Family**

Device	Operating Temp Range (TA.)	Package	Memory Options	Description
MC13211	-40° to 85° C	LGA	1KB RAM, 16KB Flash	Intended for proprietary applications and Freescale Simple MAC (SMAC)
MC13211R2	-40° to 85° C	LGA Tape and Reel	1KB RAM, 16KB Flash	Intended for proprietary applications and Freescale Simple MAC (SMAC)
MC13212	-40° to 85° C	LGA	2KB RAM, 32KB Flash	Intended for 802.15.4 Standard compliant applications and Freescale 802.15.4 MAC
MC13212R2	-40° to 85° C	LGA Tape and Reel	2KB RAM, 32KB Flash	Intended for 802.15.4 Standard compliant applications and Freescale 802.15.4 MAC
MC13213	-40° to 85° C	LGA	4KB RAM, 60KB Flash	Intended for 802.15.4 Standard compliant applications and the Freescale 802.15.4 MAC and fully ZigBee compliant Freescale BeeStack.
MC13213R2	-40° to 85° C	LGA Tape and Reel	4KB RAM, 60KB Flash	Intended for 802.15.4 Standard compliant applications and Freescale 802.15.4 MAC and fully ZigBee compliant Freescale BeeStack.

## 1.2 General Platform Features

- 802.15.4 Standard compliant on-chip transceiver/modem
  - 2.4GHz
  - 16 selectable channels
  - programmable output power
- Multiple power saving modes
- 2V to 3.4V operating voltage with on-chip voltage regulators for modem
- -40°C to +85°C temperature range
- Low external component count
- Supports single 16 MHz crystal clock source operation or dual crystal operation
- Support for SMAC, 802.15.4, and ZigBee software
- 9mm x 9mm x 1mm 71-pin LGA

## 1.3 Microcontroller Features

- Low voltage MCU with 40 MHz low power HCS08 CPU core
- Up to 60K flash memory with block protection and security and 4K RAM
  - MC13211: 16KB Flash, 1KB RAM
  - MC13212: 32KB Flash, 2KB RAM
  - MC13213: 60KB Flash, 4KB RAM
- Low power modes (Wait plus three Stop modes)
- Dedicated serial peripheral interface (SPI) connected internally to 802.15.4 modem
- One external 4-channel (5-channel internal) 16-bit timer/pulse width modulator (TPM) module and one external 1-channel (3-channel internal) 16-bit timer/pulse width modulator module, each with selectable input capture, output capture, and PWM capability.
- 8-bit port keyboard interrupt (KBI)
- 8-channel 8-10-bit ADC
- Two independent serial communication interfaces (SCI)
- Multiple clock source options
  - Internal clock generator (ICG) with 243 kHz oscillator that has +/-0.2% trimming resolution and +/-0.5% deviation across voltage.
  - Start-up oscillator of approximately 8 MHz
  - External crystal or resonator
  - External source from modem clock for very high accuracy source or system low-cost option
- Inter-integrated circuit (IIC) interface with 100 kbps operation
- In-circuit debug and flash programming available via on-chip background debug module (BDM)
  - Two comparator and 9 trigger modes
  - Eight deep FIFO for storing change-of-flow addresses and event-only data
  - Tag and force breakpoints
  - In-circuit debugging with single breakpoint
- System protection features
  - Programmable low voltage interrupt (LVI)
  - Optional watchdog timer (COP)
  - Illegal opcode detection
- Up to 32 MCU GPIO with programmable pullups

## 1.4 RF Modem Features

- Fully compliant 802.15.4 Standard transceiver supports 250 kbps O-QPSK data in 5.0 MHz channels and full spread-spectrum encode and decode
- Operates on one of 16 selectable channels in the 2.4 GHz ISM band
- -1 to 0 dBm nominal output power, programmable from -27 dBm to +3 dBm typical
- Receive sensitivity of <-92 dBm (typical) at 1% PER, 20-byte packet, much better than the 802.15.4 Standard of -85 dBm
- Integrated transmit/receive switch
- Dual PA output pairs which can be programmed for full differential single port or dual port operation that supports an external LNA and/or PA.
- Three low power modes for increased battery life
- Programmable frequency clock output for use by MCU
- Onboard trim capability for 16 MHz crystal reference oscillator eliminates need for external variable capacitors and allows for automated production frequency calibration
- Four internal timer comparators available to supplement MCU timer resources
- Supports both Packet Mode and Streaming Mode
- Seven GPIO to supplement MCU GPIO

## 1.5 Software Solutions

Freescale provides a powerful software environment called the Freescale BeeKit Wireless Connectivity Toolkit. BeeKit is a comprehensive codebase of wireless networking libraries, application templates, and sample applications. The BeeKit Graphical User Interface (GUI), part of the BeeKit Wireless Connectivity Toolkit, allows users to create, modify, and update various wireless networking implementations. A wide range of software functionality is available to complement the MC1321x and these are provided as codebases within BeeKit. The following sections describe the available tools.

### 1.5.1 Simple Media Access Controller (SMAC)

The Freescale Simple Media Access Controller (SMAC) is a simple ANSI C based code stack available as sample source code. The SMAC can be used for developing proprietary RF transceiver applications using the MC1321x.

- Small memory footprint (about 3 kilobytes typical)
- Supports point-to-point and star network configurations
- Proprietary networks
- Source code and application examples provided

### 1.5.2 IEEE 802.15.4 2006 Standard-Compliant MAC

The Freescale 802.15.4 Standard-Compliant MAC is a code stack available as object code. The 802.15.4 MAC can be used for developing MC1321x networking applications based on the full IEEE® 802.15.4 Standard that use custom Network Layer and application software.

- Supports star, mesh and cluster tree topologies
- Supports beacons networks
- Supports GTS for low latency
- Multiple power saving modes
- AES-128 Security module
- 802.15.4 Sequence support
- 802.15.4 Receiver Frame filtering.

### 1.5.3 SynkroRF Platform

The SynkroRF Network is a general purpose, proprietary networking layer that sits on top of the IEEE<sup>®</sup> 802.15.4 MAC and PHY layers. It is designed for Wireless Personal Area Networks (WPANs) and conveys information over short distances among the participants in the network. It enables small, power efficient, inexpensive solutions to be implemented for a wide range of applications. Some key characteristics of an SynkroRF Network are:

- An over-the-air data rate of 250 kbit/s in the 2.4 GHz band.
- 3 independent communication channels in the 2.4 GHz band (15, 20, and 25).
- 2 network node types, controller and controlled nodes.
- Channel Agility mechanism.
- Low Latency Tx mode automatically enabled in conditions of radio interference.
- Fragmented mode transmission and reception, automatically enabled in conditions of radio interference.
- Robustness and ease of use.
- Essential functionality to build and support a CE network.

The SynkroRF Network layer uses components from the standard HC(S)08 Freescale platform, which is also used by the Freescale's implementations of 802.15.4. MAC and ZigBee<sup>™</sup> layers. For more details about the platform components, see the *Freescale Platform Reference Manual*.

### 1.5.4 BeeStack Consumer

Freescale's ZigBee RF4CE stack, called BeeStack Consumer, is a networking layer that sits on top of the IEEE<sup>®</sup> 802.15.4 MAC and PHY layers. It is designed for standards-based Wireless Personal Area Networks (WPANs) of home entertainment products and conveys information over short distances among the participants in the network. It enables small, power efficient, inexpensive solutions to be implemented for a wide range of applications. Targeted applications include DTV, set top box, A/V receivers, DVD players, security, and other consumer products.

Some key characteristics of a BeeStack Consumer network are:

- An over-the-air data rate of 250 kbit/s in the 2.4 GHz band
- 3 independent communication channels in the 2.4 GHz band
- 2 network node types, controller node and target node



- Channel Agility mechanism
- Provides robustness and ease of use
- Includes essential functionality to build and support a CE network

The BeeStack Consumer layer uses components from the standard HCS08 Freescale platform, which is also used by the Freescale implementations of 802.15.4. MAC or ZigBee™ layers. For more details about the platform components, see the *Freescale Platform Reference Manual*.

### 1.5.5 ZigBee-Compliant Network Stack

Freescale's BeeStack architecture builds on the ZigBee protocol stack. Based on the OSI Seven-Layer model, the ZigBee stack ensures inter-operability among networked devices. The physical (PHY), media access control (MAC), and network (NWK) layers create the foundation for the application (APL) layers. BeeStack defines additional services to improve the communication between layers of the protocol stack.

At the Application Layer, the application support layer (ASL) facilitates information exchange between the Application Support Sub-Layer (APS) and application objects. Finally, ZigBee Device Objects (ZDO), in addition to other manufacturer-designed applications, allow for a wide range of useful tasks applicable to home and industrial automation.

BeeStack uses the IEEE 802.15.4-compliant MAC/PHY layer that is not part of ZigBee itself. The NWK layer defines routing, network creation and configuration, and device synchronization. The application framework (AF) supports a rich array of services that define ZigBee functionality. ZigBee Device Objects (ZDO) implement application-level services in all nodes via profiles. A security service provider (SSP) is available to the layers that use encryption (NWK and APS), i.e., Advanced Encryption Standard (AES) 128-bit security.

The complete Freescale BeeStack protocol stack includes the following components:

- ZigBee Device Objects (ZDO) and ZigBee Device Profile (ZDP)
- Application Support Sub-Layer (APS)
- Application Framework (AF)
- Network (NWK) Layer
- Security Service Provider (SSP)
- IEEE 802.15.4-compliant MAC and Physical (PHY) Layer

## 1.6 System Block Diagram

Figure 1-1 shows a simplified block diagram of the MC1321x solution.

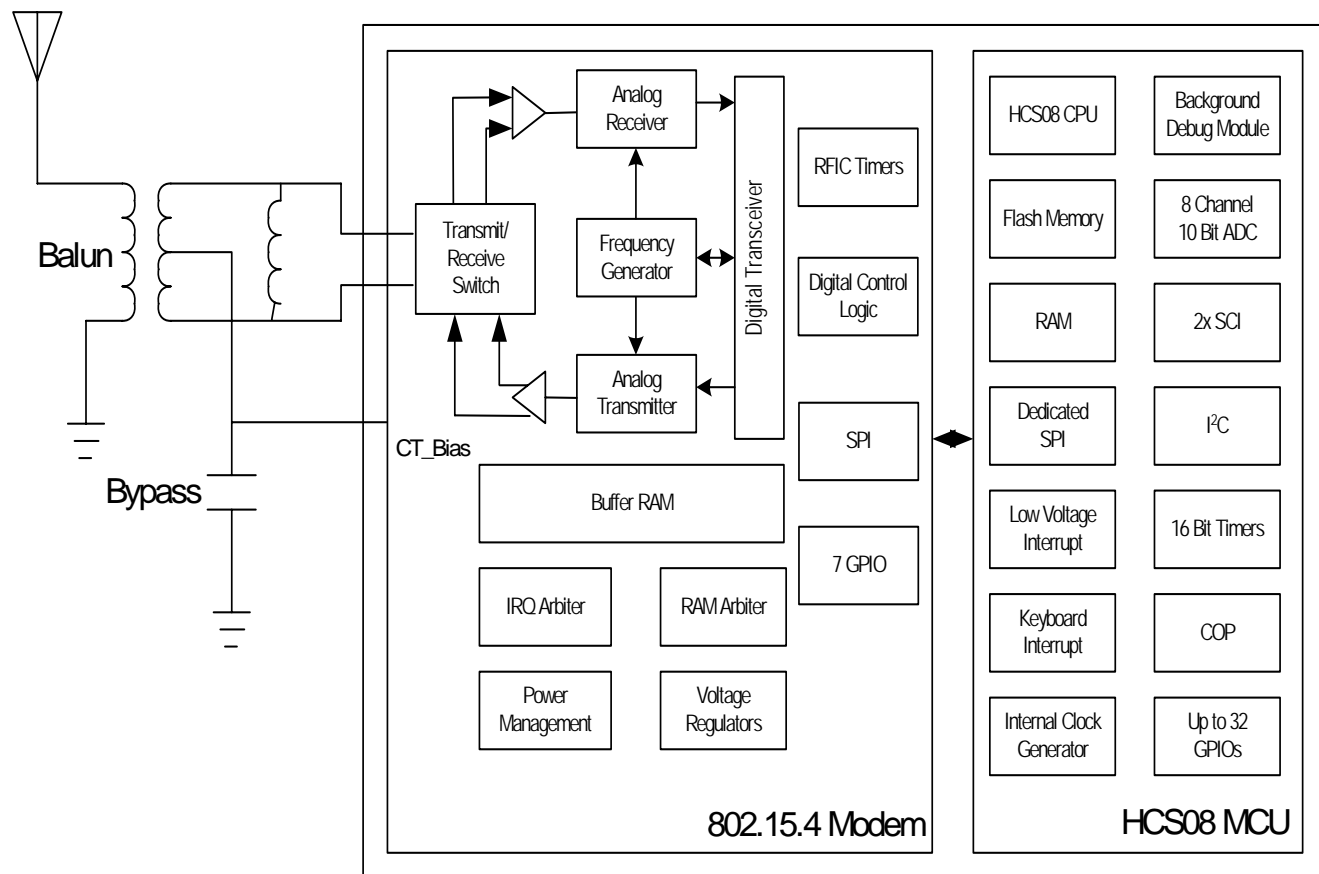


Figure 1-1. MC1321x System Level Block Diagram

## 1.7 802.15.4 Modem Overview

### 1.7.1 Modem Block Diagram

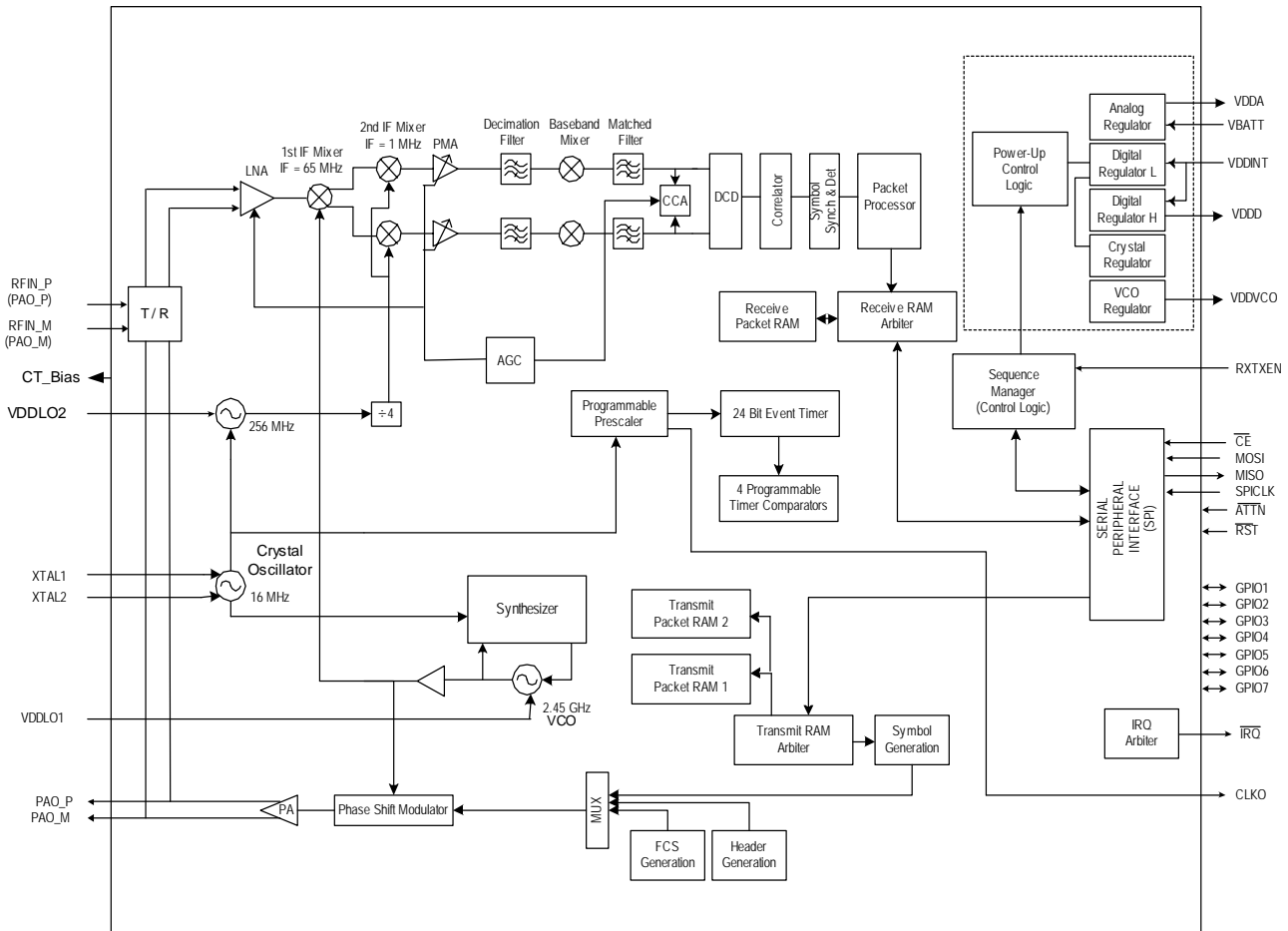


Figure 1-2. 802.15.4 Modem Block Diagram

### 1.7.2 Modem Data Transfer Modes

The MC1321x has two data transfer modes:

1. Packet Mode — Data is buffered in on-chip RAM
2. Streaming Mode — Data is processed word-by-word

The Freescale 802.15.4 MAC software only supports the Streaming Mode of data transfer. For proprietary applications, Packet Mode can be used to conserve MCU resources.

### 1.7.3 Modem Packet Structure

Figure 1-3 shows the 802.15.4 Standard compliant packet structure of the MC1321x. Payloads of up to 125 bytes are supported. The MC1321x adds a four-byte preamble, a one-byte Start of Frame Delimiter (SFD), and a one-byte Frame Length Indicator (FLI) before the data. A Frame Check Sequence (FCS) is calculated and appended to the end of the data.

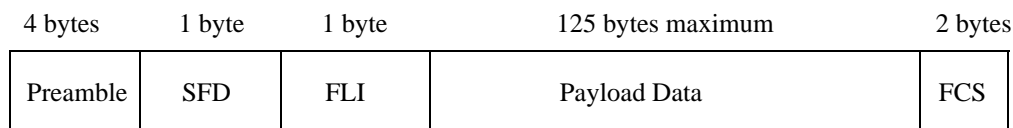


Figure 1-3. MC1321x Packet Structure

### 1.7.4 Modem Receive Path Description

In the receive signal path, the RF input is converted to low IF In-phase and Quadrature (I & Q) signals through two down-conversion stages. A Clear Channel Assessment (CCA) can be performed based upon the baseband energy integrated over a specific time interval. The digital back end performs Differential Chip Detection (DCD), the correlator “de-spreads” the Direct Sequence Spread Spectrum (DSSS) Offset QPSK (O-QPSK) signal, determines the symbols and packets, and detects the data.

The preamble, SFD, and FLI are parsed and used to detect the payload data and FCS (which are stored in RAM in Packet Mode). A two-byte FCS is calculated on the received data and compared to the FCS value appended to the transmitted data, which generates a Cyclical Redundancy Check (CRC) result. A parameter of received energy during the reception called the Link Quality Indicator is measured over a 64  $\mu$ s period after the packet preamble and stored in an SPI register.

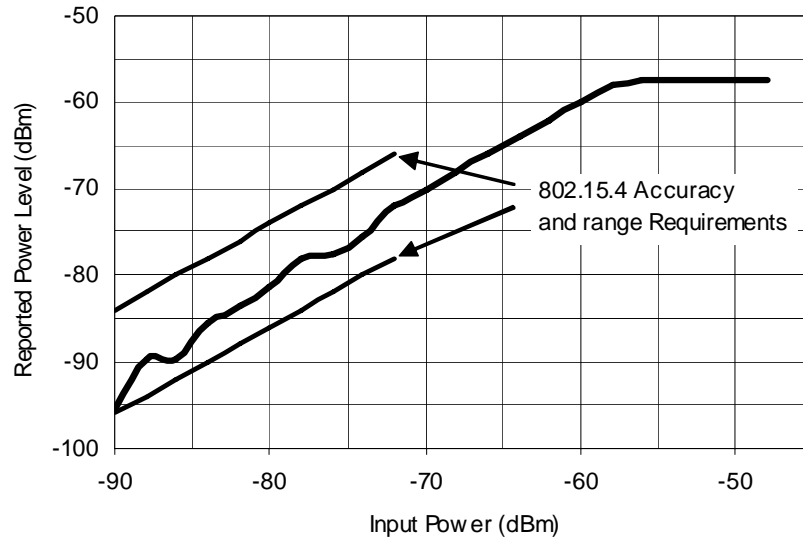
If the MC1321x is in Packet Mode, the data is stored in RAM and processed as an entire packet. The MCU is notified that an entire packet has been received via an interrupt.

If the MC1321x is in Streaming Mode, the MCU is notified by a recurring interrupt on a word-by-word basis as data is received.

Figure 1-4 shows CCA reported power level versus input power. Note that CCA reported power saturates at about -57 dBm input power which is well above 802.15.4 Standard requirements.

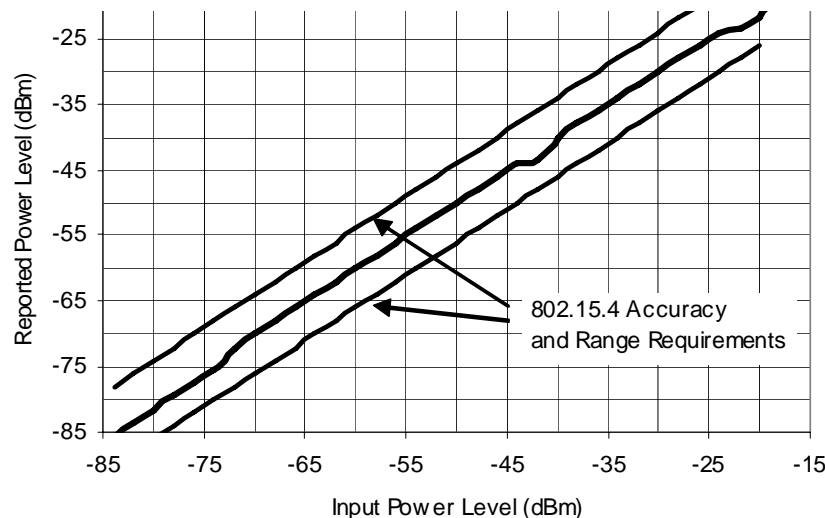
**NOTE**

For both graphs, the required 802.15.4 Standard accuracy and range limits are shown. A 3.5 dBm offset has been programmed into the CCA reporting level to center the level over temperature in the graphs.



**Figure 1-4. Reported Power Level versus Input Power in Clear Channel Assessment Mode**

Figure 1-5 shows energy detection/LQI reported level versus input power.



**Figure 1-5. Reported Power Level Versus Input Power for Energy Detect or Link Quality Indicator**

## 1.7.5 Modem Transmit Path Description

For the transmit path, the TX data that was previously written to the internal RAM is retrieved (Packet Mode) or the TX data is clocked in via the SPI (Streaming Mode), formed into packets per the 802.15.4 PHY, spread, and then up-converted to the transmit frequency.

If the MC1321x is in Packet Mode, data is processed as an entire packet. The data is first loaded into the TX buffer. The MCU then requests that the modem transmit the data. The MCU is notified via an interrupt when the whole packet has successfully been transmitted.

In Streaming Mode, the data is fed to the MC1321x on a word-by-word basis with an interrupt serving as a notification that the MC1321x is ready for more data. This continues until the whole packet is

transmitted. In both modes, a two-byte FCS is calculated in hardware from the payload data and appended to the packet. This done without intervention from the user.

### 1.7.6 Radio Usage

The MC1321x RF analog interface has been designed to provide maximum flexibility as well as low external part count and cost. An optional on-chip transmit/receive (T/R) switch with the center tap bias switch (CT\_Bias) output can be used for a simple single antenna interface with an external balun. Alternately, separate full differential RFIN and PAO outputs can be utilized for separate RX and TX antennas or external LNA and PA designs.

Figure 1-6 shows a single antenna configuration in which the MC1321x internal T/R switch is used. The balun converts the single-ended antenna to differential signals that interface to the RFIN\_x (PAO\_x) pins of the radio. The CT\_bias pin provides the proper bias point to the balun depending on operation, that is, CT\_bias is at VDDA voltage for transmit and is at ground for receive. The internal T/R switch enables the signal to an onboard LNA for receive and enables the onboard PAs for transmit.

Use of the RF interface is described in detail in Section 3.10, “Transceiver RF Configurations and External Connections”.

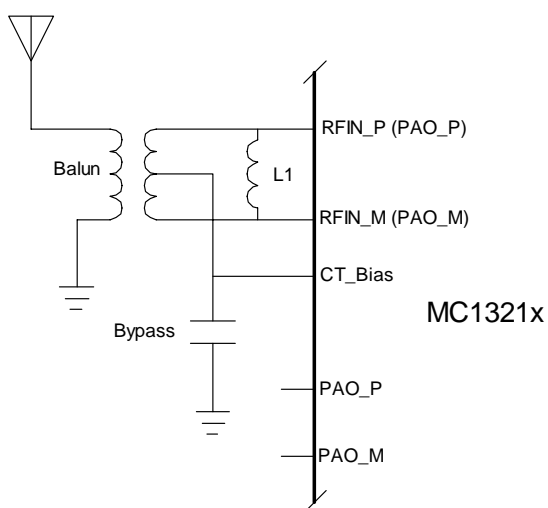


Figure 1-6. Using the MC1321x with the Onboard T/R Switch

## 1.8 MCU Overview

The MC1321x MCU is a member of the Freescale family of HCS08 microcontrollers which contains new instructions to implement rapid debugging and development and still be fully compatible with all legacy code written for the M68HC08 family. The MCU consists of the HCS08 core plus memory and peripheral modules. The maximum clock speed of the HCS08 CPU is 40 MHz and the maximum MCU bus rate is 20 MHz (half the CPU clock frequency). The MCU is an excellent low voltage, low power device suitable to 802.15.4 Standard and ZigBee specification.

The functional options available in the MC1321x family of devices vary only in the memory configurations as detailed in Table 1-1. All peripheral modules and CPU functionality are similar across the MC1321x family.

Although detailed information is provided in this reference manual, the user is directed to the *HCS08 Family Reference Manual*, (HCS08RMv1, Rev. 1) for general family reference. Background information on Development Support and extended Instruction Set Details are especially useful.

### 1.8.1 MCU Block Diagram

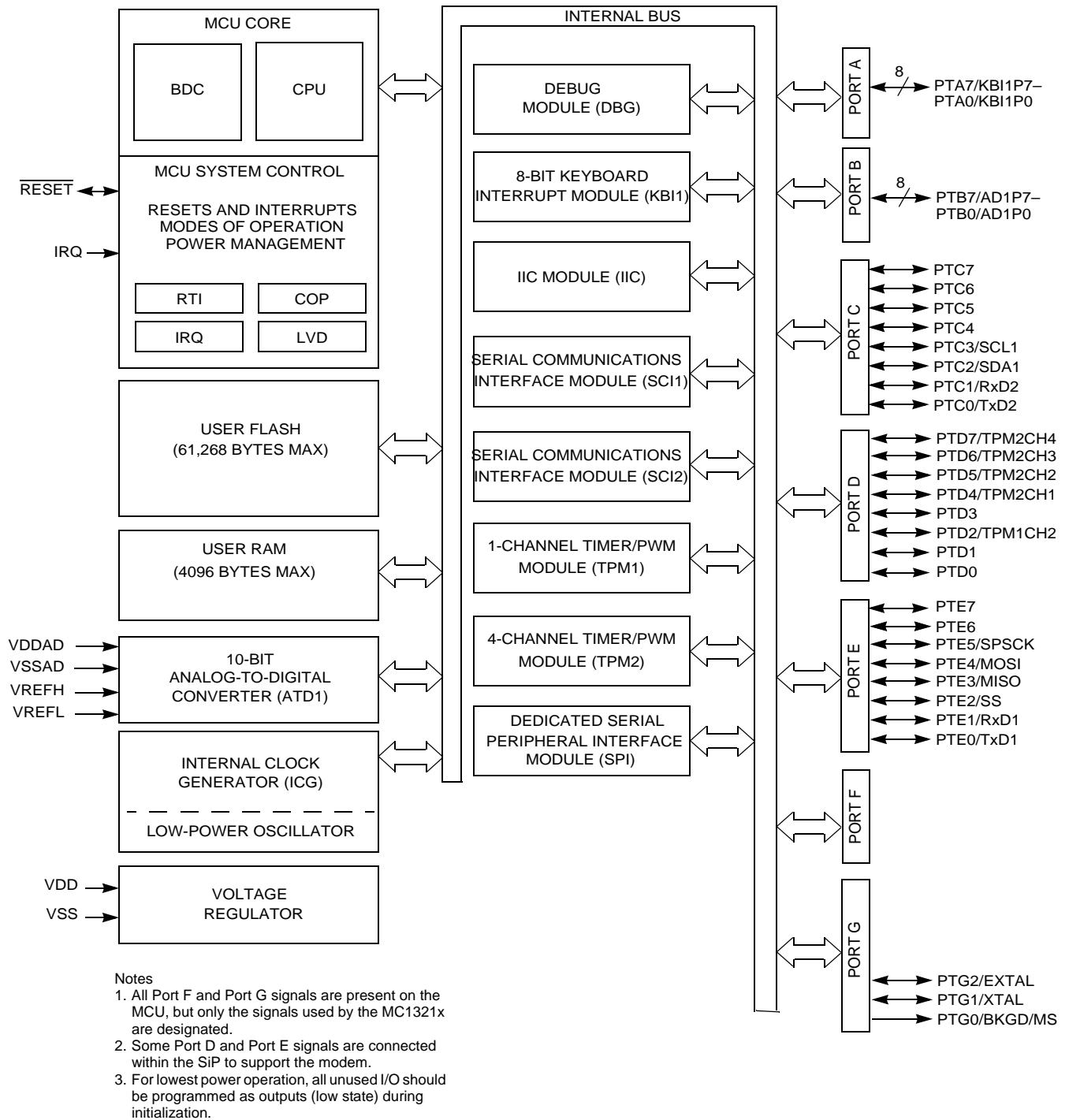


Figure 1-7. MCU Block Diagram

## NOTE

The MC1321x MCU has port signals that are not pinned-out. These signals should be initialized to a known condition for low power modes.

### 1.8.2 HCS08 Central Processing Unit (CPU)

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). The HCS08 has five CPU registers that are not part of the memory map (the HCS08 is an accumulator-based instruction set). Figure 1-8 show the programmer's model for the CPU.

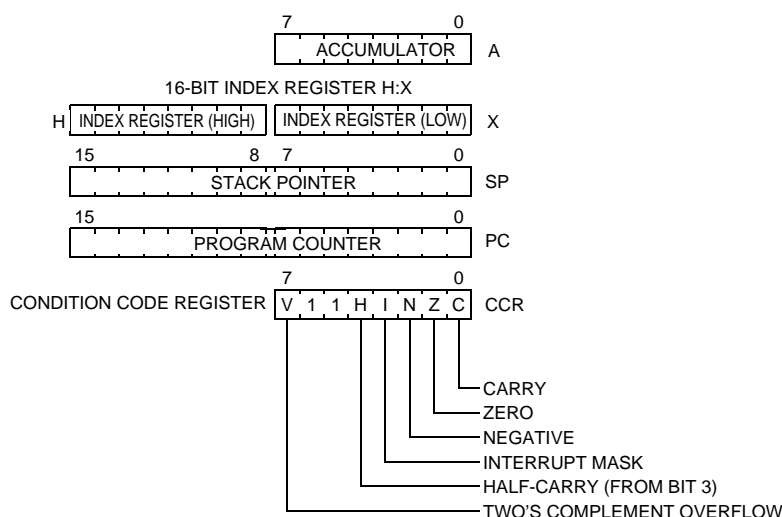


Figure 1-8. CPU Registers

### 1.8.3 MCU Peripheral Modules

The MCU includes the following peripheral modules:

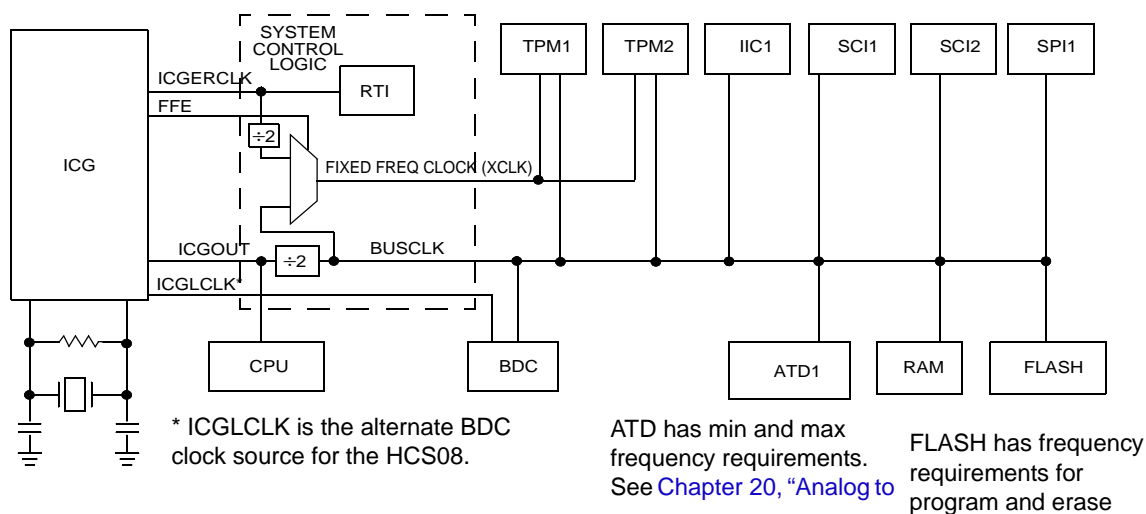
- Dedicated serial peripheral interface (SPI) connected internally to 802.15.4 modem - The MCU SPI port is normally a full function SPI bus with both master and slave capability and programmable clock interface. Because the SPI is dedicated to the modem and internally tied to the modem SPI port, the MCU SPI port has to be programmed as a master and meet the timing requirement of the modem SPI interface
- One 4-channel (TPM2) 16-bit timer/pulse width modulator module and one 1-channel (TPM1) 16-bit timer/pulse width modulator module, each with selectable input capture, output capture, and PWM capability. The MCU contains the hardware for a full 3-channel TPM1 and a full 5-channel TPM2, but pin count limits the use of TPM1 to a single channel and TPM2 to 4 channels
- 8-bit port keyboard interrupt (KBI)
- 8-channel 10-bit ADC
- Two independent serial communication interfaces (SCI1 and SCI2) - Each SCI has a common TX and RX baud rate and each has a separate baud rate generator



- Internal Clock Generator (ICG) - Supports external clock source or external crystal oscillator and has internal reference generator with two controlled sources. One is approximately 8 MHz and can be selected as local clock for background debug controller. The other is typically 243 kHz and can be trimmed for a low-cost MCU clock source
- Inter-integrated circuit (IIC) with 100 kbps operation
- In-circuit debug and flash programming available via on-chip background debug module (BDM)
- Up to 32 MCU GPIO with programmable pullups - The HCS08 MCU has a total of 7 ports with each having 8 GPIO. Because of the MC1321x package limitations, only 32 of the GPIO are available for use. There are multiple use IO, high drive IO, programmable pullups and other features. The MCU GPIO are covered in detail in [Chapter 13, “MCU Parallel Input/Output”](#). Unused GPIO and GPIO not pinned-out should be initialized to a known condition.

### 1.8.4 MCU Internal Clock Distribution

Some of the modules inside the MCU have clock source choices. [Figure 1-9](#) shows a simplified clock connection diagram.



**Figure 1-9. MCU Internal Clock Distribution**

The clocks are derived from the ICG and the ICG supplies the clock sources:

- ICGOUT is an output of the ICG module. It is one of the following:
  - The external crystal oscillator
  - An external clock source (can be driven by CLKO output from the modem)
  - The output of the digitally-controlled oscillator (DCO) in the frequency-locked loop sub-module

Control bits inside the ICG determine which source is connected

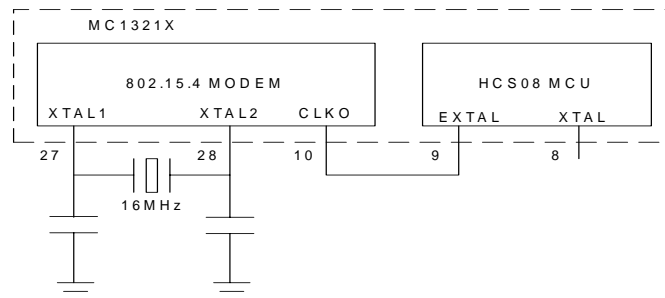
- FFE is a control signal generated inside the ICG. If the frequency of ICGOUT  $> 4 \times$  the frequency of ICGERCLK, this signal is a logic 1 and the fixed-frequency clock will be the ICGERCLK. Otherwise the fixed-frequency clock will be BUSCLK

- ICGLCLK — Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow
- ICGERCLK — External reference clock can be selected as the real-time interrupt clock source

The MCU ICG is described in detail by [Chapter 14, “MCU Internal Clock Generator \(ICG\)”](#).

## 1.9 System Clock Configuration

The MC1321x provides several options for the system clock configuration. The modem requires a 16MHz crystal for its source oscillator and can also supply a selectable frequency clock out (CLKO). The CLKO frequency can be programmed for 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, 62.5 kHz, 32.786+ kHz (default), or 16.393+ kHz. The MCU can use a second crystal if desired or it can derive its external clock source from the modem CLKO as shown in [Figure 1-10](#). The MCU also has an internal clock generator that can be used for start-up and when the modem is shutdown (for lowest power). The MCU internal reference is approximately 243 kHz.



**Figure 1-10. MC1321x Single Clock**

# Chapter 2

## MC1321x Pins and Connections

### 2.1 Device Pin Assignment

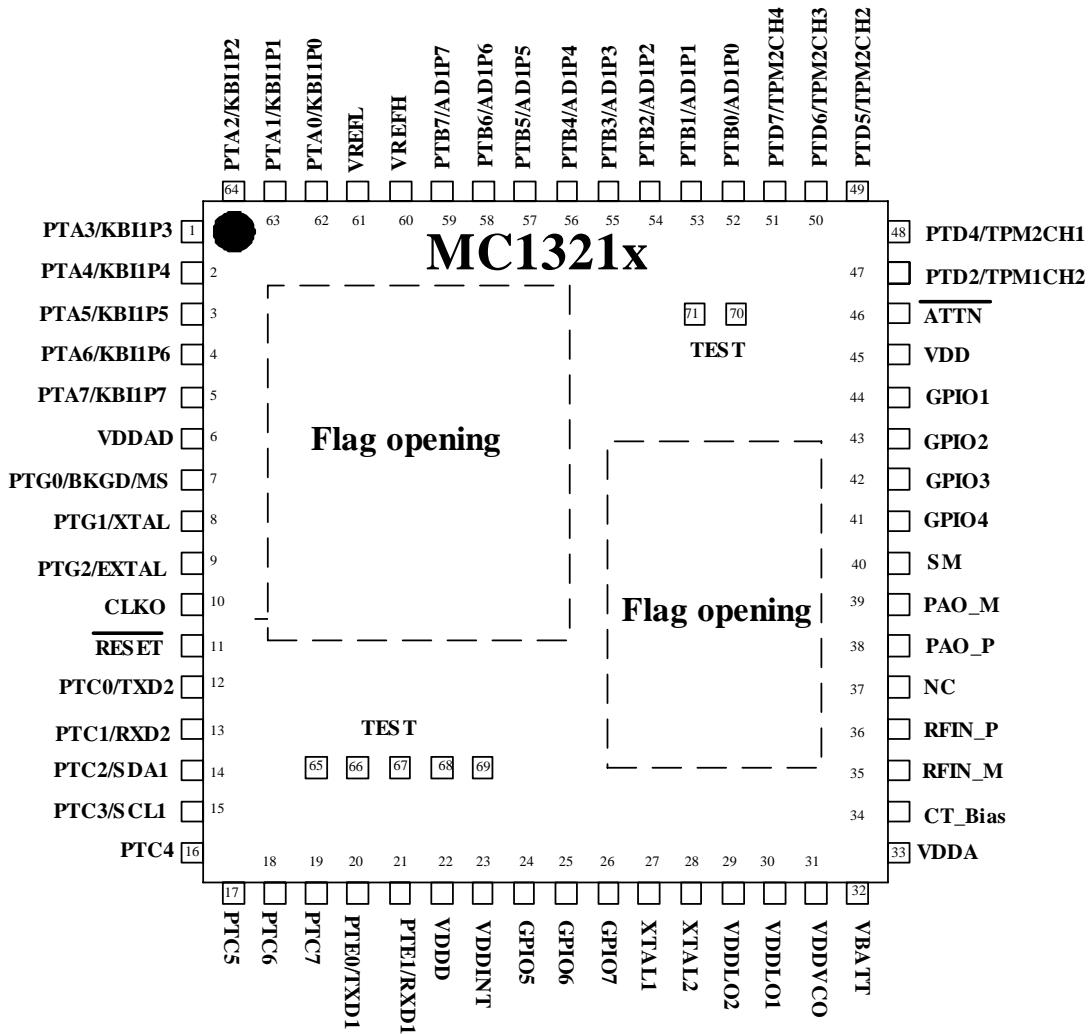


Figure 2-1. MC1321x Pinout

## 2.2 Pin Definitions

Table 2-1 details the MC1321x pinout and functionality.

**Table 2-1. Pin Function Description**

Pin #	Pin Name	Type	Description	Functionality
1	PTA3/KBI1P3	Digital Input/Output	MCU Port A Bit 3 / Keyboard Input Bit 3	
2	PTA4/KBI1P4	Digital Input/Output	MCU Port A Bit 4 / Keyboard Input Bit 4	
3	PTA5/KBI1P5	Digital Input/Output	MCU Port A Bit 5 / Keyboard Input Bit 5	
4	PTA6/KBI1P6	Digital Input/Output	MCU Port A Bit 6 / Keyboard Input Bit 6	
5	PTA7/KBI1P7	Digital Input/Output	MCU Port A Bit 7 / Keyboard Input Bit 7	
6	VDDAD	Power Input	MCU power supply to ATD	Decouple to ground.
7	PTG0/BKGND/MS	Digital Input/Output	MCU Port G Bit 0/ Background/Mode Select	PTG0 is output only. Pin is I/O when used as BDM function.
8	PTG1/XTAL	Digital Input/Output	MCU Port G Bit 1/ Crystal oscillator output	Full I/O when not used as clock source.
9	PTG2/EXTAL	Digital Input/Output	MCU Port G Bit 2/ Crystal oscillator input	Full I/O when not used as clock source.
10	CLKO	Digital Output	Modem Clock Output	Programmable frequencies of: 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, 62.5 kHz, 32.786+ kHz (default), and 16.393+ kHz.
11	RESET	Digital Input/Output	MCU reset. Active low	
12	PTC0/TXD2	Digital Input/Output	MCU Port C Bit 0 / SCI2 TX data out	
13	PTC1/RXD2	Digital Input/Output	MCU Port C Bit 1/ SCI2 RX data in	
14	PTC2/SDA1	Digital Input/Output	MCU Port C Bit 1/ IIC bus data	
15	PTC3/SCL1	Digital Input/Output	MCU Port C Bit 1/ IIC bus clock	
16	PTC4	Digital Input/Output	MCU Port C Bit 4	
17	PTC5	Digital Input/Output	MCU Port C Bit 5	
18	PTC6	Digital Input/Output	MCU Port C Bit 6	

**Table 2-1. Pin Function Description (continued)**

Pin #	Pin Name	Type	Description	Functionality
19	PTC7	Digital Input/Output	MCU Port C Bit 7	
20	PTE0/TXD1	Digital Input/Output	MCU Port E Bit 0 / SCI1 TX data out	
21	PTE1/RXD1	Digital Input/Output	MCU Port E Bit 1/ SCI1 RX data in	
22	VDDD	Power Output	Modem regulated output supply voltage	Decouple to ground.
23	VDDINT	Power Input	Modem digital interface supply	2.0 to 3.4 V. Decouple to ground. Connect to Battery.
24	GPIO5	Digital Input/Output	Modem General Purpose Input/Output 5	
25	GPIO6	Digital Input/Output	Modem General Purpose Input/Output 6	
26	GPIO7	Digital Input/Output	Modem General Purpose Input/Output 7	
27	XTAL1	Input	Modem crystal reference oscillator input	Connect to 16 MHz crystal and load capacitor.
28	XTAL2	Input/Output	Modem crystal reference oscillator output	Connect to 16 MHz crystal and load capacitor. Do not load this pin by using it as a 16 MHz source. Measure 16 MHz output at CLKO, programmed for 16 MHz.
29	VDDLO2	Power Input	Modem LO2 VDD supply	Connect to VDDA externally.
30	VDDLO1	Power Input	Modem LO1 VDD supply	Connect to VDDA externally.
31	VDDVCO	Power Output	Modem VCO regulated supply bypass	Decouple to ground.
32	VBATT	Power Input	Modem voltage regulators' input	Decouple to ground. Connect to Battery.
33	VDDA	Power Output	Modem analog regulated supply output	Decouple to ground. Connect to directly VDDLO1 and VDDLO2 externally and to PAO_P and PAO_M through a bias network.
34	CT_Bias	RF Control Output	Modem bias voltage/control signal for RF external components	When used with internal T/R switch, provides ground reference for RX and VDDA reference for TX. Can also be used as a control signal with external LNA, antenna switch, and/or PA.
35	RFIN_M	RF Input (Output)	Modem RF input/output negative	When used with internal T/R switch, this is a bi-directional RF port for the internal LNA and PA

**Table 2-1. Pin Function Description (continued)**

Pin #	Pin Name	Type	Description	Functionality
36	RFIN_P	RF Input (Output)	Modem RF input/output positive	When used with internal T/R switch, this is a bi-directional RF port for the internal LNA and PA
37	NC		Not used	May be grounded or left open
38	PAO_P	RF Output	Modem power amplifier RF output positive	Open drain. Connect to VDDA through a bias network when used with external balun. Not used when internal T/R switch is used.
39	PAO_M	RF Output	Modem power amplifier RF output negative	Open drain. Connect to VDDA through a bias network when used with external balun. Not used when internal T/R switch is used.
40	SM	Input	Test Mode pin	Must be grounded for normal operation
41	GPIO4	Digital Input/Output	Modem General Purpose Input/Output 4	
42	GPIO3	Digital Input/Output	Modem General Purpose Input/Output 3	
43	GPIO2	Test Point	MCU Port E Bit 6 / Modem General Purpose Input/Output 2	Internally connected pins. When gpio_alt_en, Register 9, Bit 7 = 1, GPIO2 functions as a "CRC Valid" indicator.
44	GPIO1	Test Point	MCU Port E Bit 7 / Modem General Purpose Input/Output 1	Internally connected pins. When gpio_alt_en, Register 9, Bit 7 = 1, GPIO1 functions as an "Out of Idle" indicator.
45	VDD	Power Input	MCU main power supply	Decouple to ground.
46	ATTN	Test Point	MCU Port D Bit 0 / Modem attention input	Internally connected pins.
47	PTD2/TPM1CH2	Digital Input/Output	MCU Port D Bit 2 / TPM1 Channel 2	
48	PTD4/TPM2CH1	Digital Input/Output	MCU Port D Bit 4 / TPM2 Channel 1	
49	PTD5/TPM2CH2	Digital Input/Output	MCU Port D Bit 5 / TPM2 Channel 2	
50	PTD6/TPM2CH3	Digital Input/Output	MCU Port D Bit 6 / TPM2 Channel 3	
51	PTD7/TPM2CH4	Digital Input/Output	MCU Port D Bit 7 / TPM2 Channel 4	
52	PTB0/AD1P0	Input/Output	MCU Port B Bit 0 / ATD analog Channel 0	
53	PTB1/AD1P1	Input/Output	MCU Port B Bit 1 / ATD analog Channel 1	
54	PTB2/AD1P2	Input/Output	MCU Port B Bit 2 / ATD analog Channel 2	

**Table 2-1. Pin Function Description (continued)**

Pin #	Pin Name	Type	Description	Functionality
55	PTB3/AD1P3	Input/Output	MCU Port B Bit 3 / ATD analog Channel 3	
56	PTB4/AD1P4	Input/Output	MCU Port B Bit 4 / ATD analog Channel 4	
57	PTB5/AD1P5	Input/Output	MCU Port B Bit 5 / ATD analog Channel 5	
58	PTB6/AD1P6	Input/Output	MCU Port B Bit 6 / ATD analog Channel 6	
59	PTB7/AD1P7	Input/Output	MCU Port B Bit 7 / ATD analog Channel 7	
60	VREFH	Input	MCU high reference voltage for ATD	
61	VREFL	Input	MCU low reference voltage for ATD	
62	PTA0/KBI1P0	Digital Input/Output	MCU Port A Bit 0 / Keyboard Input Bit 0	
63	PTA1/KBI1P1	Digital Input/Output	MCU Port A Bit 1 / Keyboard Input Bit 1	
64	PTA2/KBI1P2	Digital Input/Output	MCU Port A Bit 2 / Keyboard Input Bit 2	
65	PTE5/SPSCK1	SPICLK	MCU SPI master SPI clock output drives modem SPICLK slave clock input.	Normally factory test. Do not connect
66	PTE4/MOSI1	MOSI	MCU SPI master MOSI output drives modem slave MOSI input	Normally factory test. Do not connect
67	PTE3/MISO1	MISO	Modem SPI slave MISO output drives MCU master MISO input	Normally factory test. Do not connect
68	PTE2/ $\overline{SS1}$	CE	MCU SPI master $\overline{SS}$ output drives modem slave $\overline{CE}$ input	Normally factory test. Do not connect
69	IRQ	M_IRQ	Modem interrupt request $\overline{M\_IRQ}$ output drives MCU IRQ input	Normally factory test. Do not connect
70	PTD1	RXTXEN	MCU Port D Bit 1 drives the RXTXEN input to the modem to enable TX or RX or CCA operations.	Normally factory test. Do not connect
71	PTD3	M_RST	MCU Port D Bit 3 drives the reset $\overline{M\_RST}$ input to the modem.	Normally factory test. Do not connect
FLAG	VSS	Power input	External package flag. Common VSS	Connect to ground.





# Chapter 3

## System Considerations

### 3.1 Introduction

The MC1321x is the embodiment of a 802.15.4 node in a single SiP package which can provide solutions to proprietary nets, 802.15.4 MAC-compatible nets, or full ZigBee-compatible nets. All control of the node is done through the onboard HCS08 processor, and all MCU peripherals, MCU GPIO, modem functionality, and modem GPIO are manipulated by the processor. The MCU GPIO and MCU peripherals are accessed as ports from the MCU internal bus and can be programmed directly.

Communication to the modem function is through the common SPI bus, the MCU interrupt request, and several MCU GPIO lines. Primary interface with the modem is through the SPI command structure that allows reading/writing modem registers and provides initialization of parameters, reading of status, and control of modem operation. The modem can ask for real time response through the interrupt request structure, and four GPIO signals allow control of the modem reset and monitoring of some real time status.

This chapter presents information addressing application and operation of the node from a system level. The areas considered here are also covered in greater detail in the following sections of the book. The book is organized such that the first three chapters present the top-level view of the MC1321x device and the following chapters present individual functions in detailed descriptions.

### 3.2 Power Connections

The MC1321x power connections at the SiP level are listed in [Table 3-1](#).

**Table 3-1. Power Pin Descriptions**

Pin #	Pin Name	Type	Description	Functionality
45	VDD	Power Input	MCU main power supply	Decouple to ground.
6	VDDAD	Power Input	MCU power supply to ATD	Decouple to ground.
23	VDDINT	Power Input	Modem digital interface supply	Decouple to ground.
22	VDDD	Power Output	Modem regulated output supply voltage	Decouple to ground.
32	VBATT	Power Input	Modem voltage regulators' input.	Decouple to ground.
33	VDDA	Power Output	Modem analog regulated supply output	Decouple to ground. Connect to directly VDDLO1 and VDDLO2 externally.
31	VDDVCO	Power Output	Modem VCO regulated supply bypass	Decouple to ground.
30	VDDLO1	Power Input	Modem LO1 VDD supply	Connect to VDDA externally.

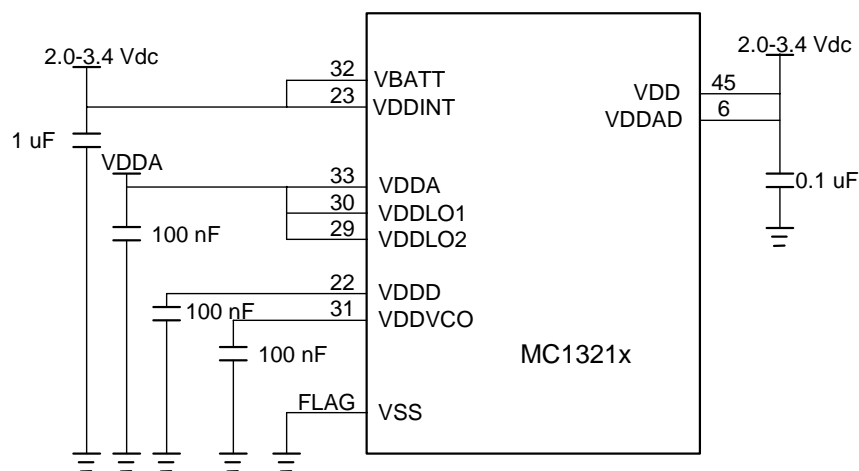
**Table 3-1. Power Pin Descriptions (continued)**

Pin #	Pin Name	Type	Description	Functionality
29	VDDL02	Power Input	Modem LO2 VDD supply	Connect to VDDA externally.
FLAG	VSS	Power input	External package flag. Common VSS	Connect to ground.

When designing power to the MC1321x SiP, the following points need to be considered:

- The LNA SiP package has a single common ground flag (VSS).
- The MCU has a single common power pin VDD except for the VDDAD power supply to the ATD module.
- For the modem there are two primary power inputs which include VBATT for modem power and VDDINT for digital interface power.
- For logic level compatibility between the chips, VDD, VBATT, and VDDINT must be connected to a common source supply of 2.0 - 3.4 VDC.
- VDDAD analog supply to the MCU ATD is also normally wired to the common source supply.
- Modem input VBATT feeds the common supply to the analog and digital circuitry regulators. The analog regulator output VDDA is provided both for bypassing and to supply VDDL01 and VDDL02 which are the power rails for the local oscillators.
- Modem output VDDVCO is provided to allow separate bypass of the modem radio VCO regulated supply.

Power supply connections are shown in [Figure 3-1](#).



**Figure 3-1. MC1321x Power Supply Connections**

**NOTE**

There are separate bypass capacitors on VDDA, VDDD, and VDDVCO. In some RF circuitry configurations, VDDA may also need to be DC-coupled to the radio PA outputs. Depending on the application, the VREFH high reference voltage for the ATD module is commonly also tied to the 2.0-3.4 Vdc common supply.

### 3.3 Test Pin SM

Input SM is a test pin that must be grounded for normal operation.

### 3.4 Internal Functional Interconnects and System Reset

The MCU provides control for the 802.15.4 modem. The required interconnects between the devices are routed onboard the SiP. In addition, the signals are brought out to external pads primarily for use as test points. These signals can be useful when writing and debugging software. These interconnects are listed in [Table 3-2](#).

**Table 3-2. Internal Functional Interconnects**

Pin #	MCU Signal	Modem Signal	Description
43	PTE6	GPIO2	Modem GPIO2 output acts as “CRC Valid” status indicator for Stream Data Mode to MCU.
44	PTE7	GPIO1	Modem GPIO1 output acts as “Out of Idle” status indicator for Stream Data Mode to MCU.
46	PTD0	ATTN	MCU Port D Bit 0 drives the attention ( $\overline{\text{ATTN}}$ ) input of the modem to wake modem from Hibernate or Doze Mode.
65	PTE5/SPSCK1	SPICLK	MCU SPI master SPI clock output drives modem SPICLK slave clock input.
66	PTE4/MOSI1	MOSI	MCU SPI master MOSI output drives modem slave MOSI input
67	PTE3/MISO1	MISO	Modem SPI slave MISO output drives MCU master MISO input
68	PTE2/ $\overline{\text{SS1}}$	CE	MCU SPI master SS output drives modem slave $\overline{\text{CE}}$ input
69	IRQ	M_IRQ	Modem interrupt request $\overline{\text{M\_IRQ}}$ output drives MCU IRQ input
70	PTD1	RXTXEN	MCU Port D Bit 1 drives the RXTXEN input to the modem to enable TX or RX or CCA operations.
71	PTD3	M_RST	MCU Port D Bit 3 drives the reset $\overline{\text{M\_RST}}$ input to the modem.

#### NOTE

To use the MCU and modem signals as described in [Table 3-2](#), the MCU needs to be programmed appropriately for the stated function.

The internal interconnects are shown graphically and grouped functionally in [Figure 3-2](#). The MCU reset input  $\overline{\text{RESET}}$  is also shown. System reset, control and monitoring of the modem, and SPI communication are all concerned with these signals.

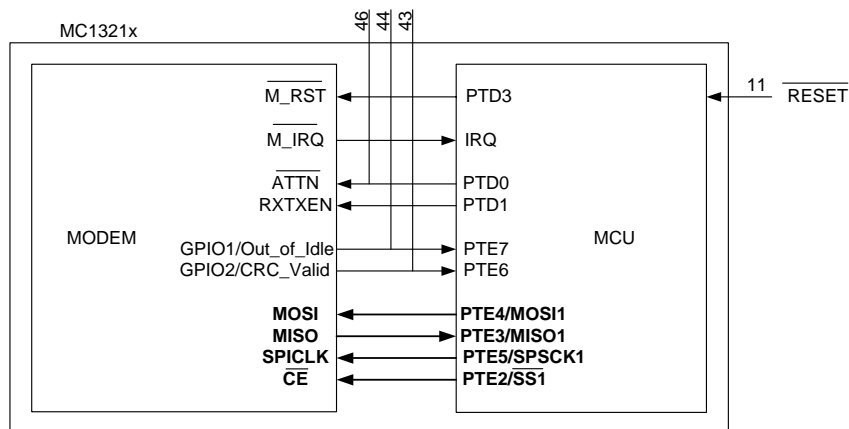


Figure 3-2. MC1321x Internal Functional Interconnects

### 3.4.1 System Reset

The MC1321x system does not have a master input that resets the entire SiP. Instead, the MCU reset input  $\overline{\text{RESET}}$  is the overriding reset and the reset input to the modem  $\overline{\text{M\_RST}}$  must be controlled by the MCU (in Run Mode).

#### 3.4.1.1 MCU Reset Input $\overline{\text{RESET}}$

$\overline{\text{RESET}}$  is an active low dedicated pin with a pull-up device built in. It has input hysteresis, a high current output driver, and no output slew rate control. Internal power-on reset and low-voltage reset circuitry typically make external reset circuitry unnecessary. This pin is normally connected to the standard 6-pin background debug connector so a development system can directly reset the MCU system. If desired, a manual external reset can be added by supplying a simple switch to ground (pull  $\overline{\text{RESET}}$  pin low to force a reset).

Whenever any reset is initiated (whether from an external signal or from an internal system), the reset pin is driven low for approximately 34 cycles of  $f_{\text{Self\_reset}}$  (which is the approximate 8 MHz startup clock), released, and sampled again approximately 38 cycles of  $f_{\text{Self\_reset}}$  later. If reset was caused by an internal source such as low-voltage reset or watchdog time-out, the circuitry expects the reset pin sample to return a logic 1. If the pin is still low at this sample point, the reset is assumed to be from an external source. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the MCU system control Reset Status Register (SRS).

Never connect any significant capacitance to the reset pin because that would interfere with the circuit and sequence that detects the source of reset. If an external capacitance prevents the reset pin from rising to a valid logic 1 before the reset sample point, all resets will appear to be external resets.

MCU reset operation and control is detailed in [Chapter 12, “MCU Resets, Interrupts, and System Configuration”](#).

### 3.4.1.2 Modem Reset

The modem active low reset input  $\overline{M\_RST}$  is driven onboard the SiP from MCU GPIO pin PTD3. In the interest of lowest power, there is no pull-up resistor on input  $\overline{M\_RST}$ . Output PTD3 has a software controlled pull-up resistor, however, it would typically not be used because the modem can be held in hardware reset by the MCU for extended periods of time for lowest power applications.

From a reset condition, the MCU initiates PTD3 as a high-impedance input with the internal pull-up disabled. As a result, the modem reset input will be floating and the modem will not be held in reset. As part of the MCU initialization, PTD3 must be programmed as an output and then driven low to reset the modem. The  $\overline{M\_RST}$  input is asynchronous and needs to be held low for only a short period.

In the reset condition, the modem is totally powered down and no clocks are available. After  $\overline{M\_RST}$  is released, the modem will power up, initialize, and go to its idle condition within 10 - 25 milliseconds, and in turn, this causes an ATTN interrupt request and allows CLKO to start toggling at 32.768+ kHz (both of which are default conditions). The ATTN hardware interrupt request is normally caused by asserting modem signal  $\overline{ATTN}$ , however, coming out of reset the ATTN status bit is set and the ATTN interrupt request mask is set.

Once the interrupt request is seen by the MCU, the MCU can assume the modem is alive and ready for programming via the SPI bus.

Modem reset operation and control is detailed in [Chapter 9, “Modem Miscellaneous Functions”](#).

### 3.4.2 Modem Interrupt Request to MCU

The modem interrupt request  $\overline{M\_IRQ}$  is an active low open drain output that is asserted when an interrupt request is pending. The signal is released to high by reading the modem IRQ\_Status register via a SPI transaction.  $\overline{M\_IRQ}$  has a programmable pull-up resistor and it also can be programmed for drive strength.  $\overline{M\_IRQ}$  is covered in detail in [Section 8.1.2, “Output Pin IRQ”](#).

The modem interrupt request is tied to the MCU external interrupt input IRQ. The MCU IRQ must be programmed as an external interrupt and can be enabled for edge-only or edge-and-level events. The MCU IRQ also has a programmable pull-up. IRQ is covered in detail in [Section 8.1, “Modem Interrupts”](#).

In the MC1321x the IRQ must be programmed for falling edge or edge/low-level sensitivity. Also, the MCU pullup should be selected as a pullup is required for the open drain modem output. On the modem side, the maximum drive strength is suggested as this will give fastest performance for the interrupt fall time.

### 3.4.3 SPI Command Channel

The MCU SPI Module is dedicated to the modem SPI interface. The modem is a slave only and the MCU SPI must be programmed and used as a master only. Further, the SPI performance is limited by the modem constraints of 8 MHz maximum SPI clock frequency, and use of the MCU SPI must be programmed to meet the modem SPI protocol. The SPI bus connections are:

- MCU PTE4/MOSI1 output drives modem MOSI.
- Modem MISO output drives MCU PTE3/MISO1.

- MCU PTE5/SPSCK1 output drives modem SPICLK.
- MCU PTE2/ $\overline{SS1}$  output drives modem  $\overline{CE}$ .

The use and programming of the SPI command channel is described in [Chapter 4, “MC1321x Serial Peripheral Interface \(SPI\)”](#).

### 3.4.4 Modem Control Signals

The modem requires two additional input signals that are controlled by the MCU GPIO:

- $\overline{ATTN}$  input controlled by PTD0 (pin 47) - The  $\overline{ATTN}$  is an active low attention signal that is used wake the modem from Hibernate or Doze Mode. The MCU GPIO PTD0 must be programmed as an output and controls this input
- RXTXEN input is controlled by PTD1 - The RXTXEN is an active high input used to enable transmit, receive, and CCA operations in the modem. The MCU GPIO PTD1 must be programmed as an output and controls this input

### 3.4.5 Modem Status Signals

The modem has two signals that provide real-time status to the MCU:

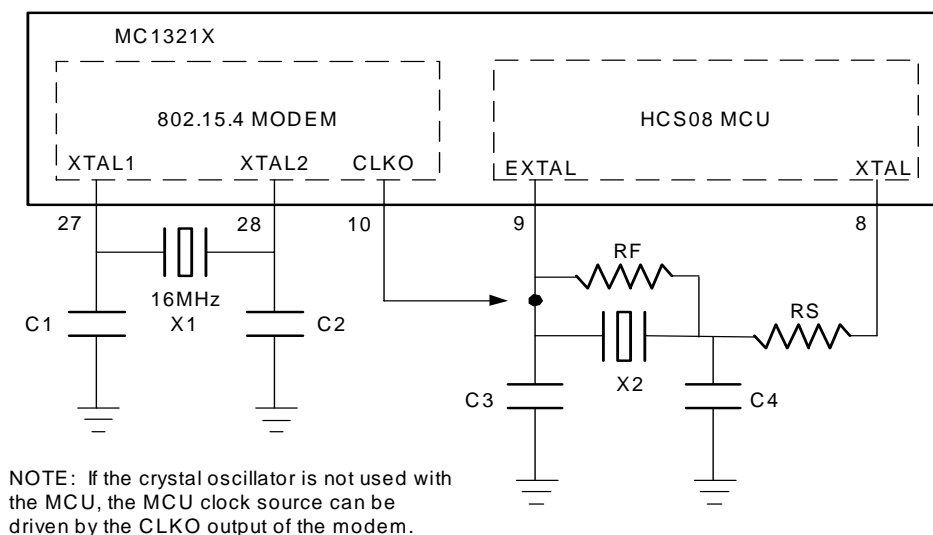
- GPIO1/Out\_of\_Idle output drives PTE7 (pin 44) - The modem GPIO1 signal can optionally be programmed as an “out-of-idle” indicator for monitoring RX, TX, or CCA operation. The MCU GPIO PTE7 must be programmed as an input to monitor this signal. The modem GPIO1 signal can also be used as a general purpose IO as long as there is no conflict with the MCU GPIO PTE7
- GPIO2/CRC\_Valid output drives PTE6 (pin 43) - The modem GPIO2 signal can optionally be programmed as an “CRC valid” indicator for monitoring an RX operation. The MCU GPIO PTE6 must be programmed as an input to monitor this signal. The modem GPIO2 signal can also be used as a general purpose IO as long as there is no conflict with the MCU GPIO PTE6

## 3.5 Clock Sources

The MC321x device allows for a wide array of system clock configurations:

- Pins are provided for a separate external clock source for the CPU. The external clock source can be derived from a crystal oscillator or from an external clock source
- Pins are provided for a 16 MHz crystal for the modem clock source
- The modem crystal oscillator frequency can be trimmed through programming to maintain tight tolerances required by the 802.15.4 Standard
- The modem provides a CLKO programmable frequency clock output that can be used as an external source to the CPU. As a result, a single crystal system clock solution is possible
- Out of reset, the MCU uses an internally generated clock (approximately 8-MHz) for start-up. This allows recovery from stop or reset without a long crystal start-up delay
- The MCU contains an internal clock generator (which can be trimmed) that can be used to run the MCU for low power operation

The MC321x clock connections are shown in [Figure 3-3](#).



**Figure 3-3. MC1321x Clock Connections**

### 3.5.1 Modem Oscillator

The modem oscillator source must always be present and an external crystal is used to implement the oscillator. The source frequency must be 16 MHz with a total accuracy of  $\pm 40$  ppm or greater as required by the 802.15.4 Standard. A detailed discussion of required crystal characteristics is in [Section 9.3.1](#), “Crystal Requirements”.

In [Figure 3-3](#) crystal X1 and capacitors C1 and C2 form the modem crystal oscillator circuit. An onboard feedback resistor of approximately 1 MOhm (not shown) between input XTAL1 and output XTAL2 provides DC biasing for the oscillator buffer. An important parameter for the 16 MHz crystal X1 is a load capacitance of  $<9$  pF. The oscillator needs to see a balanced load capacitance at each terminal of about 18pF. As a result, the sum of the stray capacitance of the pcb board, device pin (XTAL1 or XTAL2), and load capacitor (C1 or C2) at each terminal must equal about 18 pF. C1 and C2 are typically values of 6-9 pF. Higher values can load the crystal buffer and cause oscillator start-up problems.

As described in [Section 9.3.2](#), “Crystal Trim Operation”, the MC1321x crystal oscillator frequency can be trimmed by programming modem CLKO\_Ctl Register 0A, Bits 15-8 (xtal\_trim[7:0]). The trimming procedure varies the frequency by a few hertz per step, depending on the type of crystal. As xtal\_trim[7:0] is increased, the frequency is decreased. This feature is useful for factory calibration of the crystal frequency to set the accuracy for the radio as required by the 802.15.4 Standard.

### 3.5.2 Modem CLKO Clock Source Output

The modem provides a programmable clock source output CLKO. The CLKO output frequency can be varied from 16.393+ kHz to 16 Mhz. The primary uses of CLKO include:

- MCU clock source - for lowest cost operation, a separate crystal is not required for the MCU. CLKO can drive input pin EXTAL to drive the MCU as an external clock source. This is covered in detail in [Chapter 14](#), “MCU Internal Clock Generator (ICG)”.

- A monitor point for the modem clock frequency - the CLKO is a buffered output that can be used to observe the modem frequency for calibration and design purposes. Modem oscillator pins XTAL1 and XTAL2 cannot be used for monitoring the oscillator frequency because this will load the circuit and cause either a frequency error or shutdown the oscillator.

The CLKO output frequency is controlled by programming the modem 3-bit field `clko_rate[2:0]`, CLKO\_Ctl Register 0A, Bits 2-0. Default frequency is 32.786+ kHz with a field value of `clko_rate[2:0] = 110`. [Table 5-12](#) lists the CLKO frequencies versus `clko_rate[2:0]` program value. The use and programming of CLKO is detailed in [Section 9.4, “Output Clock Pin CLKO”](#).

### 3.5.3 MCU Clock Sources

The MCU has several options for its primary clock source depending on its mode of operation as well as its hardware configuration. A detailed discussion of the MCU ICG and associated clock sources is in [Chapter 14, “MCU Internal Clock Generator \(ICG\)”](#).

#### 3.5.3.1 MCU External Clock Source

As shown in [Figure 3-3](#), the external pins associated with the MCU clock source are output XTAL and input EXTAL. An external clock source can have a frequency as high as 40 MHz with no minimum frequency. The external source must have compatible logic levels and drive input EXTAL. Note that no external components are required for the clock oscillator if an external source is used.

#### 3.5.3.2 MCU External Crystal Oscillator

Another choice for the MCU oscillator is the use of an external crystal or ceramic resonator as shown in [Figure 3-3](#). The MCU oscillator is a Pierce type that can accommodate crystals or resonators in either of two frequency ranges (low range and high range) as shown in [Table 3-3](#). Note that the frequency range must be selected properly by programming the RANGE bit in the MCU ICGC1 register (the default condition is the high range).

External resistor RF provides a bias path for the oscillator to keep the input in its linear range. The typical value for RF is dependent on the frequency range and is shown in [Table 3-3](#) although its value is typically not critical. Resistor RS is typically not used and is only required if the crystal or resonator needs protection from the power of the oscillator output.

**Table 3-3. MCU Oscillator Components**

Characteristic	Symbol	Min	Typ <sup>1</sup>	Max	Unit
Load capacitors	C <sub>3</sub> C <sub>4</sub>	<sup>2</sup>			
Feedback resistor Low range (32 kHz - 100 kHz) High range (1 MHz - 16 MHz)	R <sub>F</sub>		10 1		MΩ MΩ
Series Resistor	R <sub>S</sub>		0		Ω

<sup>1</sup> Data in Typical column was characterized at 3.0 V, 25°C or is typical recommended value.

<sup>2</sup> See crystal or resonator manufacturer’s recommendation.



Load capacitors C3 and C4 are typically in the 5 pF to 25 pF range and are determined by the requirements of the specific crystal or resonator. The manufacturer will specify the load capacitance and the capacitors selected for C3 and C4 are used to match the stray circuit capacitance (the pin and PCB capacitance) to the required load capacitance. As a first-order approximation, use 10 pF as an estimate of the combined stray capacitance for a pin plus its PCB capacitance. The load capacitance is the series equivalent of  $(C3 + C_{\text{extalstray}})$  and  $(C4 + C_{\text{extalstray}})$ , where  $C_{\text{extalstray}} = \text{pin capacitance} + \text{trace capacitance}$ . If C3 and C4 get too large the frequency can be affected or the oscillator may not start reliably.

### 3.5.3.3 MCU Internal Clock Sources

Two internal MCU sources to provide system clock are provided such that the external clock sources are not used at MCU start-up and may not be required for normal operation (see [Chapter 14, “MCU Internal Clock Generator \(ICG\)”](#)). The two internal sources derive from an Internal Reference Generator, and one or the other is engaged when the ICG is in Self-Clocked Mode (SCM):

- Approximate 8 MHz source - This source is not exact and is used to start the MCU from a reset or stop condition in which the clocks had been stopped. This clock could be used as the system source, but is not the best choice for accuracy and is not typically used other than at start-up.
- Internal reference clock (which can be trimmed), typically 243 kHz - The low frequency internal reference is specified to be 243 kHz +/- 25% at start-up. However, this source can be trimmed to +/- 0.2% resolution with +/- 0.5% across voltage. This internal reference can also be used with the MCU frequency locked loop (FLL) to generate higher system clock speeds without need for an external reference. This source provides a low power, low cost source with reasonable accuracy.

In addition to internal system clock sources, there is a third oscillator that is used only for a real-time interrupt (RTI). RTI can be used to return from a stop condition while the MCU is in a low power mode and can use an external clock or the internal RTI oscillator. The internal oscillator has a very low frequency accuracy (a ratio from approximately 1 to 1.8), however, it has the lowest power. As a result, the internal RTI oscillator can be used to determine long delays in non-critical applications where the battery life is paramount compared to period accuracy. The Real-Time Interrupt is described in [Section 12.7.6, “System Real-Time Interrupt Status and Control Register \(SRTISC\)”](#).

### 3.5.4 System Clock Configurations

Because of the multiple clock configurations of the MCU, the availability of external clock source pins of both the modem and the MCU, and the CLKO output from the modem, there are a number of variations for MC1321x system clock configurations. Key considerations for any system clock configuration are:

- The modem 16 MHz source (typically the crystal oscillator) must always be present. The crystal has special requirements and the reference frequency must meet the 802.15.4 Standard.
- Battery-operated application requirements for low power can impact the choices for MCU clock source.
- The system clock configuration will impact system initialization procedures.
- Software requirements can impact MCU processor and bus speed. The user must be aware of the performance requirements for the MCU. The CPU clock is always 2X the internal bus speed, and the application software such as the Freescale BeeStack and/or the 802.15.4 MAC often require

either an 8 MHz or 16 MHz bus rate. The implication is that the MCU clock source must be capable of generating these bus rates with or without the MCU FLL.

As far as external connections are concerned, there are three possibilities which are covered in the following sections.

### 3.5.4.1 Single crystal with CLKO driving MCU EXTAL input

The single crystal (modem crystal) with CLKO driving the MCU EXTAL input (external clock) is the most common configuration for low cost and excellent frequency accuracy. The CLKO frequency is programmable from 16.393+ kHz to 16 Mhz and drives the MCU external source. Using the onboard MCU FLL, the required 32 MHz or 16 MHz CPU clock can be generated.

In this configuration, clock start-up from a reset condition involves:

- MCU reset is released and MCU starts on internal 8 MHz clock
- Initialization software must reset and then release reset to the modem (MCU still running on start-up clock)
- Wait for modem start-up interrupt request (approximately 25 msec). CLKO default is active with a frequency of 32.786+ kHz
- Program CLKO to a different frequency (if desired)
- Wait for the CLKO source plus FLL to lock, and then switch MCU clock to external source

Additional considerations for this mode of operation include:

- If the modem is forced to an Off condition (such as a sleep mode), there is a 25 msec wait for the modem CLKO to start from the off condition after M\_RST is released
- If the MCU puts the modem in a low power mode, keeping the CLKO alive is a higher power option
- The internal reference for the MCU can still be used. Disable the CLKO output on the modem and program the MCU clock for internal operation
- If an accurate period is required for longer time delays (such as a beacon period), keeping CLKO alive for very long periods is not the lowest power option

### 3.5.4.2 Single crystal with MCU Using Internal Clock Only

The single crystal (modem crystal) with the MCU using internal clock only has no real advantage over using the CLKO output. The CLKO output can be disabled to save power (EXTAL is grounded), and using the onboard FLL, the required 32 MHz or 16 MHz CPU clock can be generated from the 243 kHz internal reference.

In this configuration, clock start-up from a reset condition involves:

- MCU reset is released and MCU starts on internal 8 MHz clock
- Initialization software must assert reset and then release reset to the modem (MCU still running on start-up clock)
- While waiting for modem start-up interrupt request (approximately 25 msec), the MCU clock can be switched to the internal source

- Program modem to disable CLKO.

Additional considerations for this mode of operation include:

- This mode of operation is also possible with CLKO tied to EXTAL
- The MCU does not have a high accuracy time base when using the internal reference

### 3.5.4.3 Dual Crystal Operation

The modem crystal can be augmented by the use of a second crystal on the MCU. The typical application would use a 32.768 kHz watch crystal that would allow an accurate time base in the MCU for long power down delays. The obvious disadvantage of this configuration is additional cost.

In this configuration, clock start-up from a reset condition involves:

- MCU reset is released and MCU starts on internal 8 MHz clock
- Initialization software must assert reset and then release reset to the modem (MCU still running on start-up clock)
- While waiting for modem start-up interrupt request (approximately 25 msec), the MCU clock can be switched to the external source after the FLL is locked
- Program modem to disable CLKO

Additional considerations for this mode of operation include:

- This mode allows CLKO to be disabled for lower power in the modem
- The second crystal is justified when accurate power down time periods are required. The external clock with the 32.768 kHz crystal allows an accurate 1 second time tick for RTI at very low power. Although power is not as low as the RTI internal oscillator, the time tick now has crystal accuracy.

## 3.6 MC1321x Transceiver Initialization

For latest devices where the transceiver SPI register Chip\_ID Register 0x2C reads as 0x6800, the transceiver must have over-write values written to SPI registers 0x31 and 0x34. See [Section 5.1, “Overview”](#) for details. Previous valid ID values are 0x6000 and 0x6400.

## 3.7 MCU Background/Mode Select (PTG0/BKGD/MS)

The Background/Mode Select (BKGD/MS) shares its function with the MCU I/O port pin PTG0. While in reset, the pin functions as a mode select pin. Immediately after  $\overline{\text{RESET}}$  rises, the pin functions as the background pin and can be used for background debug communication. While functioning as a Background/Mode Select pin, the pin includes an internal pull-up device, input hysteresis, a standard output driver, and no output slew rate control. When used as an I/O port (PTG0) the pin is limited to output only.

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

The BKGD pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCUs BDC clock per bit time. The target MCUs BDC clock could be as fast as the bus clock rate, so there should never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speed-up pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pullup device play almost no role in determining rise and fall times on the BKGD pin.

## 3.8 MC1321x GPIO (Mixed I/O from Modem and MCU)

The MC1321x SiP supports a total of 41 GPIO pins that originate from both the modem and the MCU:

- Five package pins are GPIO from the modem (GPIO3-GPIO7) and can be used as inputs or outputs
- Two package pins can be used as GPIO from either the MCU or the modem because they are connected internally between the MCU and modem. However, they are more commonly used as modem status bits to the MCU
  - GPIO1/Out\_of\_Idle output drives PTE7 (pin 44) - The modem GPIO1 signal can optionally be programmed as an “out-of-idle” indicator for monitoring RX, TX, or CCA operation
  - GPIO2/CRC\_Valid output drives PTE6 (pin 43) - The modem GPIO2 signal can optionally be programmed as an “CRC valid” indicator for monitoring an RX operation
- Thirty-four MAXIMUM additional MCU GPIO are available for use in the SiP package. The actual MCU hardware supports a total of 56 GPIO signals in 7 ports, however, due to package limitations and internal interconnects only the stated 34 can be used with the following limitations
  - If a crystal is used with the MCU, PTG1/XTAL and PTG2/EXTAL pins cannot be used as GPIO. These are available only using the internal clock reference
  - If CLKO is used to drive the external clock source to the MCU, EXTAL is not available as GPIO
  - PTG0/BKGD/MS is not commonly used as GPIO because it is dedicated to the MCU debug port (BDM)
  - In the most common clock configuration (CLKO driving EXTAL and PTG0/BKGD/MS used with the debug port), there are 32 usable MCU GPIO

### 3.8.1 MCU GPIO Characteristics

The internal MCU GPIO hardware consists of 8 ports with 7 signals per port for a total of 56 signals (not all are available on the package). There are 55 GPIO and one output only (PTG0\_BKGD\_MS).

Immediately after MCU reset, all 55 of the GPIO pins are configured as high-impedance general-purpose inputs with internal pull-up devices disabled.

## NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should either enable on-chip pullup devices or change the direction of unused pins to outputs (programmed low) so the pins do not float. Outputs programmed low is the preferred option for lowest power. This includes signals not pinned-out on the package.

Many of these pins share functionality with MCU peripheral functions. Also, the pins internally connected to the modem are dedicated to SPI communication, monitoring modem status, or modem control. For all these cases the functionality of the GPIO are controlled by the applications program and direct programming of the MCU peripherals.

For information about controlling these pins as general-purpose I/O pins, see [Chapter 13, “MCU Parallel Input/Output”](#). For information about how and when on-chip peripheral systems use these pins, refer to the appropriate section from [Table 3-4](#). Not all of the pins are available for external use.

**Table 3-4. MC1321x Pin Sharing References**

Port Pins	Alternate Function	Reference <sup>1</sup>
PTA7–PTA0	KBI1P7–KBI1P0	<a href="#">Chapter 16, “MCU Keyboard Interrupt (KBI)”</a>
PTB7–PTB0	AD1P7–AD1P0	<a href="#">Chapter 20, “Analog to Digital (ATD) Module”</a>
PTC7–PTC4	—	<a href="#">Chapter 13, “MCU Parallel Input/Output”</a>
PTC3–PTC2	SCL1–SDA1	<a href="#">Chapter 19, “Inter-Integrated Circuit (IIC)”</a>
PTC1–PTC0	RXD2–TXD2	<a href="#">Chapter 18, “MCU Serial Communications Interface (SCI)”</a>
PTD7–PTD3	TPM2CH4– TPM2CH0	<a href="#">Chapter 17, “MCU Timer/PWM (TPM Module)”</a>
PTD2–PTD0	TPM1CH2– TPM1CH0	<a href="#">Chapter 17, “MCU Timer/PWM (TPM Module)”</a>
PTE7–PTE6	—	<a href="#">Chapter 13, “MCU Parallel Input/Output”</a>
PTE5 PTE4 PTE3 PTE2	SPSCK1 MISO1 MOSI1 SS1	<a href="#">Chapter 4, “MC1321x Serial Peripheral Interface (SPI)”</a>
PTE1–PTE0	RXD1–TXD1	<a href="#">Chapter 18, “MCU Serial Communications Interface (SCI)”</a>
PTF7–PTF0	—	<a href="#">Chapter 13, “MCU Parallel Input/Output”</a>
PTG7–PTG3	—	<a href="#">Chapter 13, “MCU Parallel Input/Output”</a>
PTG2–PTG1	EXTAL–XTAL	<a href="#">Chapter 14, “MCU Internal Clock Generator (ICG)”</a>
PTG0	BKGD/MS	<a href="#">Chapter 21, “Development Support”</a>
GPIO1–GPIO2	Out_of_Idle, CRC_Valid	<a href="#">Chapter 9, “Modem Miscellaneous Functions”</a>
GPIO3–GPIO7	—	<a href="#">Chapter 9, “Modem Miscellaneous Functions”</a>

<sup>1</sup> See this section for information about modules that share these pins.

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. See [Chapter 13, "MCU Parallel Input/Output"](#) for details.

Pullup enable bits for each input pin control whether on-chip pullup devices are enabled whenever the pin is acting as an input even if it is being controlled by an on-chip peripheral module. When the PTA7–PTA4 pins are controlled by the KBI module and are configured for rising-edge/high-level sensitivity, the pullup enable control bits enable pull-down devices rather than pullup devices. Similarly, when IRQ is configured as the IRQ input and is set to detect rising edges, the pullup enable control bit enables a pull-down device rather than a pullup device.

### 3.8.2 Modem GPIO Characteristics

The modem GPIO hardware consists of seven (7) signals total (GPIO1-GPIO7). Immediately after reset, all the GPIO pins are configured as high-impedance general-purpose inputs. There are no internal pullup devices on these pins.

#### NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should change the direction of unused pins to outputs (programmed low) so the pins do not float. Outputs programmed low is the preferred option for lowest power.

GPIO1 and GPIO2 share functionality with MCU GPIO. These pins are internally connected to two MCU GPIO that can monitor modem status. The alternate functionality of the GPIO1-GPIO2 are controlled by the applications program and use of these pins is described in [Chapter 9, "Modem Miscellaneous Functions"](#)

The functionality of the modem GPIO is controlled by programming of the modem SPI registers via the SPI interface. For information about controlling all these pins as general-purpose I/O pins, see [Chapter 9, "Modem Miscellaneous Functions"](#).

## 3.9 MC1321x Digital Signal Properties Summary

### 3.9.1 Signal Properties Summary

Table 3-5 summarizes digital I/O pin characteristics. These characteristics are determined by the way the common pin interfaces are hard-wired to internal circuits.

**Table 3-5. MC1321x Digital Signal Properties**

Pin Name	Dir	High Current Pin	Output Slew <sup>1</sup>	Pull-Up <sup>2</sup>	Comments
<b>MCU SIGNALS</b>					
$\overline{\text{RESET}}$	I/O	Y	N	Y	Pin contains integrated pullup.
IRQ	I	—	—	Y	Enable bit IRQPE must be set to enable IRQ function. IRQ does not have a clamp diode to $V_{DD}$ . IRQ should not be driven above $V_{DD}$ . Pull-up/pull-down active when IRQ pin function enabled. Pull-up forced on when IRQ enabled for falling edges; pull-down forced on when IRQ enabled for rising edges. Internal connection
PTA0/KBI1P0	I/O	N	SWC	SWC	
PTA1/KBI1P1	I/O	N	SWC	SWC	
PTA2/KBI1P2	I/O	N	SWC	SWC	
PTA3/KBI1P3	I/O	N	SWC	SWC	
PTA4/KBI1P4	I/O	N	SWC	SWC	Pullup/pulldown active when KBI pin function enabled. Pullup forced on when KBI1Px enabled for falling edges; pulldown forced on when KBI1Px enabled for rising edges.
PTA5/KBI1P5	I/O	N	SWC	SWC	
PTA6/KBI1P6	I/O	N	SWC	SWC	
PTA7/KBI1P7	I/O	N	SWC	SWC	
PTB0/AD1P0	I/O	N	SWC	SWC	
PTB1/AD1P1	I/O	N	SWC	SWC	
PTB2/AD1P2	I/O	N	SWC	SWC	
PTB3/AD1P3	I/O	N	SWC	SWC	
PTB4/AD1P4	I/O	N	SWC	SWC	
PTB5/AD1P5	I/O	N	SWC	SWC	
PTB6/AD1P6	I/O	N	SWC	SWC	
PTB7/AD1P7	I/O	N	SWC	SWC	
PTC0/TxD2	I/O	Y	SWC	SWC	When pin is configured for SCI function, pin is configured for partial output drive.
PTC1/RxD2	I/O	Y	SWC	SWC	
PTC2/SDA1	I/O	Y	SWC	SWC	

Table 3-5. MC1321x Digital Signal Properties (continued)

Pin Name	Dir	High Current Pin	Output Slew <sup>1</sup>	Pull-Up <sup>2</sup>	Comments
PTC3/SCL1	I/O	Y	SWC	SWC	
PTC4	I/O	Y	SWC	SWC	
PTC5	I/O	Y	SWC	SWC	
PTC6	I/O	Y	SWC	SWC	
PTC7	I/O	Y	SWC	SWC	
PTD0/TPM1CH0	I/O	N	SWC	SWC	Not available in this package
PTD1/TPM1CH1	I/O	N	SWC	SWC	Not available in this package
PTD2/TPM1CH2	I/O	N	SWC	SWC	
PTD3/TPM2CH0	I/O	N	SWC	SWC	Not available in this package
PTD4/TPM2CH1	I/O	N	SWC	SWC	
PTD5/TPM2CH2	I/O	N	SWC	SWC	
PTD6/TPM2CH3	I/O	N	SWC	SWC	
PTD7/TPM2CH4	I/O	N	SWC	SWC	
PTE0/TXD1	I/O	N	SWC	SWC	
PTE1/RXD1	I/O	N	SWC	SWC	
PTE2/SS1	I/O	N	SWC	SWC	Internal connection
PTE3/MISO1	I/O	N	SWC	SWC	Internal connection
PTE4/MOSI1	I/O	N	SWC	SWC	Internal connection
PTE5/SPSCK1	I/O	N	SWC	SWC	Internal connection
PTE6	I/O	N	SWC	SWC	Internal connection
PTE7	I/O	N	SWC	SWC	Internal connection
PTF0	I/O	Y	SWC	SWC	Not available in this package
PTF1	I/O	Y	SWC	SWC	Not available in this package
PTF2	I/O	Y	SWC	SWC	Not available in this package
PTF3	I/O	Y	SWC	SWC	Not available in this package
PTF4	I/O	Y	SWC	SWC	Not available in this package
PTF5	I/O	Y	SWC	SWC	Not available in this package
PTF6	I/O	Y	SWC	SWC	Not available in this package
PTF7	I/O	Y	SWC	SWC	Not available in this package
PTG0/BKGD/MS	O	N	SWC	SWC	Pullup enabled and slew rate disabled when BDM function enabled.
PTG1/XTAL	I/O	N	SWC	SWC	Pullup and slew rate disabled when XTAL pin function.



**Table 3-5. MC1321x Digital Signal Properties (continued)**

Pin Name	Dir	High Current Pin	Output Slew <sup>1</sup>	Pull-Up <sup>2</sup>	Comments
PTG2/EXTAL	I/O	N	SWC	SWC	Pullup and slew rate disabled when EXTAL pin function.
PTG3	I/O	N	SWC	SWC	Not available in this package
PTG4	I/O	N	SWC	SWC	Not available in this package
PTG5	I/O	N	SWC	SWC	Not available in this package
PTG6	I/O	N	SWC	SWC	Not available in this package
PTG7	I/O	N	SWC	SWC	Not available in this package
<b>MODEM SIGNALS</b>					
IRQB	O	N	SWC	SWC	Internal connection
XTAL1	I	—	—	N	
XTAL2	O	N	N	N	
$\overline{\text{ATTN}}$	I	—	—	N	Internal connection
RXTXEN	I	—	—	N	Internal connection
$\overline{\text{M\_RST}}$	I	—	—	N	Internal connection
CLKO	O	N	SWC	N	
SPICLK	I	—	—	N	
MOSI	I	—	—	N	
MISO	O	N	SWC	N	Off state is SWC
$\overline{\text{CE}}$	I	—	—	N	
GPIO1/Out_of_Idle	I/O	N	SWC	N	Internal connection
GPIO2/CRC_Valid	I/O	N	SWC	N	Internal connection
GPIO3	I/O	N	SWC	N	
GPIO4	I/O	N	SWC	N	
GPIO5	I/O	N	SWC	N	
GPIO6	I/O	N	SWC	N	
GPIO7	I/O	N	SWC	N	

<sup>1</sup> SWC is software controlled slew rate, the register is associated with the respective port.

<sup>2</sup> SWC is software controlled pull-up resistor, the register is associated with the respective port.

### 3.10 Transceiver RF Configurations and External Connections

The MC1321x family radio has features that allow for a flexible as well as low cost RF interface:

- Programmable output power - 0 dBm nominal output power, programmable from -27 dBm to +4 dBm typical
- <-94 dBm (typical) receive sensitivity - At 1% PER, 20-byte packet (well above 802.15.4 Standard of -85 dBm)
- Optional integrated transmit/receive (T/R) switch for low cost operation - With internal PAs and LNA, the internal T/R switch allows a minimal part count radio interface using only a single balun to interface to a single-ended antenna
- Maximum flexibility - There are full differential RF I/O pins for use with the internal T/R switch. Optionally, these pins become the RF\_IN signals and a separate set of full differential PA outputs are also provided. Separate inputs and outputs allow for a variety of RF configurations including external LNA and PA for increased range
- CT\_Bias Output - The CT\_Bias signal provides a switched bias reference for use with the internal T/R switch, and alternatively can be programmed as an antenna switch signal for use with an external antenna switch
- Onboard trim capability for 16 MHz crystal reference oscillator - The 802.15.4 Standard puts a +/- 40 ppm requirement on the carrier frequency. The onboard trim capability of the modem crystal oscillator eliminates need for external variable capacitors and allows for automated production frequency calibration. Also tighter tolerance can produce greater receive sensitivity

#### 3.10.1 RF Interface Pins

Figure 3-4 shows the RF interface pins and the associated analog blocks. Notice that separate PA blocks are associated with RFIN\_x and PAO\_x signal pairs. The RF interface allows both single port differential operation and dual port differential operation.

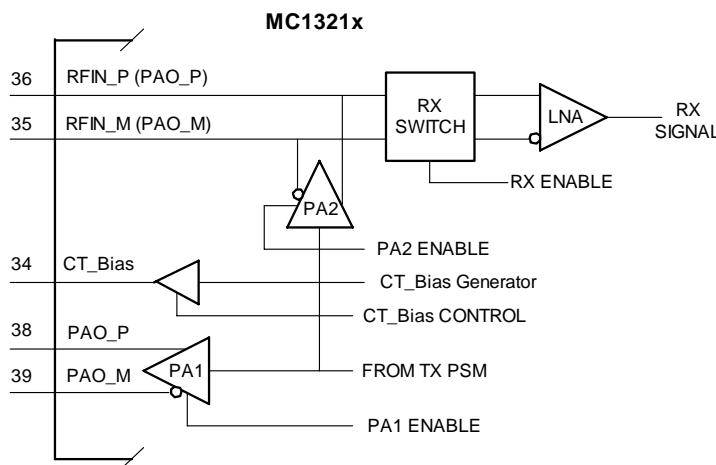


Figure 3-4. RF Interface Pins

### 3.10.1.1 Single Port Operation

The integrated RF switch allows users to operate in a single port configuration. In Single Port Mode, an internal RX switch and separate PA are used and pins RFIN\_P (PAO\_P) and RFIN\_M (PAO\_M) become bidirectional and connect both for TX and RX. When receiving, the RX switch is enabled to the internal LNA and the TX PA1 is disabled. When transmitting, the RX switch is disabled (isolating the LNA) and TX PA1 is enabled. The optional CT\_Bias pin provides a reference or bias voltage which is at VDDA for transmit and is at ground for receive. This signal can be used to provide the proper bias voltage to a balun that converts a single-ended antenna to the differential interface required by the SiP.

Figure 3-5 shows a single port example with a balun. The CT\_Bias is connected to the balun center-tap providing the proper DC bias voltage to the balun depending on RX or TX.

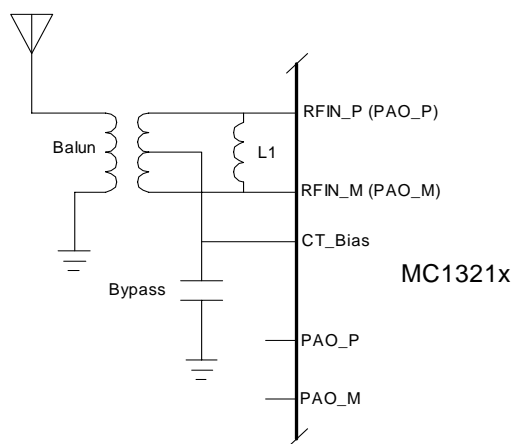


Figure 3-5. Single Port RF Operation with a Balun

### 3.10.1.2 Dual Port Operation

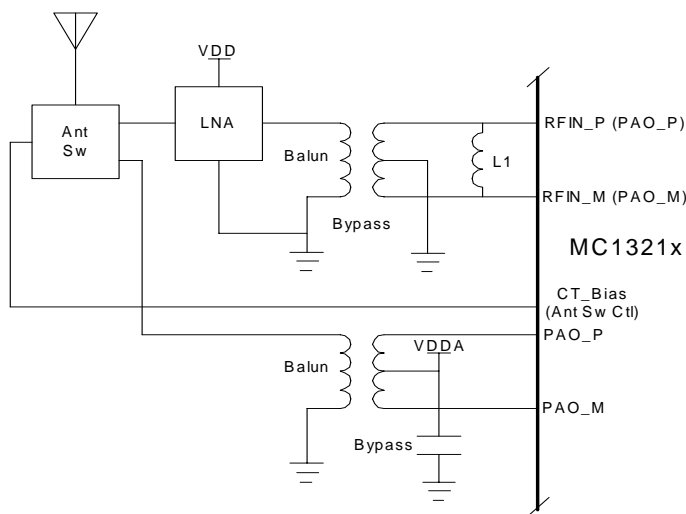
A second set of pins designated PAO\_P and PAO\_N allow operation in a dual port configuration. There are separate paths for transmit and receive with the optional CT\_Bias pin providing a signal that indicates if the radio is in TX or RX Mode which then can be used to drive an external low noise amplifier or power amplifier.

In dual port operation, the RFIN\_P and RFIN\_N are inputs only, the internal RX switch to the LNA is enabled to receive, and the associated TX PA1 stays disabled. Pins PAO\_P and PAO\_N become the differential output pins and the associated TX PA2 is enabled for transmit.

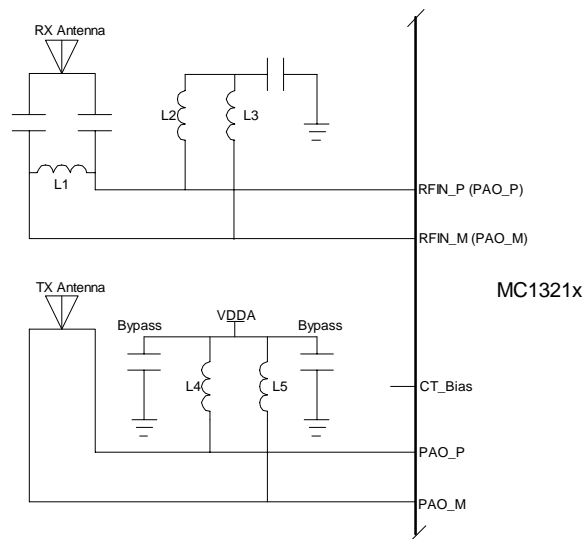
Figure 3-6 shows two dual port configurations. First is a single antenna configuration with an external low noise amplifier (LNA) for greater range. An external antenna switch is used to multiplex the antenna between receive and transmit. An LNA is in the receive path to add gain for greater receive sensitivity. Two external baluns are required to convert the single-ended antenna switch signals to the differential signals required by the radio. Separate RFIN and PAO signals are provided for connection with the baluns, and the CT\_bias signal is programmed to provide the external switch control. The polarity of the external switch control is selectable.

Figure 3-6 also shows a dual antenna configuration where there is a RX antenna and a TX antenna. For the receive side, the RX antenna is ac-coupled to the differential RFIN inputs and these capacitors along with

inductor L1 form a matching network. Inductors L2 and L3 are ac-coupled to ground to form a frequency trap. For the transmit side, the TX antenna is connected to the differential PAO outputs, and inductors L4 and L5 provide DC-biasing to VDDA but are ac isolated. CT\_Bias is not required or used.



Using External Antenna Switch with LNA



Using Dual Antennae

Figure 3-6. Dual Port RF Configuration Examples

### 3.10.2 Controlling RF Modes of Operation

Use of the RF interface pins and RF modes of operation are controlled through several bits of modem Control\_B Register 07. [Figure 3-7](#) shows the model for Register 07 with the RF interface control bits highlighted.

Register 07															0x07	
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	tmr_load	ct_bias_en	ct_bias_inv	RF_switch_mode	miso_hiz_en		clko_doze_en		tx_done_mask	rx_done_mask	use_strm_mode				hib_en	doze_en
TYPE	r/w	r/w	r/w	r/w	r/w		r/w		r/w	r/w	r/w				r/w	r/w
RESET	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
	0x0C00															

**Figure 3-7. Modem Control\_B Register 07 Model**

The RF interface control bits include:

- RF\_switch\_mode (Bit 12) - This bit selects Dual Port Mode versus Single Port Mode
  - The default condition (Bit 12 = 0) is Dual Port Mode where the RF inputs are RFIN\_M and RFIN\_P and the RF outputs are PAO\_M and PAO\_P and operation is as described in [Section 3.10.1.2, “Dual Port Operation”](#). The use of CT\_Bias pin in Dual Port Mode is controlled by Bit 13 and Bit 12
  - When Bit = 1, the Single Port Mode is selected where RFIN\_M (PAO\_M) and RFIN\_P (PAO\_P) become bidirectional pins and operation is as described in [Section 3.10.1.1, “Single Port Operation”](#). The use of CT\_Bias pin in Single Port Mode is controlled by Bit 14 and operation of the radio
- ct\_bias\_en (Bit 14) - This bit is the enable for the CT\_Bias output. When Bit 14 = 0 (default), the CT\_Bias is disabled and stays in a Hi-Z or tri-stated condition. When Bit 14 = 1 the CT\_Bias output is active and its state is controlled by the selected mode (Bit 12), ct\_bias\_inv, and operation of the radio
- ct\_bias\_inv (Bit 13) - This bit only affects the state of CT\_Bias when Dual Port Mode is selected and CT\_Bias is active. The CT\_Bias changes state in Dual Port Mode based on the TX or RX state of the radio. The ct\_bias\_inv bit causes the sense of the active state to change or invert based on Bit 13’s setting. In this manner the user can select the CT\_Bias as a control signal for external components and make the control signal active high or active low

[Table 3-6](#) summarizes the operation of the RF interface control bits.

**Table 3-6. RF Interface Control Bits**

Bit	Designation	Default	Operation
14	ct_bias_en	0	1 = CT_Bias enabled. Output state is defined by <a href="#">Table 3-7</a> . 0 = CT_Bias disabled. Output state is tri-stated.
13	ct_bias_inv	0	The output state of CT_Bias under varying conditions is defined in <a href="#">Table 3-7</a> . This bit only has effect for dual port operation. 1 = CT_Bias inverted. 0 = CT_Bias not inverted
12	RF_switch_mode	0	1= Single Port Mode selected where RF switch is active and RFIN_M and RFIN_P and bidirectional signals. 0 = Dual Port Mode selected where RFIN_M and RFIN_P are inputs only and PAO_P and PAO_N are separate outputs. (This is default operation).

### 3.10.3 RF Control Output CT\_Bias

CT\_Bias is a useful signal for interface with external RF components. It must be enabled via the ct\_bias\_en control bit, and then its state is determined first by the selected RF mode and then by the active state of the radio, i.e., whether a TX or RX operation is active:

- Single Port Operation - In this mode, the CT\_Bias can be used to establish the proper DC bias voltage to a balun depending on the RX state versus TX state as described in [Section 3.10.1.1, “Single Port Operation”](#). Note that in single port operation, the ct\_bias\_inv has no effect and CT\_Bias is at VDDA for TX and is at ground for RX
- Dual Port Operation - In this mode, the CT\_Bias can be used as a control signal to enable an LNA or to determine the direction of an antenna switch as described in [Section 3.10.1.2, “Dual Port Operation”](#). In dual port operation ct\_bias\_inv is used to control the sense of the output control, i.e., CT\_Bias can be active high or active low for TX and vice-versa for RX

[Table 3-7](#) defines the CT\_Bias output state depending on control bits and operation mode of the modem. Note that the output state is also defined for the modem low power states of Idle, Hibernate, and Doze as well as RX and TX operation.

**Table 3-7. CT\_Bias Output vs. Register Settings**

Mode	CT_Bias_en	RF_switch_mode	CT_Bias_inv	CT_Bias
RX	1	1	0	0
RX	1	1	1	0
RX	1	0	0	0
RX	0	X	X	Hi-Z
RX	1	1	0	1
TX	1	1	0	1
TX	1	1	1	1
TX	1	0	0	1

**Table 3-7. CT\_Bias Output vs. Register Settings (continued)**

Mode	CT_Bias_en	RF_switch_mode	CT_Bias_inv	CT_Bias
TX	1	0	1	0
TX	0	X	X	Hi-Z
Idle	1	X	X	0
Idle	0	X	X	Hi-Z
Doze	1	X	X	0
Doze	0	X	X	Hi-Z
Hibernate	1	X	X	0 (Low-Z)
Hibernate	0	X	X	Hi-Z
Off	X	X	X	Unknown

## 3.11 MC1321x Timer Resources

When writing software for the ZigBee specification and/or the 802.15.4 Standard applications, the user should be aware of the set of timer resources available on the MC1321x. The timer resources derive from both the MCU and modem functions and are individually best suited to different types of uses. The functions include two MCU Timer/PWM (TPM) modules, the modem Event Timer block with four timer compare registers, the MCU real time interrupt (RTI) for wake up from a low power mode, and the MCU computer operating properly (COP) watchdog.

### 3.11.1 MCU TPM Modules

There are two MCU TPM modules that are primarily used to implement software timers and peripheral timing based interfaces. The TPM module interface is memory mapped into the MCU address space and has full separate interrupt capability. Each TPM has an independent 16-bit counter with prescaler and modulo features to control frequency and range of a time reference. The time base can be sourced from the bus clock, fixed system clock, or an external pin.

The MCU has hardware to support a 3-channel TPM1 and a separate 5-channel TPM2. However, the package limitations of the SiP allow only TPM1 Channel 2, and TPM2 Channels 1, 2, 3, and 4. TPM1 would be the first choice to implement a “time tick” for a software scheduler because of its limitation of having only one I/O channel.

The input capture, output compare, and pulse width modulation (PWM) capabilities of the TPM allow for a wide range of control applications and peripheral functions. The TPM can be a useful interface into an external sensor function.

The detailed description for the TPM is found in [Chapter 17, “MCU Timer/PWM \(TPM Module\)”](#).

### 3.11.2 Modem Event Timer Block with Four Timer Compare Registers

The MC1321x contains an internal Event Timer block whose clock is derived from a programmable prescaler which is driven by the 16 MHz crystal source. The Event Timer consists of the prescaler and a 24-bit counter which increment whenever the modem crystal clock is operating. The Event Timer provides the following functions:

- The 24-bit counter generates “current” system time - The counter runs continuously when the modem is not in a low power mode and provides a local “current” time. The present value can be read from Current\_Time\_A Register 26, Bits 7-0, and Current\_Time\_B Register 27, Bits 15-0. The counter can be programmed to be clocked at a rate varying from 15+ kHz to 2 MHz. The counter value is used to generate a time stamp for received frames and is also used for comparison to four different comparators that allow up to four different time delays
- Interrupt generation at pre-determined system times - There are four comparator functions that can generate an interrupt to the MCU when the value of the comparator matches the system counter. These function as timers to generate time delays or trigger events and can augment the MCU timers for generating time delays required within the 802.15.4 application software
- Exit from Doze Mode at pre-determined system time - Timer comparator Tmr\_Cmp2 can be used to wake-up the modem from low power Doze Mode
- Latches “timestamp” value during packet reception - With every received frame, there is a timestamp generated and stored in a modem SPI register. The timestamp is taken from the counter when the FLI is received
- Initiates timer-triggered sequences - The modem has three basic active modes of RX, TX, and CCA. Each of these sequences can be timer-triggered to become active after a set delay based on timer comparator Tmr\_Cmp2. This feature can be useful for implementing 802.15.4 application sequences

Operation, loading, and reading of the modem timer registers is done through the SPI interface. [Chapter 6, “Modem Timer Information”](#), describes the modem timer operations in detail.

### 3.11.3 MCU Real-Time Interrupt (RTI)

The MCU has a real-time interrupt capability that can be used to generate periodic interrupts based on a multiple of the source clock period. The RTI is programmable to have a period of 256, 1024, 2048, 4096, 8192, 16384, or 32768 times the source clock period. The RTI clock source can be:

- RTI Internal Clock Source - This source is independent of the ICG and allows lowest power operation. The period of the internal clock is nominally 1 ms, however, it has a very loose tolerance with a spec of 700 -1300  $\mu$ S. The maximum RTI period then is 1.024 second nominal
- External Clock Source - This source uses an external crystal or clock source. For best accuracy and longest RTI period, a 32.768 kHz clock crystal is commonly used which allows a maximum RTI period of exactly 1 second. Use of the external clock requires slightly more power and is available in run, wait, and Stop3 modes

The RTI is covered in detail in [Chapter 12, “MCU Resets, Interrupts, and System Configuration”](#).



The RTI can generate a “time tick” that is useful for a run time software time tick or to wake up the MCU from a low power (Sleep) Mode. Long battery life in many ZigBee and/or 802.15.4 applications is based on long periods of “sleep” or low power modes. The MCU can put the modem in the Off or Hibernate Mode and then go to a low power mode such as a Stop2 or Stop3 Mode. Although the application may require the sleep period to be much longer than 1 second, the MCU can wake up every second, increment a software counter, and then return to sleep if the software counter has not reached the target count.

### 3.11.4 MCU Computer Operating Properly (COP) Watchdog

Another independent MCU timer is the COP timer which is intended to force a system reset when the application fails to execute as expected. To prevent the reset, the software must reset the COP timer (if enabled) periodically before the timer times out. The COP timer use is optional and can be programmed to have a period of  $2^{18}$  or  $2^{13}$  cycles of the bus rate clock. The COP is covered in [Chapter 17, “MCU Timer/PWM \(TPM Module\)”](#).

## 3.12 Low Power Considerations

Many ZigBee and/or 802.15.4 applications such as sensor End Devices are required to be battery operated. As expected, long battery life is highly desirable and is very dependent on application parameters. Over-the-air operation uses modes, i.e., RX, TX, and CCA, where power is highest. As a result, the time between radio operations should be kept at the longest possible period that the application will allow.

When designing low power operation of the MC1321x SiP, consider:

- The modem and the MCU each have several low power options
- The modem is entirely controlled by the MCU; the low power options/combinations will be determined by MCU programming
- The power down control of the modem must be maintained by the MCU in its power down configuration
- All unused port GPIO (including those not pinned-out) must be configured to a known state (preferably output low) to prevent unwanted leakage current.

Lowest power in a system is more than just putting the modem and/or the MCU in a low power mode. The relationship between the functions, the timing between them, and clock management must all be considered. The duty cycle between active operations is also very important as it can impact whether sleep operation or active operation will have the biggest impact over an extended time period.

### 3.12.1 Modem Low Power States

[Table 3-8](#) lists the modem low power states and the modes are covered in detail in [Chapter 7, “Modem Modes of Operation”](#). There are three low power modes available:

- Off - Requires the modem reset  $\overline{M\_RST}$  input to stay asserted low. Lowest possible power and all functions are disabled. Digital I/O are tristated
- Hibernate - Has next lowest power, all hardware blocks deactivated, RAM and SPI register data are retained. Digital I/O retain their state

Doze - Allows use of the Event Timer when active. The Event Timer can be used to cause a timed exit from Doze, and the CLKO output can be kept active in Doze to provide a clock to the MCU. Doze uses considerable more current than Off or Hibernate. RAM and register data are retained and digital I/O retain their state.

**Table 3-8. MC1321x Modem Low Power States**

Mode	Current (typ @ 2.7V)	Advantages	Disadvantages	Comment
Off	0.2 $\mu$ A	Lowest power (leakage only)	Digital outputs tristated. All RAM/register data lost.	$\overline{\text{M\_RST}}$ must remain asserted. All IC functions off.
Hibernate	1.0 $\mu$ A	RAM/register data retained. Digital outputs retain their states.		$\overline{\text{ATTN}}$ or $\overline{\text{M\_RST}}$ used to exit state.
Doze <sup>1</sup>	35 $\mu$ A (no CLKO)	Crystal reference oscillator is on and CLKO can be enabled. Timed exit is possible. RAM/register data retained. Digital outputs retain their states.	Much higher current than Hibernate or Off.	Can exit on $\overline{\text{ATTN}}$ or $\overline{\text{M\_RST}}$ or timed exit is option. CLKO can be kept active as a clock source.
Idle	500 $\mu$ A	Fast transition to RX, TX, CCA	Much higher current than Doze or Hibernate	State from which all TX, RX or CCA is initiated.

<sup>1</sup> CLKO frequency at default value of 37.786 kHz.

Although Idle is not considered a low power state, it is listed in [Table 3-8](#) for comparison. It is also important to remember that all active states of RX, TX and CCA must be initiated from the Idle condition.

### 3.12.2 Special Consideration Where Modem Doze Current Is Higher Than Specified

The Doze current (no CLKO output active) is specified as 35  $\mu$ A (typical) on the data sheet with the programmed CLKO frequency at a default of 32.786 kHz. The Doze current can be considerably higher for certain combinations of higher CLKO frequencies and event timer prescale options. These combinations consist of:

1. CLKO frequency = 16 MHz with prescale select at 5, 6, or 7.
2. CLKO frequency = 8 MHz with prescale select at 6, or 7.
3. CLKO frequency = 4 MHz with prescale select at 7.

All other combinations have no problems. The higher current will not occur every time Doze is enabled. There is no potential harm either to the transceiver or its operation, the Doze current is simply higher.

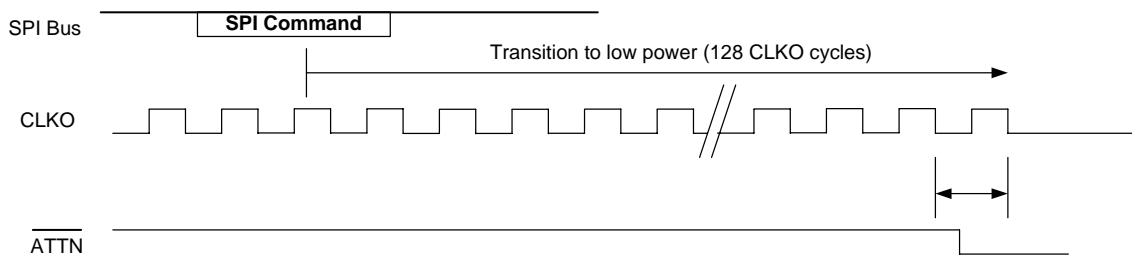
To work around this issue, there are three choices:

1. Accept higher current in Doze mode.
2. Do not use any of the described combinations in Doze mode.
3. If a higher CLKO frequency is desired when using CLKO as an MCU clock source, and the desired prescale select can cause a problem, just before entering Doze mode, program the CLKO frequency to a lower value. Next, use the desired prescale value while in Doze. Finally, after exiting Doze mode, reprogram CLKO to the desired frequency before releasing the MCU clock to the CLKO source

### 3.12.3 Modem Low Power Exit Using $\overline{\text{ATTN}}$

Although a hardware reset will also cause the MC1321x modem to exit its low power state, the standard means to exit is use of  $\overline{\text{ATTN}}$ . The  $\overline{\text{ATTN}}$  input is asserted by a high-to-low signal transition.

- If the transceiver is active (not in Hibernate or Doze), a high-to-low  $\overline{\text{ATTN}}$  signal transition is ignored and has no effect.
- $\overline{\text{ATTN}}$  can be negated (low-to-high transition) at any time and has no effect.
- When the transceiver is fully in Hibernate or Doze,  $\overline{\text{ATTN}}$  may be asserted at any time.
- When the transceiver is entering Hibernate or Doze,  $\overline{\text{ATTN}}$  may be asserted at any time EXCEPT during the final cycle of CLKO (see Figure 3-8) -
  - Background - A SPI command is written to the transceiver to initiate a low power mode. The transceiver continues to run CLKO for 128 clock cycles after the command is recognized; this is true whether or not the external CLKO output is enabled. After completion of the last clock cycle, the transceiver completes the transition to low power
  - $\overline{\text{ATTN}}$  can be asserted any time except for the last CLKO clock cycle. If  $\overline{\text{ATTN}}$  does assert during this period, the transition is ignored and the transceiver continues into low power mode. The expected wake-up will not occur.



**Figure 3-8. Disallowed Timing for  $\overline{\text{ATTN}}$  Transition**

- Preventative strategies -
  - Do not assert  $\overline{\text{ATTN}}$  during the power down sequence - be sure a sufficient delay is provided as to prevent  $\overline{\text{ATTN}}$  going active until the transceiver is fully in low power mode.
  - Cycle  $\overline{\text{ATTN}}$  twice - if it is difficult to be sure  $\overline{\text{ATTN}}$  does not cycle too soon,  $\overline{\text{ATTN}}$  can be asserted more than once to be sure to cause the wake-up. The cycle time between the high-to-low edges should be longer than the cycle time of CLKO during power down.
  - Speed-up CLKO frequency before power down to shorten power down delay - the Freescale MAC and stacks use CLKO at its default 32.786 kHz frequency and the elapsed time for 128 CLKO cycles is ~3.9 ms. This time can be reduced to ~8  $\mu\text{s}$  if CLKO is reprogrammed to 16 MHz just before entering low power mode. This can help in applications where a quick turnaround from low power to normal operation is desired at certain times.

### 3.12.4 MCU Low Power States

Table 3-9 lists the MCU low power states and the modes are covered in detail in [Chapter 10, “MCU Modes of Operation”](#). Also, a suggested application note for in-depth coverage of the MCU low power modes is *MC9S08GB/GT Low-Power Modes, AN2493*. There are three low power modes available:

- Stop1 - This MCU Mode IS NOT USABLE in the MC1321x device. The Stop1 is the off condition for the MCU and does not allow the MCU I/O to retain their programmed states. As a result, there is no means available to force the modem to a known condition on its reset input
- Stop2 - This is a very low power state in which the I/O states are maintained, the RAM contents are maintained, and the RTI can be used to exit with the internal oscillator. This state can maintain a reset condition on the modem. The disadvantages are that the register values are reset so they must be restored before peripherals can be used and interrupts are not serviced, but instead are treated as a reset
- Stop3 - This is still a very low current mode and is the preferred Sleep Mode for the MC1321x. The I/O states are maintained (the modem can be held reset), RAM and register contents are retained, exit can be done with any active interrupt, RTI can be used with the external clock (use and external crystal or the modem CLKO), and interrupts are serviced not treated as a reset, i.e., the normal interrupt vector is used to generate an interrupt service routine. The only real disadvantage is that current is slightly higher than Stop2
- Wait - In the Wait Mode the bus clock source remains active; clocks are disabled to the CPU but peripherals can be clocked. The advantages are that current is reduced from Run Mode (useful for short “wait” periods) and there is no stop recovery time. This state is also useful for low noise operation when sampling the ADC. The real disadvantage is the increased current consumption over the stop modes. Wait is not suggested for node “sleep” operation, but typically is used for waiting for a peripheral to complete an operation

**NOTE**

All unused GPIO (including signals not pinned-out) must be initialized for low power operation. If this caution is not addressed cited low power currents may not be valid.

**Table 3-9. MC1321x MCU Low Power States**

Mode	Current (typ @ 3.0V)	Advantages	Disadvantages	Comment
Stop1	25 nA		NOT USABLE. Digital I/O do not retain their states	NOT USABLE. MCU has no means to insure Off state of modem.
Stop2	550 nA	Very low power. RAM contents maintained. I/O states retained. RTI can be used to wake up MCU w/o external clock.	RTI cannot use external oscillator (poor accuracy). Registers have to be restored after exit. Interrupts not serviced, treated as a reset.	Can exit with MCU reset or wake-up timer (RTI). Can maintain $\overline{M\_RST}$ low state.

**Table 3-9. MC1321x MCU Low Power States (continued)**

Mode	Current (typ @ 3.0V)	Advantages	Disadvantages	Comment
Stop3	675 nA	Very low power. RAM and registers contents maintained. I/O states maintained.	Power not quite as low as Stop2	Best choice for node "sleep" function. Can maintain $\overline{M\_RST}$ low state.
	300 nA (adder for RTI)	RTI is used for wake up with internal osc as clock source	RTI period has low accuracy.	Can exit with RTI (uses internal clock). Adds 300 nA to 675 nA $I_{DD}$ .
	5 $\mu$ A (adder for osc enabled)	RTI is used for wake up with external osc as clock source. Allows use of crystal for clock source for high accuracy	Additional current needed for osc.	Can exit with RTI using crystal source. Adds 5 $\mu$ A to 675 nA Stop3 $I_{DD}$
Wait	560 $\mu$ A (@ $f_{BUS} = 1$ MHz)	Bus clock source remains active. Clocks can be disabled to CPU but peripherals can be clocked. No recovery time. Reduced power versus Run Mode. Reduced noise for ADC readings.	Much higher power than Stop modes. Voltage regulator remains active.	Can be used with Run state for lowest power when using peripherals. Interrupts serviced immediately. Not best choice for node sleep mode.

### 3.12.5 Recovery Times from Low Power Modes

The Modem Mode of operation is controlled by the MCU. The modem may be powered down if it is not in use while the MCU is doing another task or while the whole node is "sleeping", i.e., the MCU is also powered down. Recovery time for both the modem and the MCU are important to system performance and the recovery times are independent of each other.

#### 3.12.5.1 Modem Low Power Mode Recovery Times

Each of the modem recovery times is from the low power condition to the Idle state. The start-up times for the Off and Hibernate conditions are considerably longer due to the start-up of the voltage regulators and clock oscillator. Figure 3-9 shows a simplified state diagram for the low power modes and gives the transition time to Idle:

- Off > Idle (25 ms) - The Off state is released by negating  $\overline{M\_RST}$  high. From that time until the modem asserts a  $\overline{ATTN}$  interrupt and CLKO starts with a default frequency of 32.786+ kHz is 25 ms maximum
- Hibernate > Idle (20 ms) - The Hibernate state is normally released via asserting  $\overline{ATTN}$  low. The startup time at 20 ms maximum is a little quicker than from the Off condition. The modem also asserts a  $\overline{ATTN}$  interrupt (if enabled) and CLKO starts (if enabled) with the value programmed before entering Hibernate
- Doze > Idle ((300 + 1/CLKO)  $\mu$ s) - The Doze state can be released via a timer or asserting  $\overline{ATTN}$  low. The start-up time is considerably less ((300 + 1/CLKO)  $\mu$ s) because the clock oscillator is already running. CLKO can be programmed to run during Doze, and if not, CLKO will start in enabled for normal operation. An  $\overline{ATTN}$  interrupt will be asserted (if enabled) when  $\overline{ATTN}$  is used to exit Doze, or an interrupt will be asserted for exiting Doze Mode via a timer

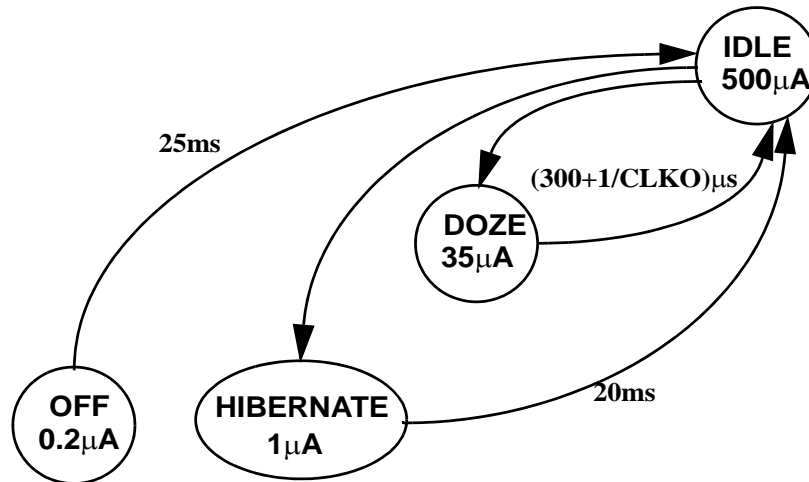


Figure 3-9. MC1321x Modem Low Power Recovery Times

### 3.12.5.2 MCU Low Power Mode Recovery Times

Recovery from MCU low power modes can be quicker than the modem because there is an internal oscillator that has a quick startup time when compared to a crystal oscillator. Because of the many clock options full recovery to Run Mode under programmed conditions can be considerably longer than the initial startup of Run Mode under the internal oscillator. The various low power startup conditions are listed below:

- Stop1 - DO NOT USE
- Stop2 - The Stop2 can be exited via an IRQ from the modem, an MCU reset, or the internal auto walk-up timer with RTI. Once wake-up has occurred, the MCU will start-up as if a power-on reset (POR) has occurred, as a result the system bus clock is driven by an internal approximately 4 MHz clock (SCM Mode) and stop recovery occurs fairly quickly. The delay is for the voltage regulator to reach full regulation and recovery time is approximately 50 µs @ VDD = 3 Vdc and 80 µs @ VDD = 2 Vdc

If the user wants the MCU to run from another clock source such as an external crystal with the FLL engaged, then the initialization software (running under the internal ~ 4 MHz clock) must set the ICG to the desired mode and then wait for the FLL to lock. As a result, initialization software can start quickly, but a wait time is necessary to have the desired frequency (and accuracy) of the programmed clock.

- Stop3 - Start-up from Stop3 varies greatly depending on clock configuration used (see [Table 3-10](#)). The minimum recovery time is about 100 µs independent of VDD, and this is when the internal reference generator (IRG) of the IGC (not SCM Mode) is used as the clock source when returning from Stop3

The clock source when returning from Stop3 can also be the external clock (crystal), and the external oscillator can be enabled or disabled while in Stop3 Mode. If the external oscillator is disabled (MCU control bit OSCSTEN = 0) during Stop3, then the MCU will start with the internal 4 MHz bus clock to service the interrupt, and then the software must wait until the FLL has locked to run with the external clock source. This condition uses the lowest power for Stop3 and requires

the RTI to use the internal RTI reference generator. The disadvantage for this condition is that the crystal oscillator must first start and stabilize and then the FLL must lock which can lead to a long wait time (similar to wait time in Stop2 after the Clock Mode is reprogrammed).

If the external oscillator stays enabled (MCU control bit OSCSTEN = 1) during Stop3, then the MCU will still start with the internal 4 MHz bus clock to service the interrupt, and then the software must also wait until the FLL has locked to run with the external clock source. This condition uses more power in Stop3, but has two advantages. First, with the external oscillator enabled during Stop3, the RTI clock source can be the crystal oscillator and the RTI period can be very accurate. Second, if the oscillator is running the recovery time for the FLL is very short because the crystal startup time is not required.

The worst case start-up time with an external crystal is probably a 32.768 kHz oscillator. The low frequency crystal oscillator has a long stabilization time. Any higher frequency crystal would stabilize faster, and as a result, the times listed in [Table 3-10](#) would tend to be a worst case situation. The external oscillator can also be used with the CLKO output of the modem. If the modem is in Off or Hibernate Mode while the MCU is in Stop3, then the CLKO is not available and the external reference cannot be enabled (OSCSTEN = 0). When starting up from Stop3, the MCU code must release the modem low power mode (under the ~ 4 MHz internal bus clock) and wait for the CLKO to be available and the FLL to lock before the MCU is at the desired clock source and frequency. In modem Doze Mode, the CLKO can be kept alive if desired or be disabled. If CLKO is kept alive in Doze Mode, the MCU can use RTI with CLKO as the external source (at the cost of higher power) and eliminate a separate crystal (such as a 32.768 kHz crystal) for the MCU.

**Table 3-10. MCU Stop3 Approximate Recovery Time**

Clock Source	Approximate Recovery Time
Internal clock used as reference	100 $\mu$ s
External 32 kHz crystal (OSCTEN = 0)	180 ms - 300 ms
External 32 kHz crystal (OSCTEN = 1)	2.4 ms
External CLKO (CLKO enabled out of restart)	Modem recovery time + 2.4 ms (worst case)
External CLKO (CLKO enabled in Doze Mode)	2.4 ms

### 3.12.6 Run Time Current

As previously stated, the modem is fully under control of the MCU. For lowest power, the modem should be kept in a low power mode as much as possible. However, it is also important to understand the current profile of the modem under active operation. In a similar sense, the MCU can use varying levels of current depending primarily on the clock sources and frequencies used.

#### 3.12.6.1 Modem Active Currents

In normal operational mode the modem's rest state in the Idle Mode. All active sequences originate from the Idle Mode and return to the Idle Mode. The three active sequences are Clear Channel Assessment

(CCA), RX, and TX, and each has a separate current profile. [Table 3-11](#) lists the typical currents while in the listed modes, but does not show the transition profiles when moving between modes.

**Table 3-11. MC1321x Active State Currents**

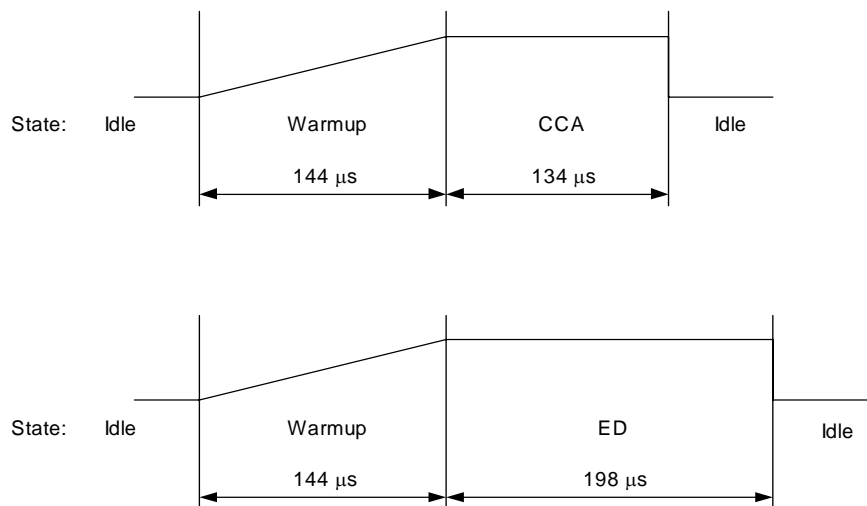
Mode	Current (typ @ 2.7V)
Idle	500 $\mu$ A
CCA/ED	37 mA
RX	37 mA
TX (0 dBm nominal output power)	30 mA

A normal sequence of events may include a 802.15.4 node performing first a CCA to see if the channel is clear, second transmitting a frame (assuming the channel is clear), and finally after the TX, going into to Receive Mode to look for an acknowledge. The modem must be programmed for each of the operations separately and each operation has a different timing profile.

### 3.12.6.1.1 Modem CCA/ED Timing profile

The modem will scan for detected energy in a CCA operation (CCA is covered in detail in [Section 7.3.5, “Clear Channel Assessment \(CCA\) Modes \(including Link Quality Indication\)”](#)). This is really a special case of RX so the CCA current is the same as RX. There are two versions of CCA where one is called CCA and the second is called Energy Detect (ED).

[Figure 3-10](#) shows the timing profiles for both variations of a CCA operation. Once the CCA operation is initiated, the state machine moves through a warm-up period of 144  $\mu$ s in which the analog regulators turn on and the analog RX circuitry comes to full power. The actual CCA or ED operation lasts 134  $\mu$ s or 198  $\mu$ s, respectively. During the warm-up period, the modem current is ramping from idle current (typically 500  $\mu$ A) to full CCA current (typically 37 mA). The modem current for the CCA or ED is the full CCA current. After the CCA/ED operation times out, the return to idle current is very quick.



**Figure 3-10. CCA and ED Timing Profiles**



### 3.12.6.1.2 Modem RX Timing profile

The receive or RX timing profile is very similar to the CCA profile. [Figure 3-11](#) shows the timing profile for an RX operation. There is the initial 144  $\mu\text{s}$  warm-up period from idle current to full RX current (typically 37 mA) followed by the RX operation (RX is covered in detail in [Section 7.3, “Active Modes”](#)). The RX time is not a set figure as it is in a CCA operation. In some applications (typically not battery operated) such as a Coordinator, the receiver can be turned on for a majority of the timer listening for End Devices or Routers. In an End Device (typically battery operated), the receiver can be typically turned on only when expecting an acknowledgement (ACK) of a transmission. The worst case RX on time can be when no ACK is received and the RX operation times out with not having received a frame.

The RX operation will end based on receiving the end of a frame or being terminated by the application having timed-out and aborted the RX operation. The return to Idle happens very quickly.

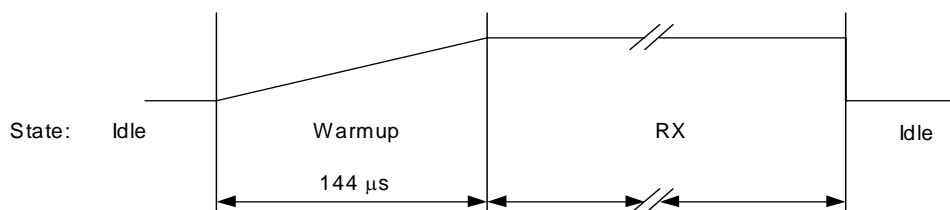


Figure 3-11. RX Timing Profile

### 3.12.6.1.3 Modem TX Timing profile

The transmit or TX timing profile is more predictable than the RX profile. [Figure 3-12](#) shows the timing profile for a TX operation. There is the usual initial 144  $\mu\text{s}$  warm-up period from idle current to full TX current (typically 30 mA) followed by the TX operation (TX is covered in detail in [Section 7.3, “Active Modes”](#)). The TX time is not a set figure but it is predictable.

The raw transmission rate of the 802.15.4 2.4 GHz physical layer is 62.5 ksymbols/s or 250 kb/s. This means the TX time for 2 symbols or 1 byte of data is 32  $\mu\text{s}$ . An 802.15.4 2.4 Standard compliant packet has 4 bytes of preamble, 1 byte of SFD, 1 byte of FLI, 2 bytes of FCS plus the payload data (125 bytes maximum). As a result the, the overhead of a frame is 8 bytes or  $8 \times 32 = 256 \mu\text{s}$ , and the maximum payload TX time is  $125 \times 32 = 4000 \mu\text{s}$ . The TX time for a packet then is:

$$\text{Total TX time } (\mu\text{s}) = 256 + (\text{payload bytes} \times 32)$$

The TX operation will end after the FCS bytes are sent. The return to Idle has a “warm down” period of 10  $\mu\text{s}$  to allow the RF transmitter to taper off in a manner to avoid RF “splatter”.

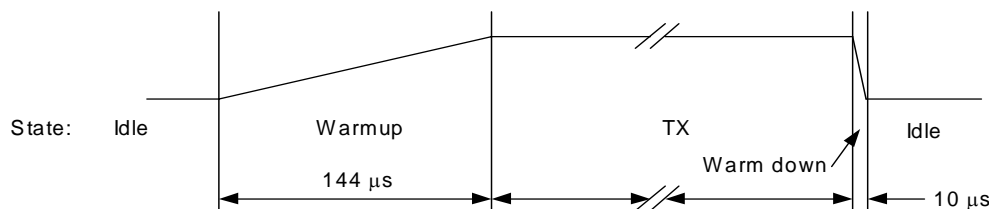


Figure 3-12. TX Timing Profile

### 3.12.6.2 MCU Active Current

The MCU active (run) current is dominated by the system clock frequency. Table 3-12 shows typical MCU supply current under several conditions. The Wait Mode is not running code but has the clocks active and has a baseline of about 0.56 mA. As the bus clock gets faster the supply current increases accordingly with a value of 1.1 mA at 1 MHz bus clock and 6.5 mA at 8 MHz bus frequency. A simple formula for estimating typical current vs. bus clock is:

$$\text{Typical current} = 0.56 \text{ mA} + (\text{bus clock frequency (MHz)} \times 0.77 \text{ mA/MHz})$$

A 16 MHz bus clock example would be:

$$\text{Typical current @ 16 MHz} = 0.56 \text{ mA} + (16 \text{ MHz} \times 0.77 \text{ mA/MHz}) = 12.9 \text{ mA}$$

Another significant factor can be the ATD. The ATD has a typical enabled current of 0.7 mA. To get lowest power only enable the ATD as required to sample inputs.

**Table 3-12. MCU Active Mode Supply Current**

Condition	Current (typ @ 3.0V)
Run supply current <sup>1</sup> at CPU clock = 2 MHz, f <sub>Bus</sub> = 1 MHz	1.1 mA
Run supply current at CPU clock = 16 MHz, f <sub>Bus</sub> = 8 MHz	6.5 mA
Run supply current at CPU clock = 32 MHz, f <sub>Bus</sub> = 16 MHz	12.9 mA (calculated)
Wait supply current with f <sub>Bus</sub> = 1 MHz	560 μA
ATD supply current when enabled	700 μA

<sup>1</sup> All MCU modules except ATD active, ICG configured for FBE, and no dc loads on port pins.

### 3.12.7 Configuration of Interconnected GPIO for Low Power Operation

As stated elsewhere, device GPIO must be configured properly for low power operation. The GPIO on the MC1321x are part of either the transceiver or the MCU, and their behavior is somewhat different. Also, some I/O are interconnected between chips onboard the device (see Section 3.4, “Internal Functional Interconnects and System Reset”), and this affects their use during low power. There are generally the following low power scenarios:

- Hardware reset ( $\overline{\text{RESET}}$  pin to the MCU is asserted low). **NOT RECOMMENDED** - this holds the MCU in reset, but not the transceiver because the transceiver reset is controlled by the MCU. This mode is not normally recommended as a long-term, “held” state.
  - The MCU GPIO default to inputs with no pullups. The user should provide hardware-based known states for all the inputs if this condition is held and low power is desired.
  - The transceiver reset connection (Pad 71) should have a pulldown added to hold the chip in reset.
  - The transceiver GPIO default to inputs with no pullups. The user should provide hardware-based known states for all the inputs if this condition is held and low power is desired.

- MCU active and transceiver held in reset. - If the transceiver is not used for an extended period, the transceiver can be held in reset (MCU port PTD3 asserted low). For lowest power in the transceiver:
  - SPI port pins SPSSCK1, MOSI1, and MISO1 on the MCU should be driven to low.  $\overline{SS1}$  should be driven to high.
  - Other interconnected MCU pins (PTD1, PTD3, PTE6, and PTE7) should be driven to low
  - Transceiver GPIO2 - GPIO7 should be held low externally through hardware.
- MCU active and transceiver programmed to Hibernate or Doze mode. - For lowest power in the transceiver:
  - SPI port pins SPSSCK1, MOSI1, and MISO1 on the MCU should be driven to low.  $\overline{SS1}$  should be driven to high. MISO pin on the transceiver should be programmed to drive low when SPI is disabled ( $\overline{CE}$  is high)
  - Other interconnected MCU pins (PTD1, PTD3, PTE6, and PTE7) should be driven to low
  - Transceiver GPIO2 - GPIO7 should be programmed to outputs in the low state.
- MCU in low power (Stop2 or Stop3) and transceiver in Hibernate or Doze mode - These variations are most common and are better as GPIO on both chips retain their programmed state.
  - SPI port pins SPSSCK1, MOSI1, and MISO1 on the MCU should be driven to low.  $\overline{SS1}$  should be driven to high. MISO pin on the transceiver should be programmed to drive low when SPI is disabled ( $\overline{CE}$  is high)
  - Other interconnected MCU pins (PTD1, PTD3, PTE6, and PTE7) should be driven to low
  - Transceiver GPIO2 - GPIO7 should be programmed to outputs in the low state.

### 3.12.8 General System Considerations for Low Power

Low power is most important for battery operated applications such as 802.15.4 End Devices. In the modem the highest instantaneous power is dominated by the CCA, RX and TX modes. In the MCU, the current draw is dominated by the clock frequency. A number of general recommendations can help the user:

- Clock Management on the MCU
  - Use clock management as required to lower required power
  - Run fast when the CPU performance is the critical path
  - Run slow when waiting on peripherals (such as ATD conversion)
  - Be aware of software performance requirements (run time bus clock may need to be 16 MHz)
  - Use the external clock option for lowest power if possible (modem CLKO can supply frequencies as high as 16 MHz)
- For long “sleep” periods between node activity and lowest power, use Stop3 with the Off condition on the modem. Use the RTI capability to wake up the MCU
  - If the wake up time period can vary by +/-30% use the internal RTI oscillator. Maximum time between RTI requests is nominally 1 second. This has lowest power and lowest cost
  - If the wake up time period must be crystal accurate, use the external clock option for the RTI timer with a 32.768 kHz crystal. Maximum time between RTI requests is 1 second. This has

- the advantage of crystal accuracy and next lowest power with the disadvantage of higher cost. The crystal can also supply high frequency clock performance through use of the FLL
- If lowest cost is important and crystal accuracy is required for the wake up time period, one option is to use the modem Doze Mode with CLKO active. CLKO can be programmed for 16.786+ kHz and is crystal accurate. Two disadvantages are a penalty of about 35-40  $\mu$ A additional “sleep” current and the reference frequency is not exactly 32.768 kHz
  - The MCU can go to a lower power mode such as Wait Mode when the modem is recovering from the Off (25 ms) or the Hibernate modes
  - When the MCU is initializing from a “cold start” POR condition, the modem reset must be driven to a low condition early in the routine. This is so the modem will start from a known condition and low power
  - Profile the use scenario of both the modem and the MCU when they are active. This is required to estimate current usage versus time and mode of operation
  - If possible put the modem in a low power mode, if the MCU is active and does not require interface with the modem. Doze can be useful to save power, provide CLKO, and be ready with a quick recovery time
  - Alternately put the MCU in a lower power mode if waiting on the modem IRQ
  - Program all unused GPIO on both the modem and the MCU (including port signals not pinned-out) as outputs for lowest power
  - Best practice is to disable the MCU LVD during low power modes. Use LVD only if required.
  - The default condition is for modem MISO to go to tristate when  $\overline{CE}$  is de-asserted. Program modem Control\_B Register 07, Bit 11, `miso_hiz_en = 0` so MISO will be driven low when CE is de-asserted. As a result MISO will not float when Doze or Hibernate mode is enabled.

# Chapter 4

## MC1321x Serial Peripheral Interface (SPI)

The MC1321x modem and CPU communicate primarily through the onboard SPI command channel. Figure 4-1 shows the SiP internal interconnects with the SPI bus highlighted. The MCU has a single SPI module that is dedicated to the modem SPI interface. The modem is a slave only and the MCU SPI must be programmed and used as a master only. Further, the SPI performance is limited by the modem constraints of 8 MHz SPI clock frequency, and use of the SPI must be programmed to meet the modem SPI protocol.

### 4.1 SiP Level SPI Pin Connections

The SiP level SPI pin connections are all internal to the device. Figure 4-1 shows the SiP interconnections with the SPI bus highlighted.

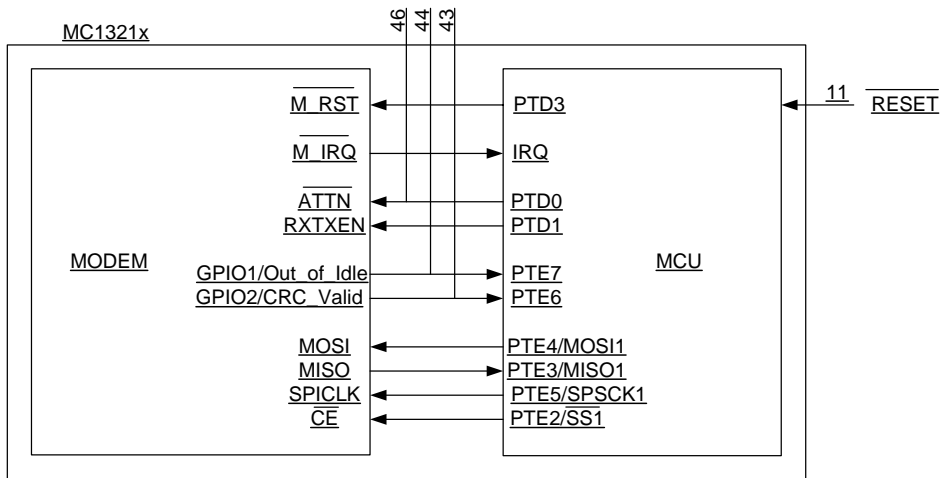


Figure 4-1. MC1321x Internal Interconnects Highlighting SPI Bus

Table 4-1. MC1321x Internal SPI Connections

MCU Signal	Modem Signal	Description
PTE5/SPSCK1	SPICLK	MCU SPI master SPI clock output drives modem SPICLK slave clock input.
PTE4/MOSI1	MOSI	MCU SPI master MOSI output drives modem slave MOSI input
PTE3/MISO1	MISO	Modem SPI slave MISO output drives MCU master MISO input
PTE2/SS1	CE	MCU SPI master SS output drives modem slave $\overline{CE}$ input

## 4.2 Features

Features of the SPI bus interface:

- MCU bus master
- Modem bus slave
- Bi-directional data transfer
- Dedicated interface; must meet modem protocol requirements
- Programmable SPI clock rate; maximum rate is 8 MHz
- Double-buffered transmit and receive at MCU
- Serial clock phase and polarity must meet modem requirements (MCU control bits CPHA = 0 and CPOL = 0)
- Slave select programmed to meet modem protocol
- MSB-first shifting

## 4.3 SPI System Block Diagram

This section shows the system level diagram for the SPI. Figure 4-2 shows the SPI modules of the MCU and modem in the master-slave arrangement.

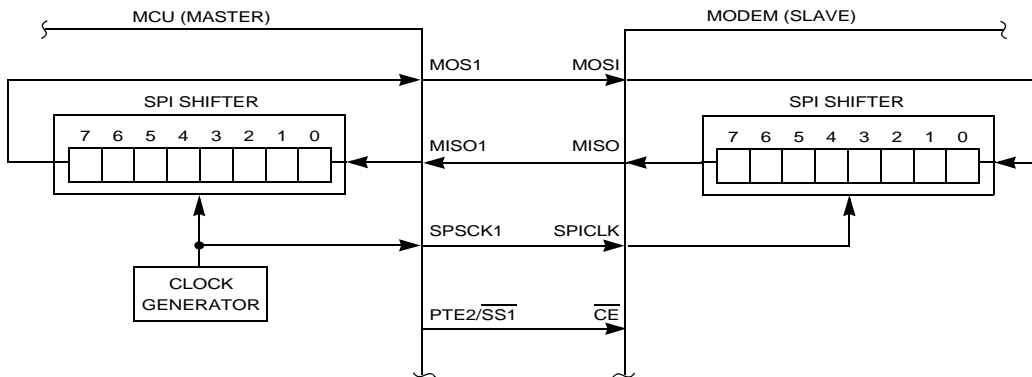


Figure 4-2. SPI System Block Diagram

The MCU (master) initiates all SPI 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. Although the SPI interface supports simultaneous data exchange between master and slave, the modem SPI protocol only uses data exchange in one direction at a time. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ( $\overline{SS1}$  pin).

## 4.4 Modem SPI Overview

Control of the modem and data transfers between the modem and CPU are accomplished by means of the 4-wire SPI. The modem SPI port is a fully static design that requires no additional clocks besides SPICLK for accessing internal registers, although Packet RAM accesses do require the modem reference oscillator to be running. This allows for lower power when the SPI must stay alive for data retention while the rest of the device is in power-down.

Although the normal SPI protocol is based on 8-bit transfers, the modem imposes a higher level transaction protocol that is based on multiple 8-bit transfers per transaction. In its simplest form, a singular SPI read or write transaction consists of an 8-bit header transfer followed by two 8-bit data transfers. The header denotes access type and register address. The following bytes are read or write data. The SPI also supports recursive 'data burst' transactions in which additional data transfers can occur. The Recursive Mode is primarily intended for Packet RAM access and fast configuration of the MC1321x. Partial word accesses are not supported.

All modem SPI accessible registers are configured with 16-bit data width. The address range is 6 bits which allows for 64 locations although not all are implemented. Internal data RAMs are accessed as dedicated addresses within the register address field.

An additional feature is a software reset capability where a write to Address 00 will accomplish most of the equivalency of a hardware reset.

## 4.5 Modem SPI Basic Operation

The modem operates as a SPI slave only. The microcontroller supplies the interface clock and acts as SPI master.

### 4.5.1 Modem SPI Pin Definition

The modem signals of  $\overline{CE}$ , SPICLK, MOSI, and MISO are defined in the following paragraphs.

#### 4.5.1.1 Chip Enable ( $\overline{CE}$ )

A transaction on the SPI port is framed by the active low Chip Enable ( $\overline{CE}$ ) input signal which is driven by the MCU. A transaction is a minimum of 3 SPI byte bursts and can extend to a greater number of bursts.

#### 4.5.1.2 SPI Clock (SPICLK)

The host drives the SPI Clock (SPICLK) input to the modem. Data is clocked into the master or slave on the leading (rising) edge of the return-to-zero SPICLK and data out changes state on the trailing (falling) edge of SPICLK.

#### NOTE

For the HCS08 microcontroller, the SPI clock format is the clock phase control bit CPHA = 0 and the clock polarity control bit CPOL = 0.

### 4.5.1.3 Master Out / Slave In (MOSI)

The Master Out/Slave In (MOSI) input presents incoming data from the host to the transceiver (slave input).

### 4.5.1.4 Master In / Slave Out (MISO)

The MC1321x presents data to the master on its MISO output. This output is user configurable for both drive strength and its off state.

#### 4.5.1.4.1 Setting MISO Output Drive Strength

MISO output drive strength is programmed by writing to `miso_drv[1:0]`, `GPIO_Data_Out` Register 0C. There are 4 levels of drive strength with field value 00 for lowest and value 11 for greatest. The default value is 00.

#### NOTE

It is suggested the user program MISO for greatest drive strength for best performance.

#### 4.5.1.4.2 Setting MISO Off Impedance

MISO off impedance (output state when  $\overline{CE}$  is negated or high) can be programmed by writing to `miso_hiz_en`, `Control_B` Register 07. Setting `miso_hiz_en` to “1” causes MISO to tri-state when  $\overline{CE}$  is high, and this is the default state.

Writing `miso_hiz_en` to “0” causes MISO to be active low when  $\overline{CE}$  is negated.

#### NOTE

It is suggested that `miso_hiz_en` be programmed to “0” so that the MCU input MISO1 signal does not float when  $\overline{CE}$  is negated.



## 4.5.2 Modem SPI Byte Burst Operation

The SPI port of the MCU transfers data in bursts of 8 bits with most significant bit (MSB) first. Although a MC1321x transaction is three or more SPI bursts long, the timing of a single SPI burst is shown in Figure 4-3.

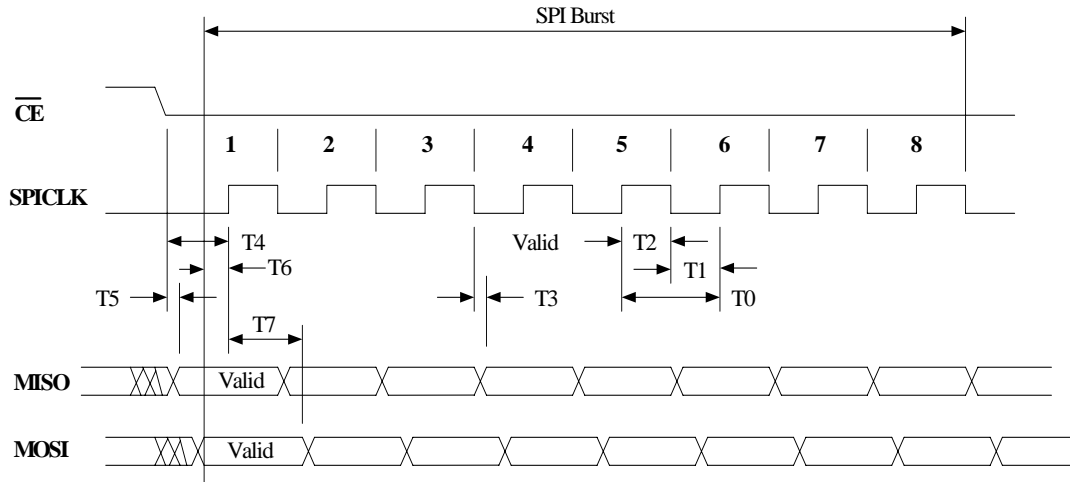


Figure 4-3. Modem SPI Burst Timing Diagram

Because the MCU is embedded in the SiP and the modem only supports one clock format, the MCU SPI must be programmed for this clock mode, i.e., clock phase control bit CPHA = 0 and the clock polarity control bit CPOL = 0. In addition, the MSB-first option must be selected.

Table 4-2. Modem SPI Timing Specifications

Symbol	Parameter	Min	Typ	Max	Unit
T0	SPICLK period	125			ns
T1	Pulse width, SPICLK low	50			ns
T2	Pulse width, SPICLK high	50			ns
T3	Delay time, MISO data valid from falling SPICLK		15		ns
T4	Setup time, $\overline{\text{CE}}$ low to rising SPICLK		15		ns
T5	Delay time, MISO valid from $\overline{\text{CE}}$ low		15		ns
T6	Setup time, MOSI valid to rising SPICLK		15		ns
T7	Hold time, MOSI valid from rising SPICLK		15		ns

## 4.6 MCU SPI Block Diagrams

This section includes block diagrams showing the internal organization of the SPI module and the SPI clock dividers that control the Master Mode bit rate.

### 4.6.1 MCU SPI Module Block Diagram

Figure 4-4 is a block diagram of the SPI module.

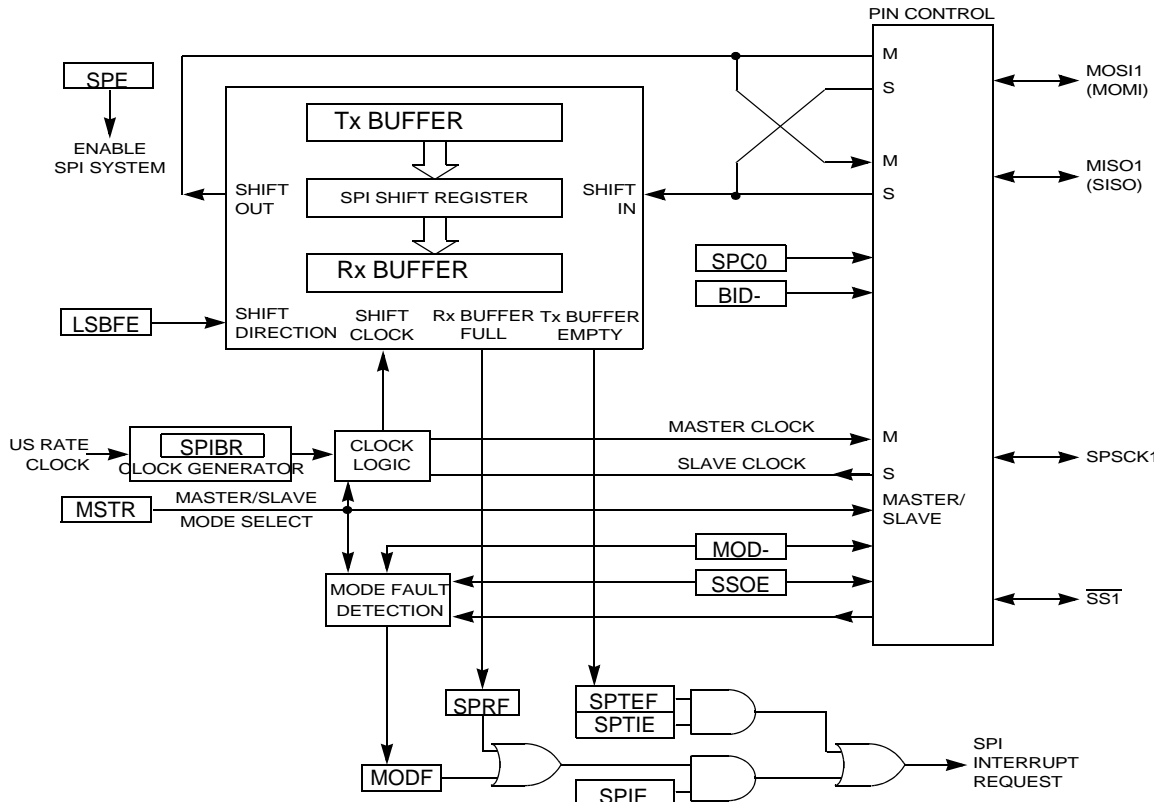


Figure 4-4. MCU SPI Module Block Diagram

The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPI1D) 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 SPI1D). 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 SPSCK1 pin, the shifter output is routed to MOSI1, and the shifter input is routed from the MISO1 pin.

When the SPI is configured as a slave, the SPSCK1 pin is routed to the clock input of the SPI, the shifter output is routed to MISO1, and the shifter input is routed from the MOSI1 pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

## 4.6.2 MCU SPI Baud Rate Generation

As shown in Figure 4-5, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI Master Mode bit-rate clock.

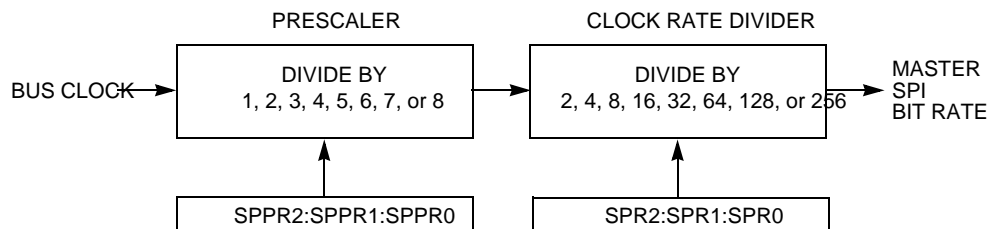


Figure 4-5. MCU SPI Baud Rate Generation

## 4.7 MCU SPI Functional Description

A SPI transfer is based on an 8-bit operation. A SPI transfer is initiated by checking for the SPI transmit buffer empty flag (SPTEF = 1) and then writing a byte of data to the SPI data register (SPI1D) in the master SPI device. When the SPI shift register is available, this byte of data is moved from the transmit data buffer to the shifter, SPTEF is set to indicate there is room in the buffer to queue another transmit character if desired, and the SPI serial transfer starts.

During the SPI transfer, data is sampled (read) on the MISO1 pin at one SPSCCK edge and shifted, changing the bit value on the MOSI1 pin, one-half SPSCCK cycle later. After eight SPSCCK cycles, the data that was in the shift register of the master has been shifted out the MOSI1 pin to the slave while eight bits of data were shifted in the MISO1 pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading SPI1D. If another byte of data is waiting in the transmit buffer at the end of a transfer, it is moved into the shifter, SPTEF is set, and a new transfer is started.

Normally, SPI data is transferred most significant bit (MSB) first. If the least significant bit first enable (LSBFE) bit is set, SPI data is shifted LSB first. For this embedded SPI interface, the LSB First Mode must not be used.

The MCU SPI module can be configured as a slave, however, this mode is not allowed in the MC1321x.

Because the transmitter and receiver are double buffered, a second byte, in addition to the byte currently being shifted out, can be queued into the transmit data buffer, and a previously received character can be in the receive data buffer while a new character is being shifted in. The SPTEF flag indicates when the transmit buffer has room for a new character. The SPRF flag indicates when a received character is available in the receive data buffer. The received character must be read out of the receive buffer (read SPI1D) before the next transfer is finished or a receive overrun error results.

In the case of a receive overrun, the new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for such an overrun condition so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

### 4.7.1 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 4-6 shows the clock format 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 line shows the order of SPI data bits determined by the setting in LSBFE. Only the allowed version of SPSCCK polarity is shown, set by 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 the 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.

For a common SPI interface  $\overline{SS}$  would go back high at the end of the transfer, however, the modem protocol as described in Section 4.9, “Modem SPI Singular Transactions” and Section 4.11, “Modem SPI Recursive Transactions”, requires that the  $\overline{SS}$  (CE) stay low for an entire multibyte transaction.

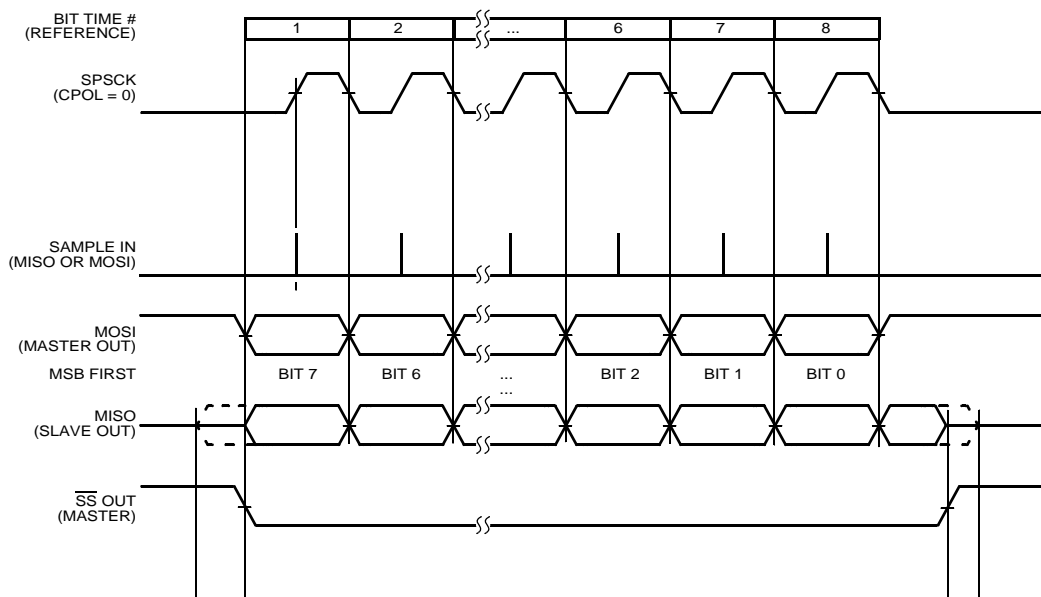


Figure 4-6. MCU SPI Clock Format (CPHA = 0 and CPOL = 0)

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.

## 4.7.2 MCU SPI Pin Controls

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.

### 4.7.2.1 SPCK1 — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input (not allowed). When the SPI is enabled as a master, this pin is the serial clock output (desired mode).

### 4.7.2.2 MOSI1 — 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 (desired mode).

When the SPI is enabled as a slave (not allowed) 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.

### 4.7.2.3 MISO1 — 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 (desired mode).

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.

### 4.7.2.4 $\overline{SS1}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input (not allowed).

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 (desired mode) PTE2. 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 ( $SSOE = 1$ ).

## 4.7.3 MCU 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).

#### 4.7.4 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  $\overline{SS1}$  pin (provided the  $\overline{SS1}$  pin is configured as the mode fault input signal). The  $\overline{SS1}$  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  $\overline{SS1}$  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 SPCK1, MOSI1, and MISO1 (if not Bidirectional Mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPI1C1). User software should verify the error condition has been corrected before changing the SPI back to Master Mode.

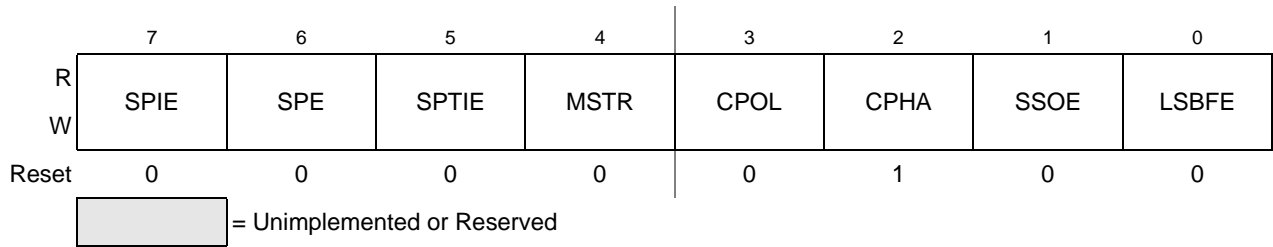
### 4.8 MCU SPI Registers and Control Bits

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 [Chapter 11, “MCU Memory”](#) of this manual 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.

## 4.8.1 SPI Control Register 1 (SPI1C1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.



**Figure 4-7. SPI Control Register 1 (SPI1C1)**

**Table 4-3. SPI1C1 Field Descriptions**

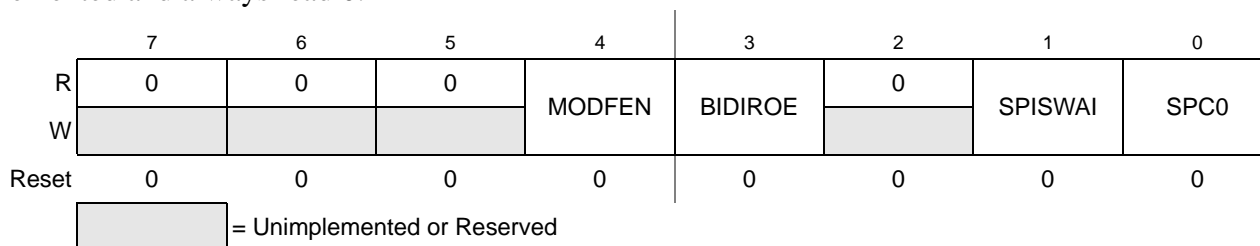
Field	Description
7 SPIE	<b>SPI Interrupt Enable (for SPRF and MODF)</b> — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling). 1 When SPRF or MODF is 1, request a hardware interrupt.
6 SPE	<b>SPI System 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.
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 4.7.1, “SPI Clock Formats”</a> , for more details. 0 Active-high SPI clock (idles low). 1 Active-low SPI clock (idles high).
2 CPHA	<b>Clock Phase</b> — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to <a href="#">Section 4.7.1, “SPI Clock Formats”</a> , for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer. 1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer.
1 SSOE	<b>Slave Select Output Enable</b> — This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the SS1 pin as shown in <a href="#">Table 4-4</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 4-4.  $\overline{SS1}$  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

## 4.8.2 SPI Control Register 2 (SPI1C2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

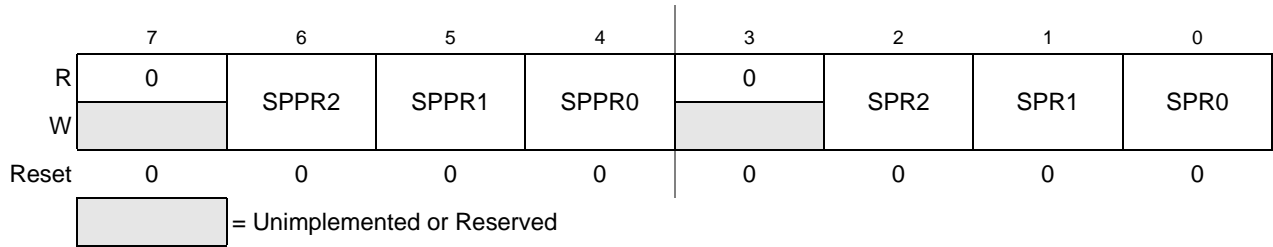

**Figure 4-8. SPI Control Register 2 (SPI1C2)**
**Table 4-5. SPI1C2 Field Descriptions**

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 $\overline{SS1}$ pin is the slave select input.) In Master Mode, this bit determines how the $\overline{SS1}$ pin is used (refer to <a href="#">Table 4-4</a> for more details). 0 Mode fault function disabled, master $\overline{SS1}$ pin reverts to general-purpose I/O not controlled by SPI. 1 Mode fault function enabled, master $\overline{SS1}$ 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 MOSI1 (MOMI) or MISO1 (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 MISO1 (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (Master Mode), the SPI uses the MOSI1 (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.



### 4.8.3 SPI Baud Rate Register (SPI1BR)

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 4-9. SPI Baud Rate Register (SPI1BR)**

**Table 4-6. SPI1BR Field Descriptions**

Field	Description
6:4 SPPR[2:0]	<b>SPI Baud Rate Prescaler Divisor</b> — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in <a href="#">Table 4-7</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 4-5</a> ).
2:0 SPR[2:0]	<b>SPI Baud Rate Divisor</b> — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in <a href="#">Figure 4-8</a> . The input to this divider comes from the SPI baud rate prescaler (see <a href="#">Figure 4-5</a> ). The output of this divider is the SPI bit rate clock for Master Mode.

**Table 4-7. SPI Baud Rate Prescaler Divisor**

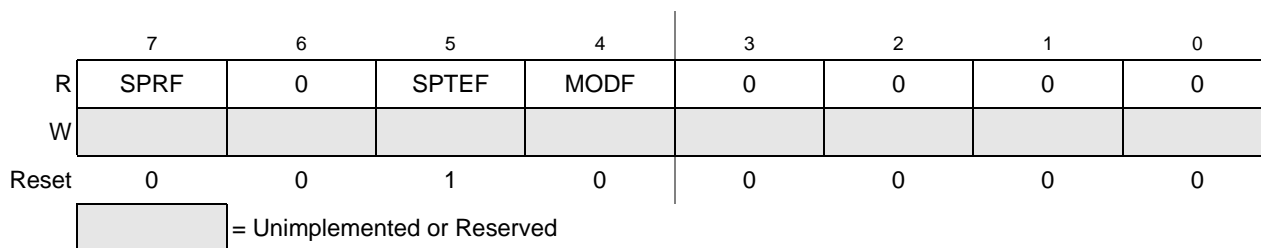
SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

**Table 4-8. SPI Baud Rate Divisor**

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

### 4.8.4 SPI Status Register (SPI1S)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0s. Writes have no meaning or effect.



**Figure 4-10. SPI Status Register (SPI1S)**

**Table 4-9. SPI1S Field Descriptions**

Field	Description
7 SPRF	<p><b>SPI Read Buffer Full Flag</b> — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPI1D). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register.</p> <p>0 No data available in the receive data buffer. 1 Data available in the receive data buffer.</p>
5 SPTEF	<p><b>SPI Transmit Buffer Empty Flag</b> — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPI1S with SPTEF set, followed by writing a data value to the transmit buffer at SPI1D. SPI1S must be read with SPTEF = 1 before writing data to SPI1D or the SPI1D write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPI1C1 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 SPI1D is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter.</p> <p>0 SPI transmit buffer not empty. 1 SPI transmit buffer empty.</p>
4 MODF	<p><b>Master Mode Fault Flag</b> — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The <math>\overline{SS1}</math> pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPI1C1).</p> <p>0 No mode fault error. 1 Mode fault error detected.</p>

## 4.8.5 SPI Data Register (SPI1D)

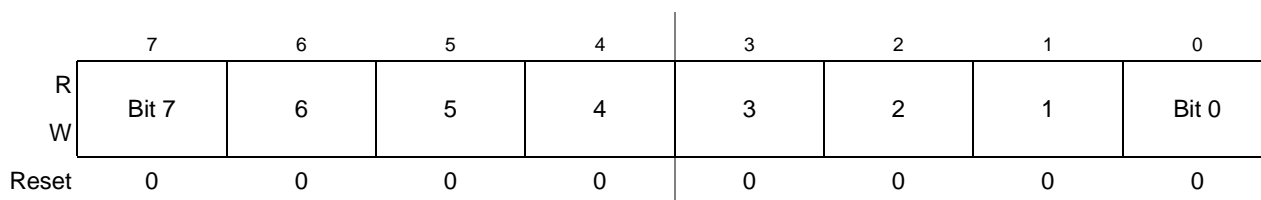


Figure 4-11. SPI Data Register (SPI1D)

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 SPI1D 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.

## 4.9 Modem SPI Singular Transactions

Although the SPI port of the MCU transfers data in bursts of 8 bits, the MC1321x requires that a complete SPI transaction be framed by  $\overline{CE}$ , and there will be 3 or more bursts per transaction. There are generally two classes of transactions, which are singular and recursive.

Both of the type transactions are used to access the modem SPI registers for reading and writing of data. To view the modem SPI register map and descriptions of the modem registers, reference [Chapter 5, “Modem SPI Register Descriptions”](#).

### 4.9.1 SPI Singular Transaction Signalling

The assertion of  $\overline{CE}$  to low signals the start of a transaction. The first SPI burst is a write of an 8-bit header to the transceiver (MOSI is valid) that defines a 6-bit address of the internal resource being accessed and identifies the access as being a read or write operation. In this context, a write is data written to the modem and a read is data written to the SPI master. The following SPI bursts will be either the write data (MOSI is valid) to the transceiver or read data from the transceiver (MISO is valid).

Although the SPI bus is capable of sending data simultaneously between master and slave, the MC1321x never uses this mode. The number of data bytes (payload) will always be 2 for a singular access. After the final SPI burst,  $\overline{CE}$  is negated to high to signal the end of the transaction. Should a SPI programming attempt fail to provide the modem with at least 24 rising SPICLK edges prior to  $\overline{CE}$  negating, no register data will be changed.

Figure 4-12 shows a read access transaction.

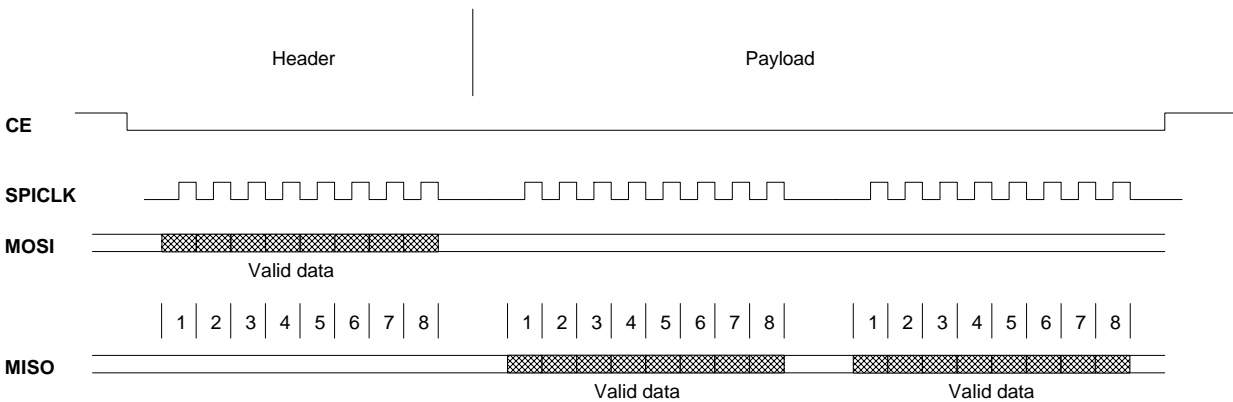


Figure 4-12. SPI Singular Read Access

Figure 4-13 shows a write access transaction.

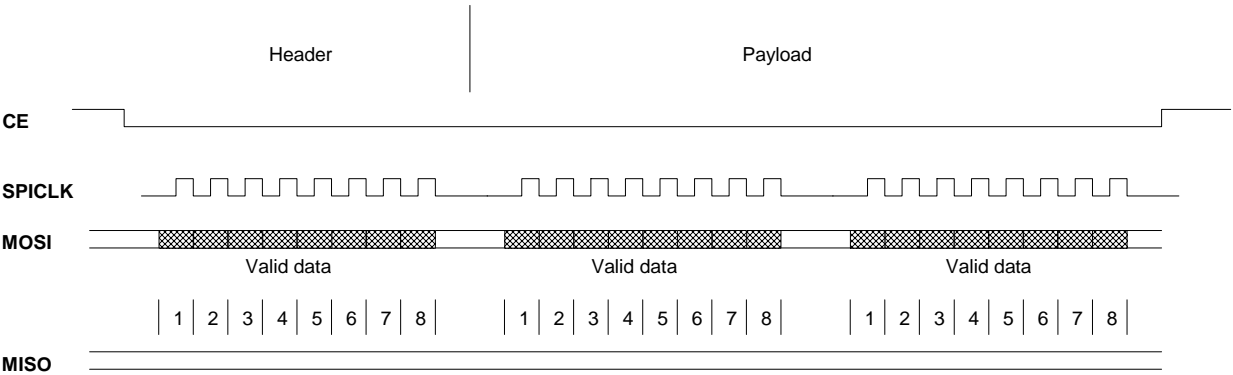


Figure 4-13. SPI Singular Write Access

### 4.9.2 SPI Singular Transaction Protocol

A SPI transaction is divided into a header field and a payload. The header field is always 8 bits, while the payload data is in multiples of 2 bytes or 16 bits. A singular SPI transaction contains 24 bits of information. The header field contains a  $R/\overline{W}$  bit and a 6-bit register address, as shown in Figure 4-14.

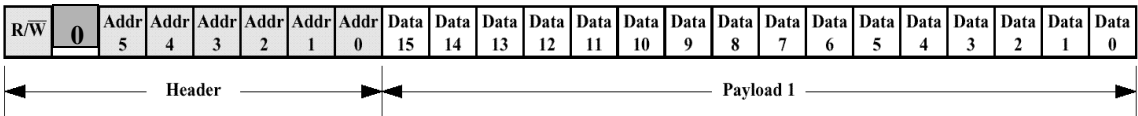


Figure 4-14. SPI Header and Payload Definition

The  $R/\overline{W}$  bit identifies the transfer as a read ( $R/\overline{W} = 1$ ) or write ( $R/\overline{W} = 0$ ). The lower 6 bits in the header determines which of the 64 possible SPI locations is to be accessed for a read or write operation although not all possible addresses are implemented. The register address field also provides the starting address for a recursive read or write operation as described later. The register data is presented MSB first.

## 4.10 Modem Symbol/Data Format

When the modem transceiver receives 802.15.4 symbols, they are assembled as 4 symbols per 16-bit word. They are then presented to the RX Packet RAM or to the SPI directly as a 16-bit word. The ordering of symbols vs. word bit ordering is shown in Figure 4-15 which details the RX data flow through the device. In Packet Mode the data is loaded into RX Packet RAM, and in Stream Mode the data is sent directly to the SPI buffer. When the symbols are read via the SPI bus, 2 SPI bursts per 16-bit word are required and the MSB is presented first such that the symbols appear to the MCU in reverse order.

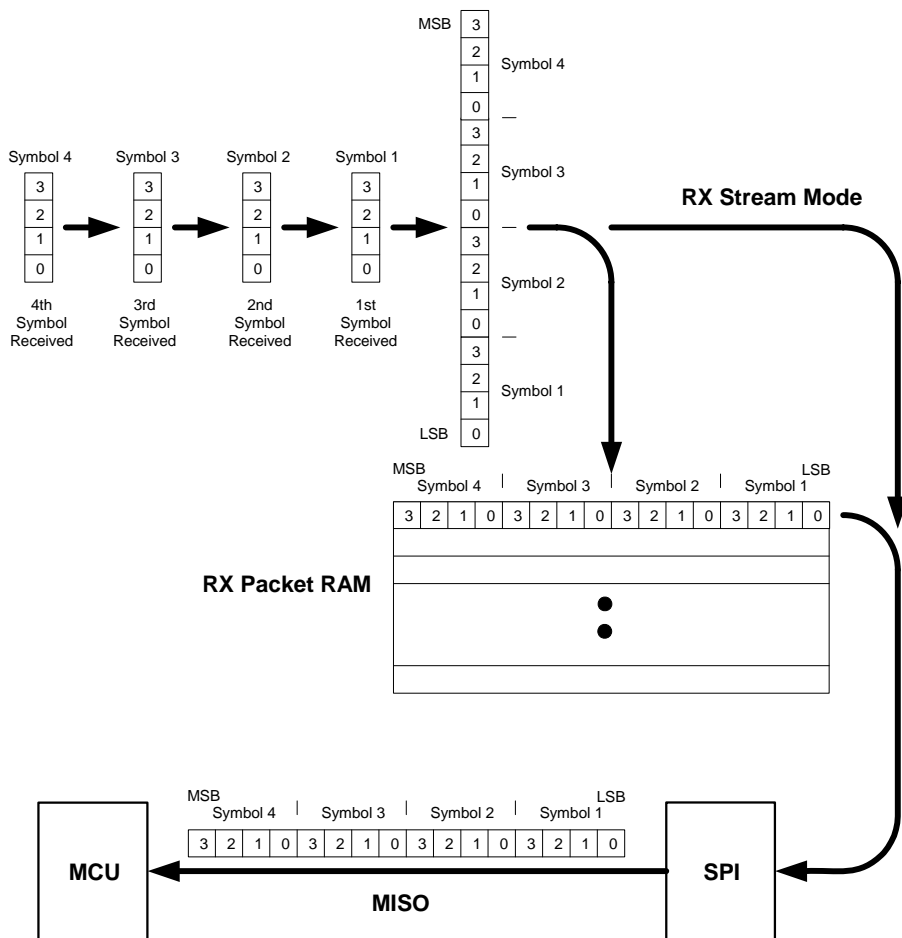


Figure 4-15. RX Symbol Flow Diagram

The inverse of the RX symbol / data flow is the case for TX data / symbols.

## 4.11 Modem SPI Recursive Transactions

The MC1321x SPI also incorporates a recursive or ‘data burst’ transaction capability. This allows multiple sequential SPI registers to be accessed with only one header field. Recursive reads and writes provide significant reduction in SPI overhead and a corresponding increase in programming speed.

1. The primary intent is for the software to be able to rapidly configure the modem.
2. Recursive reads and writes are convenient for accessing SPI register values which extend beyond the 16-bit SPI register format. Examples include writing Timer Comparator values (tmr\_cmp1[23:0] through tmr\_cmp4[23:0]) and reading the Time Stamp (timestamp[23:0]).
3. Recursive access capability enables the contents of modem Packet RAM to be read or written in an expedient manner.

### 4.11.1 Recursive SPI Register Read

Recursive register reads are invoked in an identical manner to singular read operations, however, by holding  $\overline{CE}$  asserted for additional SPI bursts after the first 16-bit data payload is shifted out, the contents of the next SPI register address is made available on the MISO pin. An internal SPI register address pointer is automatically incremented during recursive reads to point to the next sequential SPI register location. For each subsequent set of 2 SPI bursts, SPICLK shifts out the contents of the next register address.

This sequence repeats as long as the  $\overline{CE}$  is held asserted, allowing multiple sequential register contents of the SPI to be read starting at the header address. As the recursive read progresses, the SPI register address pointer continues to increment. When the address pointer reaches 63 decimal (the maximum implemented SPI register), the address pointer ‘rolls over’ to Address 03 and begins incrementing again. Address 03 is chosen to avoid the Program Reset function at Address 00 and the Rx Packet RAM and Tx Packet RAM at Addresses 01 and 02, respectively.

### 4.11.2 Recursive SPI Register Write

Recursive writes are invoked in an identical manner to singular write operations. But, by holding  $\overline{CE}$  asserted for additional SPI bursts after the first 16-bit data payload is shifted in, the contents of the next SPI register address can be programmed. An internal SPI register address pointer is automatically incremented during recursive writes to point to the next higher sequential SPI register location. For each subsequent set of 2 SPI bursts, SPICLK shifts in the write data to the next register address.

This sequence repeats as long as the  $\overline{CE}$  is held asserted, allowing multiple sequential register contents to be written starting at the header address. As the recursive write progresses, the SPI register address pointer continues to increment. When the address pointer reaches 63 decimal (the maximum implemented SPI register), the address pointer ‘rolls over’ to Address 03 and begins incrementing again. Address 3 is chosen to avoid the Program Reset function at Address 00 and the Rx Packet RAM and Tx Packet RAM at Addresses 01 and 02, respectively.

### 4.11.3 Special Case - Packet RAM Access

Modem Packet RAM access is a special case access when the modem is used in the Packet Mode. The modem contains three embedded 128-byte ‘Packet RAMs’ used to facilitate reception and transmission of packet data. One RAM is dedicated for receive packet data and two are dedicated for transmit packet data. The Packet RAMs are configured in 64-word by 16-bit format and are both read and write accessible via the SPI. The recursive data ‘burst’ access mode is used to efficiently access Packet RAM data.

Although the Packet RAMs are completely static, any RAM access requires the MC1321x to be in an active state; the reference clock circuit must be active. RAM read and write operations are prohibited while the device is in Hibernate or Doze Mode.

Reading and writing the MC1321x Packet RAMs is accomplished by SPI burst accesses to dedicated Packet RAM register addresses within the register map. The RX\_Pkt\_RAM Register 01 is mapped to the Rx Packet RAM, and TX\_Pkt\_RAM Register 02 is mapped to the Tx Packet RAMs (choice is selected by the tx\_ram2\_select bit TX\_Pkt\_Ctl Register 03, Bit 15). The 16-bit data payload of the SPI access maps directly to the 16-bit word in the Packet RAM.

#### 4.11.3.1 Recursive Receive Packet RAM Read Access

The receive Packet RAM is normally accessed when the modem is in Packet Data Mode and a valid frame has been received (as indicated by rx\_rcvd\_irq and crc\_valid). The number of receive data bytes in the queue is shown by the rx\_pkt\_latch[6:0] field (this includes the full payload with the 2 CRC bytes).

The data is read by accessing RX\_Pkt\_RAM Register 01 with a recursive read. When accessing RX\_Pkt\_RAM Register 01, the SPI register address pointer is NOT incremented, instead, the Packet RAM read address pointer is incremented. Therefore, by using a recursive read, up to 64 words of packet memory can be read from the SPI with an access that requires but a single header field. A read access to Packet RAM always starts at the bottom of the RAM, i.e., the read address pointer always starts at the beginning of the data for a given read.

##### 4.11.3.1.1 Receive Packet RAM Read Access Flow

Once receive data has been determined to be in Packet RAM, the following is a typical flow to read access the data:

1. Read rx\_pkt\_latch[6:0] to determine the number of payload bytes in receive Packet RAM. Note that this number includes 2 CRC bytes.
2. Calculate the number of SPI bursts that are required to access the Rx packet data, noting the following:
  - a) The CRC is not normally accessed, so the number of bytes is reduced by 2.

#### NOTE

If the 2-byte CRC data is read. The byte order is reversed (last CRC byte is read first).

- b) All data read during an access must be done on 16-bit or 2-byte boundaries. Therefore, for an odd number of bytes, the byte count must be rounded up to an even number.

- c) The first word (or 2 bytes) read during a Packet RAM read should be discarded as the internal Packet RAM address is not accessed for the first word read operation. This has the effect of adding 2 bytes to the byte count.
3. Do a recursive SPI read transaction where:
- a) MCU asserts  $\overline{CE}$  low.
  - b) MCU sends the MC1321x the first SPI burst with header field of R/ $\overline{W}$  bit = 1 and address field Addr[5:0] = 0x01 for the RX\_Pkt\_RAM register address.
  - c) MCU reads MC1321x data with the number of SPI byte bursts as calculated in Number 2 above. Note that the first two bytes read from the MC1321x are discarded and that the number of SPI read bursts must be an even number. For an odd number of bytes, the one byte is also discarded, and the odd byte is read from the lower 8 bits of the register.
  - d) MCU negates  $\overline{CE}$  high.

#### 4.11.3.1.2 Receive Packet RAM Read Access Error Conditions

Two types of errors can occur during a Packet RAM read:

1. RAM address error - if the recursive read access exceeds 64 words (128 SPI data bursts), the internal read address counter will exceed the RAM address and generate an error indication via status bit ram\_addr\_err, IRQ\_Status Register 24, Bit 14. An interrupt request can be generated with the error status by setting mask bit ram\_addr\_mask, IRQ\_Mask Register 5, Bit 12. As with other interrupt requests, the status is cleared by reading the IRQ\_Status register.
2. RAM arbitration busy - if the transceiver internal logic attempts to access the RAM during a SPI read access (a SPI read during an active Rx sequence), an error indication will be generated via status bit arb\_busy\_err, IRQ\_Status Register 24, Bit 13. An interrupt request can be generated with the error status by setting mask bit arb\_busy\_mask, IRQ\_Mask Register 5, Bit 11. As with other interrupt requests, the status is cleared by reading the IRQ\_Status register.

#### 4.11.3.2 Recursive Transmit Packet RAM Write Access

The transmit Packet RAM is normally accessed when the MC1321x is in Packet Data Mode and a frame is to be transmitted. The number of transmit data bytes in the transmit queue is loaded into the tx\_pkt\_length[6:0] field (this represents the full payload which includes the data bytes stored in the transmit Packet RAM plus the 2 CRC bytes).

The data is written to the TX\_Pkt\_RAM Register 02 with a recursive access. When accessing TX\_Pkt\_RAM Register 02, the SPI register address pointer is NOT incremented, instead, the Packet RAM write address pointer is incremented. Therefore, by using a recursive write, up to 64 words of packet memory can be written via the SPI with an access that requires but a single header field. A write access to Packet RAM always starts at the bottom of the RAM, i.e., the write address pointer always starts at the beginning of the data for a given write.



#### 4.11.3.2.1 Transmit Packet RAM Write Access Flow

Before data is actually written to the Tx Packet RAM, the Tx payload length must be written into field `tx_pkt_length[6:0]`, `TX_Pkt_Ctl` Register 03, Bit 6 - 0. The maximum length is 127 bytes and is the number of actual payload bytes transmitted which includes 2 CRC bytes. The CRC bytes transmitted are generated by the transceiver hardware and are not loaded into the Packet RAM.

The following is a typical flow to write data to the Packet RAM:

1. Determine which of the two Transmit Packet RAMs are to be used - If Transmit Packet RAM2 is to be used, set status bit `tx_ram2_select`, `TX_Pkt_Ctl` Register 03, Bit 15. The default is Transmit Packet RAM1 selected. Note that the `tx_ram2_select` status determines which transmit Packet RAM is accessed by an SPI transaction as well as which RAM is used during Transmit Mode.
2. Calculate the number of SPI bursts that are required to write the Tx packet data, noting the following:
  - a) The CRC bytes are not written to Transmit Packet RAM.
  - b) The maximum number of Tx packet data bytes is 125.
  - c) All data written during an access must be done on 16-bit or 2-byte boundaries. Therefore, for an odd number of bytes, the byte count must be rounded up to an even number and an extra dummy byte will be written.
3. Do a recursive SPI write transaction where:
  - a) MCU asserts  $\overline{CE}$  low.
  - b) MCU sends the MC1321x the first SPI burst with header field of  $R/\overline{W}$  bit = 0 and address field `Addr[5:0] = 0x02` for the `TX_Pkt_RAM` register address.
  - c) MCU writes the MC1321x data with the number of SPI byte bursts as calculated in Step 2. The number of SPI write bursts must be an even number.
  - d) MCU negates  $\overline{CE}$  high.

#### 4.11.3.2.2 Transmit Packet RAM Write Access Error Conditions

Two types of errors can occur during a Packet RAM write:

1. RAM address error - This can occur during software development and debug. If the recursive write access exceeds 64 words (128 SPI data bursts), the internal read address counter will exceed the RAM address and generate an error indication via status bit `ram_addr_err`, `IRQ_Status` Register 24, Bit 14. An interrupt request can be generated with the error status by setting mask bit `ram_addr_mask`, `IRQ_Mask` Register 5, Bit 12. As with other interrupt requests, the status is cleared by reading the `IRQ_Status` register.
2. RAM arbitration busy - if the transceiver internal logic attempts to access the RAM during an SPI write access (an SPI access during an active Tx sequence), an error indication will be generated via status bit `arb_busy_err`, `IRQ_Status` Register 24, Bit 13. An interrupt request can be generated with the error status by setting mask bit `arb_busy_mask`, `IRQ_Mask` Register 5, Bit 11. As with other interrupt requests, the status is cleared by reading the `IRQ_Status` register.

## 4.12 Modem Program Reset (Writing Address 0x00)

A special access is a modem software reset capability known as a “Software Reset”. When R/W Register Address 0x00 is written, an internal chip reset of the digital core is generated. All synchronous logic in the MC1321x digital core is reset and the SPI register fields are returned to their default values. This Software Reset has the same effect on the MC1321x digital core as asserting the external  $\overline{\text{RST}}$  pin except RAM contents are retained.

The Software Reset asserts internally as soon as the 8-bit header field containing Address 0 is shifted into the MOSI pin and a write operation is specified. The Software Reset remains asserted internally until the  $\overline{\text{CE}}$  pin is negated. Reading from Register 00 does not generate a reset.

## 4.13 Configuring MCU Registers for Proper SPI Operation

The MCU SPI module must be configured for proper operation to meet the modem SPI transaction format. The following conditions must be met:

- MCU is master
- Maximum baud rate is 8 MHz
- Proper clock format must be selected, i.e., CPHA = 0 and CPOL = 0
- SPI data must be transferred MSB first
- Slave select is required but cannot be toggled as a normal SPI transfer (active one byte at-a-time).  $\overline{\text{SS1}}$  must be controlled as a GPIO pin, i.e., PTE2 to meet the protocol

### 4.13.1 Set SPI Module Mode

#### 4.13.1.1 SPI1CI Register Settings

The following register settings apply:

- SPIE (Bit 7)- Used to control SPI interrupt as required
- SPE (Bit 6)- Used to enable SPI system as required
- SPTIE (Bit 5)- Used to control transmit interrupt as required
- MSTR (Bit 4) = 1 - Master Mode selected
- CPOL (Bit 3) = 0 - Active high SPI clock polarity (default); idles low
- CPHA (Bit 2) = 0 - First edge on SPSCCK occurs at middle of 1st cycle of an 8-cycle data transfer
- SSOE (Bit 1) = 0 -  $\overline{\text{SS1}}$  pin will be controlled as a GPIO (PTE2) (default), and MODFEN bit must be “0”
- LBBFE (Bit 0) = 0 - Serial data is starts with MSB (default)

### 4.13.1.2 SPI1C2 Register settings

The following register settings apply:

- MODFEN (Bit 4) = 0 -  $\overline{SS1}$  reverts to GPIO controlled as PTE2 (default)
- BIDIROE (Bit 3) = 0 - Has no meaning for selected mode (default)
- SPISWAI (Bit 1) - Used to control SPI clock in Wait Mode as required
- SPC0 (Bit 0) - Use separate pins for data in and data out (default)

### 4.13.1.3 GPIO PTE2 Control (Used as $\overline{SS1}$ )

The PTE2/ $\overline{SS1}$  is used to control to the modem  $\overline{CE}$  input, however, the MCU SPI module does not support use of this signal in the manner required by the modem protocol. As a result, the signal is enabled as a GPIO and is controlled through the port E control registers. PTE2 must be programmed as an output and its state set via software to properly bracket SPI byte transactions. For more on control of port E, see [Section 13.3.5, “Port E, SC11, and SPI”](#).

## 4.13.2 SPI Baud Rate Control

The software supporting 802.15.4 Standard applications or ZigBee applications often requires an MCU bus clock of 8 MHz or 16 MHz. The SPI baud rate is generated directly from the bus clock and the maximum baud rate is one half the bus clock rate. This translates to a baud rate of 4 MHz for an 8 MHz bus clock and a baud rate of 8 MHz (max spec limit) for a 16 MHz bus clock. A slower baud rate can be tried, but it may impact performance of the software in transferring data to/from the modem and lengthen the initialization time of the modem.

To get maximum baud rate, the baud rate generator must be set to a minimum divide ratio of 2, i.e., where the prescaler is set to “1” and the clock rate divider is set to “2”. For the SPI baud rate register SPIBR:

- SPPR[2:0] = 0b000 - Prescaler divisor is “1” (default)
- SPR[2:0] = 0b000 - Baud rate divisor is “2” (default)

#### NOTE

The default condition for SPI1BR may be used.



## Chapter 5

# Modem SPI Register Descriptions

### 5.1 Overview

All control, reading of status, writing of data, and reading of data is done through the MC1321x SPI port. The host microcontroller accesses the transceiver through SPI “transactions” in which multiple bursts of byte-long data are transmitted on the SPI bus. Each transaction is three or more bursts long depending on the transaction type, and these are described in detail in [Section 4.9, “Modem SPI Singular Transactions”](#).

Transactions are always read accesses or write accesses to register addresses. The associated data for any single register access is always 16 bits in length. This chapter describes all the registers that users should access in the MC1321x. Undocumented addresses should not be accessed.

#### NOTE

- Register default values for reserved fields should not be modified unless specifically noted. Freescale recommends that all writes to control register fields that only modify part of the 16-bit word be done as a read-modify-write operation.
- The newest version of the MC1321x now uses an updated transceiver device. Although fully compliant with earlier versions of the transceiver, proper performance of the radio requires that the following modem registers be over-programmed:

Register 0x31 to 0xA0C0

Register 0x34 to 0xFEC6

These registers must be over-programmed for MC1321x devices in which the modem Chip\_ID Register 0x2C reads 0x6800.

## 5.2 Register Model and Description Details

Table 5-1 summarizes the MC1321x Register Model and the following sections describe each register in more detail.

**Table 5-1. Modem SPI Register Table**

REGISTER NAME	Add (Hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reset	00	software_reset																
RX_Pkt_RAM	01	rx_pkt_ram[15:0]																
TX_Pkt_RAM	02	tx_pkt_ram[15:0]																
TX_Pkt_Ctl	03	tx_ram2_select											tx_pkt_length[6:0]					
CCA_Thresh	04	cca_vt[7:0]								power_comp[7:0]								
IRQ_Mask	05	attn_mask			ram_addr_mask	arb_busy_mask	strm_data_mask	pll_lock_mask	acomma_en				doze_mask	tmr4_mask	tmr3_mask	tmr2_mask	tmr1_mask	
Control_A	06				tx_strm	rx_strm	cca_mask	tx_sent_mask	rx_rcvd_mask	tmr_trig_en			cca_type[1:0]				xcvr_seq	
Control_B	07	tmr_load	ct_bias_en	ct_bias_inv	RF_switch_mode	miso_hiz_en		clko_doze_en		tx_done_mask	rx_done_mask	use_strm_mode				hib_en	doze_en	
PA_Enable	08	pa_en																
Control_C	09									gpio_alt_en		clko_en				tmr_prescale[2:0]		
CLKO_Ctl	0A	xtal_trim[7:0]													clko_rate[2:0]			

Table 5-1. Modem SPI Register Table (continued)

REGISTER NAME	Add (Hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_Dir	0B	gpio1234_drv[1:0]		gpio7_oen	gpio6_oen	gpio5_oen	gpio4_oen	gpio3_oen	gpio2_oen	gpio1_oen	gpio7_ien	gpio6_ien	gpio5_ien	gpio4_ien	gpio3_ien	gpio2_ien	gpio1_ien
GPIO_Data_Out	0C	gpio567_drv[1:0]		miso_drv[1:0]		clk_drv[1:0]		irqb_drv[1:0]		irqb_pup_en	gpio7_o	gpio6_o	gpio5_o	gpio4_o	gpio3_o	gpio2_o	gpio1_o
LO1_Int_Div	0F	lo1_div[7:0]															
LO1_Num	10	lo1_num[15:0]															
PA_Lvl	12									pa_lv_coarse[1:0]		pa_lv_fine[1:0]		pa_drv_coarse[1:0]		pa_drv_fine[1:0]	
Tmr_Cmp1_A	1B	tmr_cmp1_dis	tmr_cmp1[23:16]														
Tmr_Cmp1_B	1C	tmr_cmp1[15:0]															
Tmr_Cmp2_A	1D	tmr_cmp2_dis	tmr_cmp2[23:16]														
Tmr_Cmp2_B	1E	tmr_cmp2[15:0]															
Tmr_Cmp3_A	1F	tmr_cmp3_dis	tmr_cmp3[23:16]														
Tmr_Cmp3_B	20	tmr_cmp3[15:0]															
Tmr_Cmp4_A	21	tmr_cmp4_dis	tmr_cmp4[23:16]														
Tmr_Cmp4_B	22	tmr_cmp4[15:0]															
TC2_Prime	23	tc2_prime[15:0]															

**Table 5-1. Modem SPI Register Table (continued)**

REGISTER NAME	Add (Hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ_Status	24	pll_lock_irq	ram_addr_err	arb_busy_err	stm_data_err		attn_irq	doze_irq	tmr1_irq	rx_rcvd_irq	tx_sent_irq	cca_irq	tmr3_irq	tmr4_irq	tmr2_irq	cca	crc_valid
RST_Ind	25									reset_ind							
Current_Time_A	26									et[23:16]							
Current_Time_B	27	et[15:0]															
GPIO_Data_In	28		gpio7_i	gpio6_i	gpio5_i	gpio4_i	gpio3_i	gpio2_i	gpio1_i								
Chip_Id	2C	chip_id[8:0]															
RX_Status	2D	cca_final[7:0]									rx_pkt_latch[6:0]						
Timestamp_A	2E									timestamp[23:16]							
Timestamp_B	2F	timestamp[15:0]															
BER_Enable	30	ber_en															
PSM_Mode	31											psm_tm[2:0]					
Reserved	34																



## 5.3 Reset - Register 00

Writing to Reset Register 00 causes a reset condition where the digital logic is reset, but the transceiver is not powered down. The device is forced to the Idle Mode and the SPI registers are all reset and forced to their default condition although all data in the Packet RAMs is retained. The reset is held as long as  $\overline{CE}$  remains asserted and is released when  $\overline{CE}$  is negated high. A read of this register has no effect.

		Register 00														0x00	
BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		software_reset															
TYPE		w															
RESET		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0x0000															

**Table 5-2. Register 00 Description**

Name	Description	Operation
Bits 15-0	<b>software_reset</b> — Writing this register provides a software reset. When there is a SPI write to Register 00, the IC is reset and stays reset as long as CE remains asserted, and returns to normal operation when CE is negated. Read of this register has no effect.	Write data is “don't care”

## 5.4 RX\_Pkt\_RAM - Register 01

The receive Packet RAM register is accessed when the MC1321x is being used in Packet Mode or Stream Mode for data transfer. In Packet Mode once a packet has been received, the payload data is stored in the RX Packet RAM and the length of the packet data is contained in Register 2D, Bit 6-0. A recursive read (see [Section 4.11, “Modem SPI Recursive Transactions”](#)) to the RX\_Pkt\_RAM Register 01 is required to access the RX packet payload data.

In Stream Mode, the receive payload data is accessed word-by-word via repeated read accesses on the SPI bus. During Stream Mode when a valid data word is available in RX\_Pkt\_RAM, a rx\_strm\_irq status is set and an interrupt is generated (it must be enabled). Reading the RX\_Pkt\_RAM register will clear the rx\_strm\_irq interrupt so that reading the IRQ\_Status Register 24 is not required (see [Section 7.3.4, “Stream Mode Data Transfer TX and RX Operation”](#)).

		Register 01														0x01	
BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		rx_pkt_ram[15:0]															
TYPE		r/w															
RESET		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
		unknown from reset															

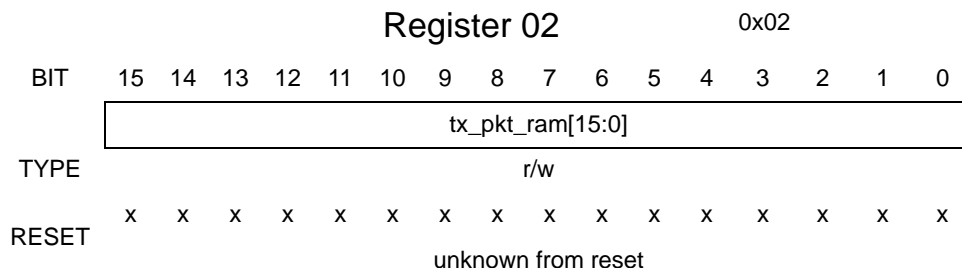
**Table 5-3. Register 01 Description**

Name	Description	Operation
Bits 15-0	<b>rx_pkt_ram[15:0]</b> — These bits are the data channel for host access to the receive Packet RAM.	Default from $\overline{\text{RST}}$ reset is indeterminate.

## 5.5 TX\_Pkt\_RAM - Register 02

The transmit Packet RAM register is accessed when the MC1321x is being used in Packet Mode or Stream Mode for data transfer. There are two transmit Packet RAMs and only one is accessed when the MC1321x is being used in Packet Mode for data transfer. In Packet Mode, the packet payload data must be written to the selected TX Packet RAM and the length of the packet data must be written to TX\_Pkt\_Ctl Register 03, Bits 6-0. A recursive write (see Section 4.11, “Modem SPI Recursive Transactions”) to the TX\_Pkt\_RAM Register 02 is required to load the TX packet payload data.

In Stream Mode, the transmit payload data is written to Register 02 on a word-by-word basis via repeated accesses on the SPI bus. During Stream Mode when a data word is required for TX\_Pkt\_RAM, a tx\_strm\_irq status is set and an interrupt is generated (it must be enabled). Writing the TX\_Pkt\_RAM register will clear the tx\_strm\_irq interrupt so that reading the IRQ\_Status Register 24 is not required (see Section 7.3.4, “Stream Mode Data Transfer TX and RX Operation”).



**Table 5-4. Register 02 Description**

Name	Description	Operation
Bits 15-0	<b>tx_pkt_ram[15:0]</b> — These bits are the data channel for host access to the selected transmit Packet RAM.	Default from reset is indeterminate.

## 5.6 TX\_Pkt\_Ctl - Register 03

The TX\_Pkt\_RAM\_Ctl Register 03 contains two fields that control operation of the two transmit Packet RAMs. The first field tx\_ram2\_select, Bit 15, determines which of the transmit Packet RAMs is selected for present operations. The second field tx\_pkt\_length, Bits 6-0, defines the length of the payload data for a transmission data packet.



**Table 5-5. Register 03 Description**

Name	Description	Operation
Bits 14- 7	Reserved	Leave default
Bit 15	<b>tx_ram2_select</b> — The Transmit RAM select bit selects between transmit Packet RAM1 and transmit Packet RAM2 for operations involving transmit RAM. These include SPI read, SPI write, packet transmission and software error interrupts.	1 = tx_ram2_sel TX Packet RAM2 selected.  0 = tx_ram2_sel TX Packet RAM1 selected.
Bits 6 - 0	<b>tx_pkt_length[6:0]</b> — The Transmit Packet Length bits represent the number of bytes to be transmitted from transmit Packet RAM plus 2 bytes for FCS.	Total transmit payload data length in bytes

## 5.7 CCA\_Thresh - Register 04

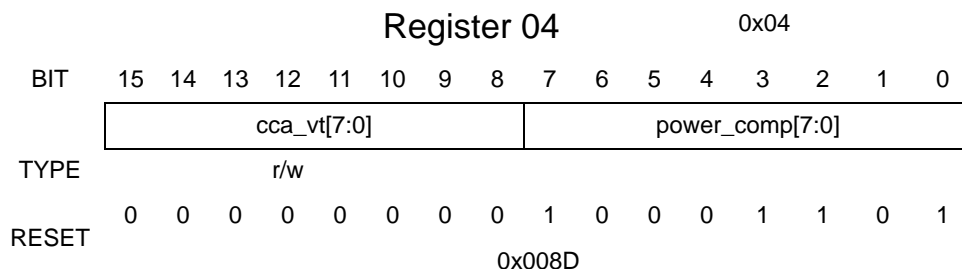
The CCA\_Thresh Register 04 contains the cca\_vt[7:0] 8-bit CCA threshold value, Bits 15 - 8. To calculate desired the cca\_vt[7:0] value:

$$\text{Threshold value} = \text{hex} ( | (\text{Threshold Power in dBm}) * 2 | )$$

A second field is power\_comp[7:0], Bits 7 - 0, which is an offset that is divided by 2 and added to the measured value of the average energy from a CCA/ED function or LQI value from an RX function, and the resulting value is stored in cca\_final[7:0], RX\_Status Register 2D. By using this power\_comp[7:0] value, users can compensate the cca\_final[7:0] value for external gain in the RX path. See [Section 7.3.5, “Clear Channel Assessment \(CCA\) Modes \(including Link Quality Indication\)”](#) for more detailed information.

### NOTE

The default value for power\_comp[7:0] is 0x8D. To center the reported cca\_final[7:0] value over temperature, it is suggested that a value of 0x9B be written to power\_comp[7:0], which equates to a 3.5dBm offset from default.

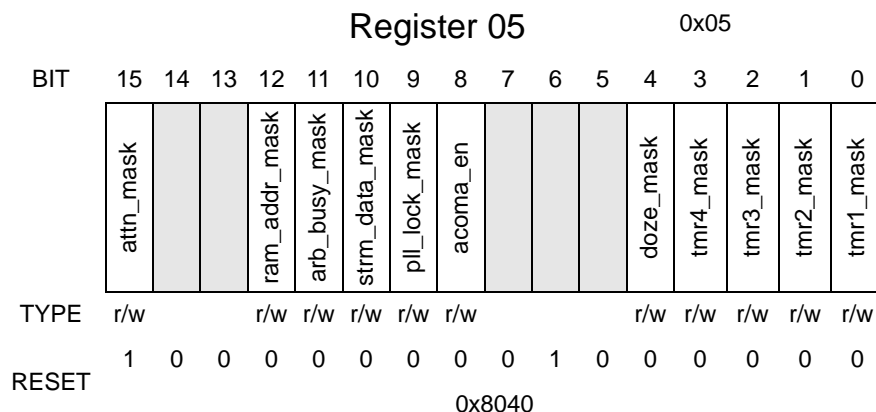


**Table 5-6. Register 04 Description**

Name	Description	Operation
Bits 15-8	<b>cca_vt[7:0]</b> - Threshold value for Clear Channel Assessment in dB-linear format	Default is 0x00.
Bits 7-0	<b>power_comp[7:0]</b> - This is a binary value that is added to the measured value of the CCA operation. The result is stored in cca_final[7:0]	Default is 0x8D

## 5.8 IRQ\_Mask - Register 05

The IRQ\_Mask Register 05 provides most, but not all, mask bits for various interrupt sources for the MC1321x. If a mask bit is set, its associated status bit being true will generate an interrupt on the MC1321x IRQ pin. The interrupt is cleared when the status bit is read via a SPI transaction.



**Table 5-7. Register 05 Description**

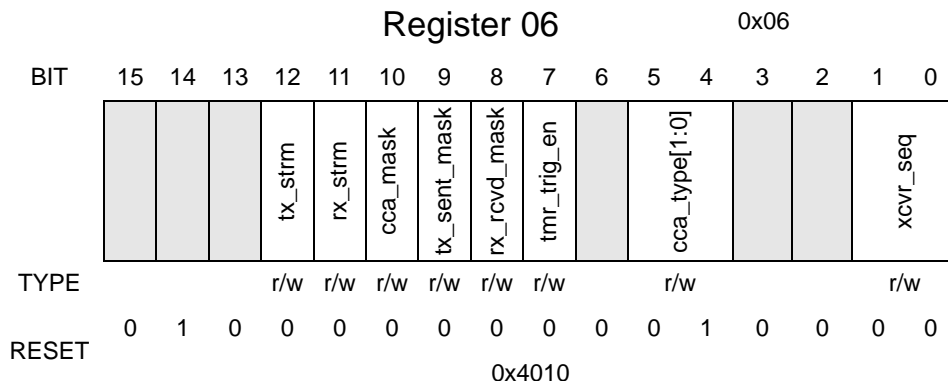
Name	Description	Operation
Bits 14-13, 7-5	Reserved	Leave default
Bit 15	<b>attn_mask</b> — The attention interrupt mask bit controls the attn_irq interrupt on the IRQ pin. Default is for the interrupt to be enabled out of reset.	1 = Allows attn_irq to generate an interrupt on the IRQ pin. 0 = When attn_irq status bit is set, IRQ pin is not asserted.
Bit 12	<b>ram_addr_mask</b> — The Packet RAM address error interrupt mask bit controls the ram_addr_err interrupt on the IRQ pin.	1 = Allows ram_addr_err to generate an interrupt on the IRQ. 0 = When ram_addr_err status bit is set, IRQ pin is not asserted.

**Table 5-7. Register 05 Description (continued)**

Name	Description	Operation
Bit 11	<b>arb_busy_mask</b> — The Packet RAM arbiter busy error interrupt mask bit controls the arb_busy_err interrupt on the IRQ pin.	1 = Allows arb_busy_err to generate an interrupt on the IRQ pin. 0 = When arb_busy_err status bit is set, IRQ pin is not asserted.
Bit 10	<b>strm_data_mask</b> — The Stream Mode data error interrupt mask bit controls the strm_data_irq interrupt on the IRQ pin.	1 = Allows strm_data_err to generate an interrupt on the IRQ pin. 0 = When strm_data_err status bit is set, IRQ pin is not asserted.
Bit 9	<b>pll_lock_mask</b> — The LO1 unlock detect mask bit controls the pll_lock_irq interrupt on the IRQ pin.	1 = Allows pll_lock_irq to generate an interrupt on the IRQ pin. 0 = When pll_lock_irq status bit is set, IRQ pin is not asserted.
Bit 8	<b>acoma_en</b> — The Acoma Mode enable bit controls Doze Mode. Acoma is an enhanced power save mode within Doze.	1 = The MC1321x stays in Doze until $\overline{ATTN}$ asserted. Event Timer and Prescaler clocks disabled for additional current savings. 0 = Normal operation. Doze is exited by TC2 match or $\overline{ATTN}$ assertion.
Bit 4	<b>doze_mask</b> — The Doze timer interrupt mask bit controls the doze_irq interrupt on the IRQ pin.	1 = Allows doze_irq to generate an interrupt on the IRQ pin. 0 = When doze_irq status bit is set, IRQ pin is not asserted.
Bit 3	<b>tmr4_mask</b> — The Event Timer four interrupt mask bit controls the tmr4_irq interrupt on the IRQ pin.	1 = Allows tmr4_irq to generate an interrupt on the IRQ pin. 0 = When tmr4_irq status bit is set, IRQ pin is not asserted.
Bit 2	<b>tmr3_mask</b> — The Event Timer three interrupt mask bit controls the tmr3_irq interrupt on the IRQ pin.	1 = Allows tmr3_irq to generate an interrupt on the IRQ pin. 0 = When tmr3_irq status bit is set, IRQ pin is not asserted.
Bit 1	<b>tmr2_mask</b> — The Event Timer two interrupt mask bit controls the tmr2_irq interrupt on the IRQ pin.	1 = Allows tmr2_irq to generate an interrupt on the IRQ pin. 0 = When tmr2_irq status bit is set, IRQ pin is not asserted.
Bit 0	<b>tmr1_mask</b> — The Event Timer one interrupt mask bit controls the tmr1_irq interrupt on the IRQ pin.	1 = Allows tmr1_irq to generate an interrupt on the IRQ pin. 0 = When tmr1_irq status bit is set, IRQ pin is not asserted.

## 5.9 Control\_A - Register 06

The Control\_A Register 06 is one of several registers that provide control fields for the MC1321x.



**Table 5-8. Register 06 Description**

Name	Description	Operation
Bits 15-13, 6,3,2	<b>Reserved</b>	Leave default
Bit 12	<b>tx_strm</b> — The Transmit Stream Mode bit enables Transmit Streaming Data Transfer Mode	1 = Real-time streaming of TX data to transceiver via SPI bus and IRQ on word-by-word basis. 0 = TX Packet Mode with data preloaded in TX RAM
Bit 11	<b>rx_strm</b> — The receive Stream Mode bit enables Receive Streaming Data Transfer Mode	1 = Real-time streaming of RX data from transceiver via SPI bus and IRQ on word-by-word basis. 0 = RX Packet Mode with data preloaded in RX RAM
Bit 10	<b>cca_mask</b> — The CCA interrupt mask bit controls the cca_irq interrupt on the IRQ pin.	1 = Allows the cca_irq to generate an interrupt on the IRQ pin. 0 = When icca_irq status bit is set, IRQ pin is not asserted.
Bit 9	<b>tx_sent_mask</b> — The transmit sent mask bit controls the tx_sent_irq interrupt on the IRQ pin.	1 = Allows the tx_sent_irq to generate an interrupt on the IRQ pin. 0 = When tx_sent_irq status bit is set, IRQ pin is not asserted.
Bit 8	<b>rx_rcvd_mask</b> — The packet received interrupt mask bit controls the rx_rcvd_irq interrupt on the IRQ pin.	1 = Allows rx_rcvd_irq to generate an interrupt on the IRQ pin. 0 = When rx_rcvd_irq status bit is set, IRQ pin is not asserted.
Bit 7	<b>tmr_trig_en</b> — The timer trigger enable bit determines whether transceiver operation is initiated via Timer Comparator 2 or manually.	1 = The selected transceiver operation is initiated via Timer Comparator 2, or TC2_Prime. 0 = The selected transceiver operation is initiated manually via the RXTXEN pin and by SPI programming.

**Table 5-8. Register 06 Description (continued)**

Name	Description	Operation
Bits 5-4	<b>cca_type[1:0]</b> — The clear channel assessment type bits selects one of two possible CCA functions. Algorithm results are reported in field <code>cca_final[7:0]</code> , RX_Status Register 2D, Bits 15 - 8.	01 = clear channel assessment 10 = energy detection
Bits 1-0	<b>xcvr_seq[1:0]</b> — The transceiver operation bits select one of four possible transceiver modes.	00 = Idle (default) 01 = CCA /energy detection 10 = Packet Mode RX 11 = Packet Mode TX

## 5.10 Control\_B - Register 07

The Control\_B Register 07 is one of several registers that provide control fields for the MC1321x.

	Register 07														0x07	
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	tmr_load	ct_bias_en	ct_bias_inv	RF_switch_mode	miso_hiz_en		clko_doze_en		tx_done_mask	rx_done_mask	use_stirm_mode				hib_en	doze_en
TYPE	r/w	r/w	r/w	r/w	r/w		r/w		r/w	r/w	r/w				r/w	r/w
RESET	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
	0x0C00															

**Table 5-9. Register 07 Description**

Name	Description	Operation
Bits 14-12, 10, 8, 4-2	Reserved	Leave default
Bit 15	<b>tmr_load</b> — The load Event Timer bit, when programmed from low to high, causes the value of the SPI field <code>tmr_cmp1[23:0]</code> to be loaded into the Event Timer.	Write from 0 to 1 to affect load. Rewrite to 0, before writing another 1 to affect another load.
Bit 14	<b>ct_bias_en</b> — The CT_Bias enable bit controls when the signal CT_Bias is active.	1 = CT_Bias enabled. Output state is defined by <a href="#">Table 3-5</a> . 0 = CT_Bias disabled. Output state is tristated.
Bit 13	<b>ct_bias_inv</b> — The CT_Bias Invert bit inverts the sense of the CT_Bias output signal when the CT_Bias is used as a control signal for dual port operation	The output state of CT_Bias under varying conditions is defined in <a href="#">Table 3-5</a> . 1 = CT_Bias inverted. 0 = CT_Bias not inverted

**Table 5-9. Register 07 Description (continued)**

Name	Description	Operation
Bit 12	<b>RF_switch_mode</b> — This bit selects the mode of operation of the RF interface, i.e., between Dual Port Mode and Single Port Mode.	1 = Single Port Mode selected where RF switch is active and RFIN_M and RFIN_P and bidirectional signals. 0 = Dual Port Mode selected where RFIN_M and RFIN_P are inputs only and PAO_P and PAO_N are separate outputs. (This is default operation).
Bit 11	<b>miso_hiz_en</b> — The MISO high impedance enable bit either tristates or drives the MISO pin to a logic low when CE is negated.	1 = MISO pin is tristated when $\overline{CE}$ is negated (default). 0 = MISO pin is driven to a logic low when $\overline{CE}$ is negated. It is recommended to program <b>miso_hiz_en = 0 for low power modes.</b>
Bit 9	<b>clko_doze_en</b> — The CLKO enable in Doze Mode bit controls toggling of the CLKO pin during Doze Mode.	1 = The CLKO pin continues to toggle at selected rate during Doze Mode if CLKO is enabled (clko_en = 1). 0 = The CLKO pin stops toggling 128 xtal or reference cycles after the <b>doze_en bit</b> is programmed to 1. <b>Note:</b> In Doze Mode only, CLKO frequencies of 1.0 MHz or less are available.
Bit 7	<b>tx_done_mask</b> — The Stream Mode transmit done interrupt mask bit controls the tx_done_irq interrupt.	1 = Allows the tx_done_irq to generate an interrupt on the IRQ pin. 0 = The tx_done_irq status bit is set, but IRQ pin is not asserted. <b>Note:</b> tx_done_irq replaces the ram_addr_err when SPI bit use_strm_mode, Register 7, Bit 5 = 1.
Bit 6	<b>rx_done_mask</b> — The Stream Mode receive done interrupt mask bit controls the rx_done_irq interrupt.	1 = Allows the rx_done_irq to generate an interrupt on the IRQ pin. 0 = The rx_done_irq status bit is set, but IRQ pin is not asserted. <b>Note:</b> The rx_done_irq replaces the arb_busy_err when SPI bit use_strm_mode, Register 7, Bit 5 = 1.
Bit 5	<b>use_strm_mode</b> — The Stream Mode select bit selects Stream Mode or Packet Mode for transmit and receive data handling.	In Packet Mode, data is stored in RAM and handled as a packet. In Stream Mode, data is managed word-by-word through the SPI. 0 = Packet Mode 1 = Stream Mode

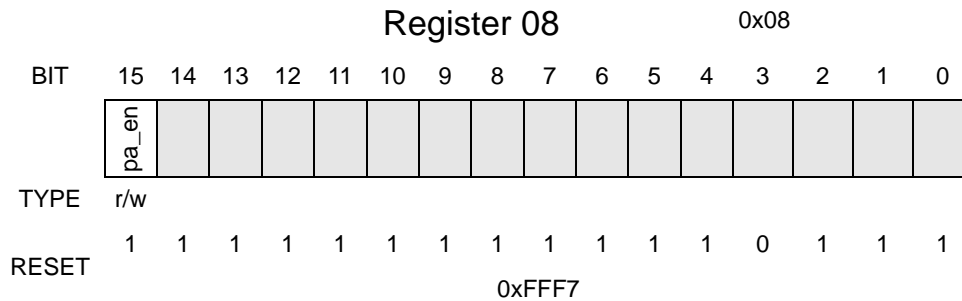


**Table 5-9. Register 07 Description (continued)**

Name	Description	Operation
Bit 1	<b>hib_en</b> — The hibernate enable bit can set the MC1321x into its lowest power saving mode without a running time base.	1 = Places the MC1321x into its lowest power operating mode without a running time base. 0 = Normal operation.
Bit 0	<b>doze_en</b> — The doze enable bit can set the MC1321x into its lowest power saving mode with a running time base.	1 = Places the MC1321x into its lowest power operating mode with a running time base. 0 = Normal operation.

## 5.11 PA\_Enable - Register 08

The PA\_Enable Register 08 contains the power amp (PA) enable bit which turns on and off the transmitter outputs. This feature is useful for sequences in RF test configurations. The default condition is with the transmitter enabled.

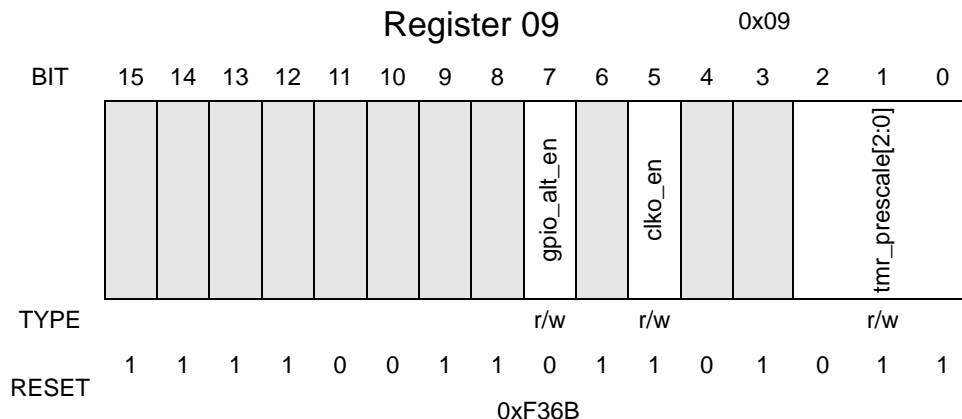


**Table 5-10. Register 08 Description**

Name	Description	Operation
Bits 14-0	Reserved	
Bit 15	<b>pa_en</b> — The power amp (PA) enable bit controls the transmitter PA outputs.	1 = PA enabled. 0 = PA disabled; the transmitter puts out no power.

## 5.12 Control\_C - Register 09

The Control\_C Register 09 is one of several registers that provide control fields for the MC1321x.



**Table 5-11. Register 09 Description**

Name	Description	Operation
Bits 15-8, 6, 4, 3	Reserved	Leave default
Bit 7	<b>gpio_alt_en</b> — The GPIO alternative MCU interface enable bit controls GPIO1 and GPIO2.	1 = GPIO1 and GPIO2 are used as status to the MCU. GPIO1 indicates when the MC1321x is out of idle. GPIO2 indicates a valid CRC or CCA result. These signals are required for Freescale's 802.15.4 MAC software. 0 = Normal GPIO operation.
Bit 5	<b>clko_en</b> — This bit enables the clock output which can be used as a reference clock for the MCU. Frequency is dependent on the value of clko_rate[2:0] (Register 0A, Bits 2 - 0).	1 = Output enabled (default). 0 = Output low
Bits 2-0	<b>tmr_prescale[2:0]</b> — The Event Timer prescale value bits select the frequency of the base clock for the Event Timer module.	See <a href="#">Chapter 6, "Modem Timer Information"</a> for more information.

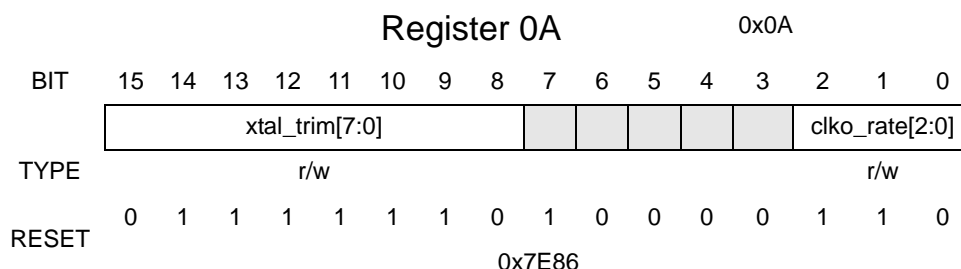
## 5.13 CLKO\_Ctl - Register 0A

The MC1321x provides the ability to trim the crystal oscillator frequency and an output clock with a programmable frequency that can be used to drive another device, such as a microcontroller. The field `xtal_trim[7:0]`, CLKO\_Ctl Register 0A, Bits 15-8, alter the capacitive loading to the crystal and affects the oscillator frequency. See [Chapter 9, “Modem Miscellaneous Functions”](#).

The CLKO frequency is programmed using `clko_rate[2:0]`, CLKO\_Ctl Register 0A, Bits 2-0. [Table 5-12](#) lists each setting and its respective frequency.

**Table 5-12. CLKO Frequency**

clko_rate	CLKO
000	16 MHz
001	8 MHz
010	4 MHz
011	2 MHz
100	1 MHz
101	62.5 kHz
110 (default)	32.786+ kHz = 16 MHz / 488
111	16.393+ kHz = 16 MHz / 976



**Table 5-13. Register 0A Description**

Name	Description	Operation
Bits 7-3	Reserved	Leave default
Bits 15-8	<b>xtal_trim[7:0]</b> — The crystal oscillator capacitor trim bits warps the crystal frequency by approximately -0.25 ppm per bit.	0x7E is the default setting and suggested start point for trimming.
Bits 2-0	<b>clko_rate[2:0]</b> — The CLKO rate bits select the clock frequency of the CLKO pin.	See <a href="#">Table 5-12</a> .

## 5.14 GPIO\_Dir - Register 0B

The GPIO\_Dir Register 0B contains control bits for GPIO1 through GPIO7 that configure the data direction of each GPIO as well as a control field that sets the output drive strength of GPIO1 through GPIO4. Each GPIO bit has a corresponding bit to set the GPIO as an output and a separate corresponding bit to set the GPIO as an input. If both enable bits are set simultaneously for a given GPIO, the GPIO is configured as an input (input setting wins). During a hardware reset on  $\overline{RST}$ , all the GPIO are tristated and the GPIO default to inputs.

		Register 0B										0x0B						
BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		gpio1234_drv[1:0]		gpio7_oen	gpio6_oen	gpio5_oen	gpio4_oen	gpio3_oen	gpio2_oen	gpio1_oen	gpio7_ien	gpio6_ien	gpio5_ien	gpio4_ien	gpio3_ien	gpio2_ien	gpio1_ien	
TYPE		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	
RESET		0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
		0x007F																

**Table 5-14. Register 0B Description**

Name	Description	Operation
Bits 15-14	<b>gpio1234_drv[1:0]</b> — These bits select output drive strength for GPIO1 through GPIO4.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bit 13	<b>gpio7_oen</b> — This bit configures GPIO7 as an output.	1 = GPIO7 enabled as output. 0 = GPIO7 disabled as output.
Bit 12	<b>gpio6_oen</b> — This bit configures GPIO6 as an output.	1 = GPIO6 enabled as output. 0 = GPIO6 disabled as output.
Bit 11	<b>gpio5_oen</b> — This bit configures GPIO5 as an output.	1 = GPIO5 enabled as output. 0 = GPIO5 disabled as output.
Bit 10	<b>gpio4_oen</b> — This bit configures GPIO4 as an output.	1 = GPIO4 enabled as output. 0 = GPIO4 disabled as output.
Bit 9	<b>gpio3_oen</b> — This bit configures GPIO3 as an output.	1 = GPIO3 enabled as output. 0 = GPIO3 disabled as output.
Bit 8	<b>gpio2_oen</b> — This bit configures GPIO2 as an output.	1 = GPIO2 enabled as output. 0 = GPIO2 disabled as output.
Bit 7	<b>gpio1_oen</b> — This bit configures GPIO1 as an output.	1 = GPIO1 enabled as output. 0 = GPIO1 disabled as output.
Bit 6	<b>gpio7_ien</b> — This bit configures GPIO7 as an input.	1 = GPIO7 enabled as input. 0 = GPIO7 disabled as input.
Bit 5	<b>gpio6_ien</b> — This bit configures GPIO6 as an input.	1 = GPIO6 enabled as input. 0 = GPIO6 disabled as input.

**Table 5-14. Register 0B Description (continued)**

Name	Description	Operation
Bit 4	<b>gpio5_ien</b> — This bit configures GPIO5 as an input.	1 = GPIO5 enabled as input. 0 = GPIO5 disabled as input.
Bit 3	<b>gpio4_ien</b> — This bit configures GPIO4 as an input.	1 = GPIO4 enabled as input. 0 = GPIO4 disabled as input.
Bit 2	<b>gpio3_ien</b> — This bit configures GPIO3 as an input.	1 = GPIO3 enabled as input. 0 = GPIO3 disabled as input.
Bit 1	<b>gpio2_ien</b> — This bit configures GPIO2 as an input.	1 = GPIO2 enabled as input. 0 = GPIO2 disabled as input.
Bit 0	<b>gpio1_ien</b> — This bit configures GPIO1 as an input.	1 = GPIO1 enabled as input. 0 = GPIO1 disabled as input.

## 5.15 GPIO\_Data\_Out - Register 0C

The GPIO\_Data\_Out Register 0C contains a bit for each GPIO that sets its output value if the GPIO is configured as an output as well as a control fields that set the output drive strength of GPIO5 through GPIO7, MISO, CLKO, and  $\overline{\text{IRQ}}$ . This register also contains the pullup enable for the  $\overline{\text{IRQ}}$  pin.

Register 0C														0x0C		
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	gpio567_drv[1:0]		miso_drv[1:0]		clko_drv[1:0]		irqb_drv[1:0]		irqb_pup_en	gpio7_o	gpio6_o	gpio5_o	gpio4_o	gpio3_o	gpio2_o	gpio1_o
TYPE	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
	0x0380															

**Table 5-15. Register 0C Description**

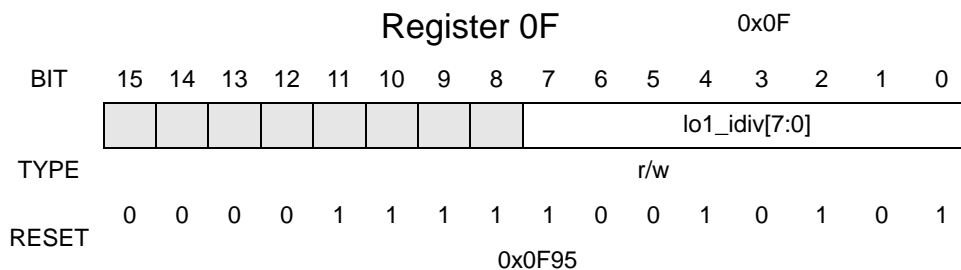
Name	Description	Operation
Bits 15-14	<b>gpio567_drv[1:0]</b> — These bits select output drive strength for GPIO5 through GPIO7.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bits 13-12	<b>miso_drv[1:0]</b> - These bits select output drive strength for signal MISO.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bits 11-10	<b>clko_drv[1:0]</b> - These bits select output drive strength for signal CLKO.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bits 9-8	<b>irqb_drv[1:0]</b> - These bits select output drive strength for signal $\overline{\text{IRQ}}$ .	00 = lowest drive strength; 11 (default) = highest drive strength.

**Table 5-15. Register 0C Description (continued)**

Name	Description	Operation
Bit 7	<b>irqb_pup_en</b> — $\overline{\text{IRQ}}$ pullup enable.	1 = Onboard pullup enabled (nominal 40 k $\Omega$ ) on $\overline{\text{IRQ}}$ pin (default) 0 = Open drain only (external pullup required).
Bit 6	<b>gpio7_o</b> — GPIO7 output value.	1 = GPIO7 driven high 0 = GPIO7 driven low
Bit 5	<b>gpio6_o</b> — GPIO6 output value.	1 = GPIO6 driven high 0 = GPIO6 driven low
Bit 4	<b>gpio5_o</b> — GPIO5 output value.	1 = GPIO5 driven high 0 = GPIO5 driven low
Bit 3	<b>gpio4_o</b> — GPIO4 output value.	1 = GPIO4 driven high 0 = GPIO4 driven low
Bit 2	<b>gpio3_o</b> — GPIO3 output value.	1 = GPIO3 driven high 0 = GPIO3 driven low
Bit 1	<b>gpio2_o</b> — GPIO2 output value.	1 = GPIO2 driven high 0 = GPIO2 driven low
Bit 0	<b>gpio1_o</b> — GPIO1 output value.	1 = GPIO1 driven high 0 = GPIO1 driven low

## 5.16 LO1\_Int\_Div - Register 0F

The LO1\_Int\_Div Register 0F contains the 8-bit integer divide value for the LO1 fractional-N synthesizer that sets transceiver channel frequency. See [Table 5-18](#).

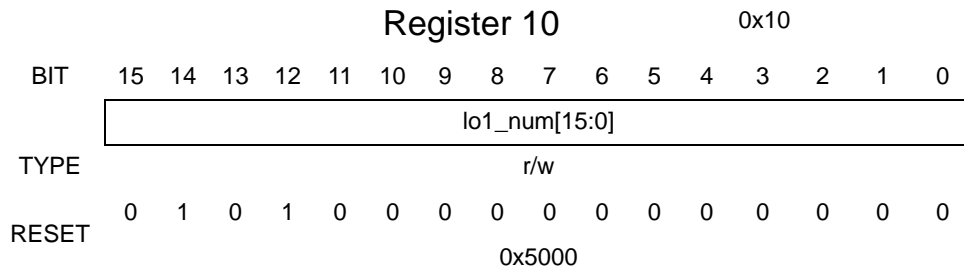


**Table 5-16. Register 0F Description**

Name	Description	Operation
Bits 15-8	Reserved.	Leave default.
Bits 7-0	<b>lo1_idiv[7:0]</b> — The LO1 integer divide bits represent the integer divide value for the LO1 fractional-N synthesizer.	Default is 149dec (0x95). See <a href="#">Table 5-18</a> .

## 5.17 LO1\_Num - Register 10

The LO1\_Num Register 10 contains the 16-bit integer numerator value for the LO1 fractional-N synthesizer that sets transceiver channel frequency. See [Table 5-18](#).



**Table 5-17. Register 10 Description**

Name	Description	Operation
Bits 15-0	<b>lo1_num[15:0]</b> — These bits represent the numerator of the fractional divide value for the LO1 fractional-N synthesizer.	Default is 20,480dec (0x5000). See <a href="#">Table 5-18</a> .

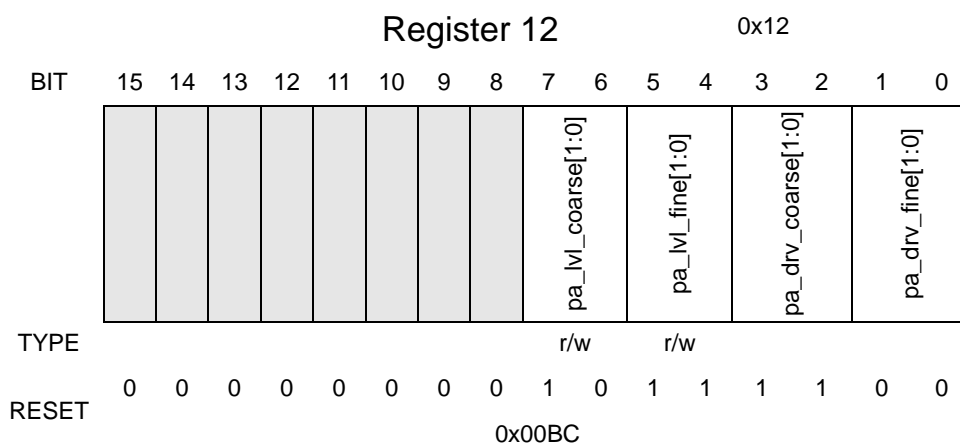
**Table 5-18. Channel Operation<sup>1</sup>**

802.15.4 Channel Number	Frequency (MHz)	Integer Setting lo1_idiv[7:0] (Dec / Hex)	Fractional Setting lo1_num[15:0] (Dec / Hex)
11	2405	149 / 0x95 (default)	20480 / 0x5000 (default)
12	2410	149 / 0x95	40960 / 0xA000
13	2415	149 / 0x95	61440 / 0xF000
14	2420	150 / 0x96	16384 / 0x4000
15	2425	150 / 0x96	36864 / 0x9000
16	2430	150 / 0x96	57344 / 0xE000
17	2435	151 / 0x97	12288 / 0x3000
18	2440	151 / 0x97	32768 / 0x8000
19	2445	151 / 0x97	53248 / 0xD000
20	2450	152 / 0x98	8192 / 0x2000
21	2455	152 / 0x98	28672 / 0x7000
22	2460	152 / 0x98	49152 / 0xC000
23	2465	153 / 0x99	4096 / 0x1000
24	2470	153 / 0x99	24576 / 0x6000
25	2475	153 / 0x99	45056 / 0xB000
26	2480	154 / 0x9A	0 / 0x0000

<sup>1</sup> See [Section 7.4, “Frequency of Operation”](#) for frequency programming equations

## 5.18 PA\_Lvl - Register 12

The PA\_Lvl Register 12 sets the power level and drive level of the transmitter power amplifier. See [Table 7-3](#) for power level versus field settings.



**Table 5-19. Register 12 Description**

Name	Description	Operation
Bits 15-8	Reserved	Leave default
Bits 7-6	<b>pa_lvl_coarse[1:0]</b> - These bits select the coarse PA power level adjustment.	See <a href="#">Section 7.5, "Transmit Power Adjustment"</a> for more information.
Bits 5-4	<b>pa_lvl_fine[1:0]</b> - These bits selects fine PA power level adjustment.	See <a href="#">Section 7.5, "Transmit Power Adjustment"</a> for more information.
Bits 3-2	<b>pa_drv_coarse[1:0]</b> - These bits select the coarse PA drive level adjustment.	Leave default
Bits 1-0	<b>pa_drv_fine[1:0]</b> - These bits select the fine PA drive level adjustment	Leave default

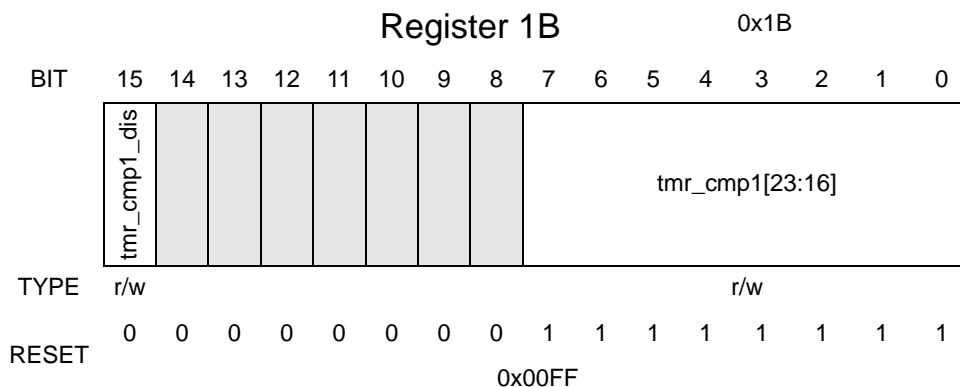
## 5.19 Tmr\_Cmp1\_A - Register 1B

The Tmr\_Cmp1\_A Register 1B contains the disable bit for Timer Comparator 1 and stores the most significant 8 bits of the 24-bit compare value (the lower 16 bits of the compare value are stored in Tmr\_CMP1\_B Register 1C). With regard to using the timer comparator:

- It is suggested that the timer be disabled (writing **tmr\_cmp1\_dis** to 1) during system initialization as the default mode out of reset is timer enabled.
- The value in Register 1B will not be loaded in the comparator and the affect of the timer disable bit will not active until Register 1C is written. The 24-bit comparator value must be loaded into the comparator register in parallel, and as a result, the load is caused by a write Register 1C.

Writing of Registers 1B and 1C can be done as a recursive SPI register write transaction or two separate singular transactions. Writing to Register 1C as a separate operation will always use the present value of Register 1B to load the comparator value and set the state of the disable bit.

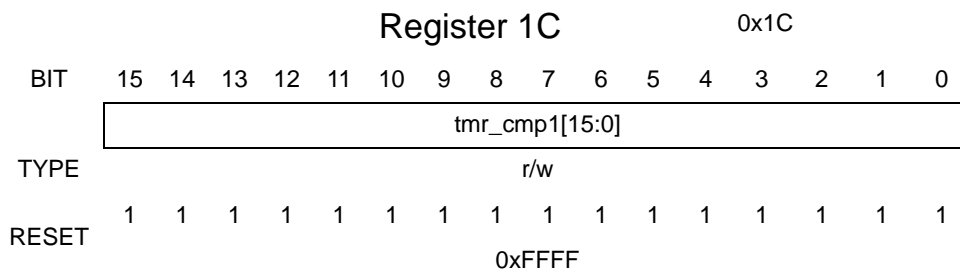



**Table 5-20. Register 1B Description**

Name	Description	Operation
Bits 14-8	Reserved	Leave default
Bit 15	<b>tmr_cmp1_dis</b> — This bit disables the Event Timer Comparator 1 function.	1 = Disables the Event Timer Compare 1 function. 0 = Enables the Event Timer Compare 1 function (default).
Bits 7-0	<b>tmr_cmp1[23:16]</b> — These bits represent the 8 most significant bits of 24-bit Event Timer 1 absolute time compare value, tmr_cmp1[23:0].	Default is 0xFF.

## 5.20 Tmr\_Cmp1\_B - Register 1C

The Tmr\_CMP1\_B Register 1C stores the least significant 16 bits of the 24-bit compare value. Writing to Register 1C causes an internal load of the full 24-bit comparator value (see [Section 5.19, “Tmr\\_Cmp1\\_A - Register 1B”](#)) and activates the mode presently set into the tmr\_cmp1\_dis control bit.


**Table 5-21. Register 1C Description**

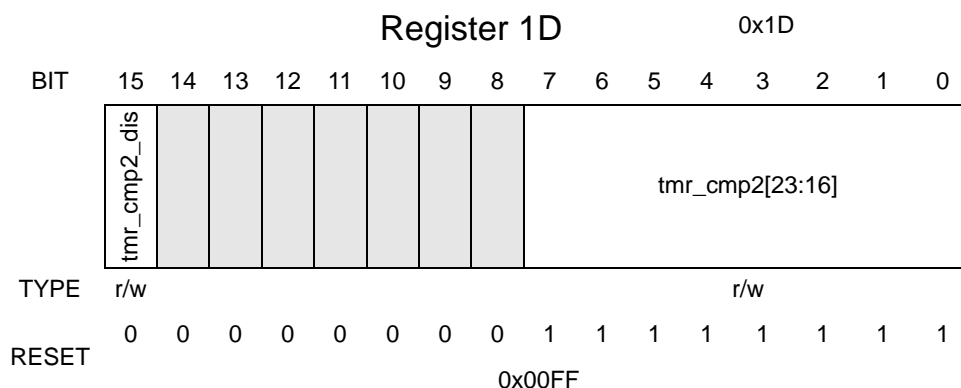
Name	Description	Operation
Bits 15-0	<b>tmr_cmp1[15:0]</b> — These bits represent the 16 least significant bits of 24-bit Event Timer 1 absolute time compare value, tmr_cmp1[23:0].	Default is 0xFFFF.

## 5.21 Tmr\_Cmp2\_A - Register 1D

The Tmr\_Cmp2\_A Register 1B contains the disable bit for Timer Comparator 2 and stores the most significant 8 bits of the 24-bit compare value (the lower 16 bits of the compare value are stored in Tmr\_CMP2\_B Register 1E). With regard to using the timer comparator:

- It is suggested that the timer be disabled (writing **tmr\_cmp2\_dis** to 1) during system initialization as the default mode out of reset is timer enabled.
- The value in Register 1D will not be loaded in the comparator and the affect of the timer disable bit will not active until Register 1E is written. The 24-bit comparator value must be loaded into the comparator register in parallel, and as a result, the load is caused by a write Register 1E.

Writing of Registers 1D and 1E can be done as a recursive SPI register write transaction or two separate singular transactions. Writing to Register 1E as a separate operation will always use the present value of Register 1D to load the comparator value and set the state of the disable bit.



**Table 5-22. Register 1D Description**

Name	Description	Operation
Bits 14-8	Reserved	Leave default
Bit 15	<b>tmr_cmp2_dis</b> — This bit disables the Event Timer Comparator 2.	1 = Disables the Event Timer Compare 2 function. 0 = Enables the Event Timer Compare 2 function (default).
Bits 7-0	<b>tmr_cmp2[23:16]</b> — These bits represent the 8 most significant bits of 24-bit Event Timer 2 absolute time compare value, tmr_cmp2[23:0].	Default is 0xFF.

## 5.22 Tmr\_Cmp2\_B - Register 1E

The Tmr\_CMP2\_B Register 1E stores the least significant 16 bits of the 24-bit compare value. Writing to Register 1E causes an internal load of the full 24-bit comparator value (see [Section 5.21, “Tmr\\_Cmp2\\_A - Register 1D”](#)) and activates the mode presently set into the tmr\_cmp2\_dis control bit.



**Table 5-23. Register 1E Description**

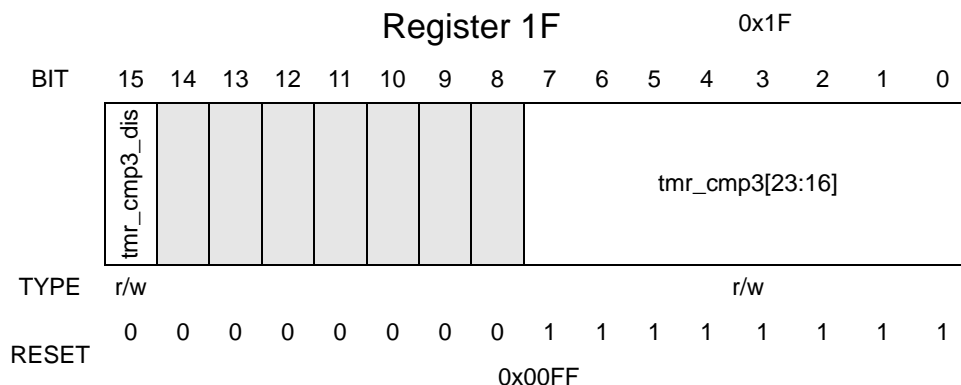
Name	Description	Operation
Bits 15-0	<b>tmr_cmp2[15:0]</b> — These bits represent the 16 least significant bits of 24-bit Event Timer 2 absolute time compare value, tmr_cmp2[23:0].	Default is 0xFFFF.

## 5.23 Tmr\_Cmp3\_A - Register 1F

The Tmr\_Cmp3\_A Register 1F contains the disable bit for Timer Comparator 3 and stores the most significant 8 bits of the 24-bit compare value (the lower 16 bits of the compare value are stored in Tmr\_CMP3\_B Register 20). With regard to using the timer comparator:

- It is suggested that the timer be disabled (writing **tmr\_cmp3\_dis** to 1) during system initialization as the default mode out of reset is timer enabled
- The value in Register 1F will not be loaded in the comparator and the affect of the timer disable bit will not active until Register 20 is written. The 24-bit comparator value must be loaded into the comparator register in parallel, and as a result, the load is caused by a write Register 20

Writing of Registers 1F and 20 can be done as a recursive SPI register write transaction or two separate singular transactions. Writing to Register 20 as a separate operation will always use the present value of Register 1F to load the comparator value and set the state of the disable bit.



**Table 5-24. Register 1F Description**

Name	Description	Operation
Bits 14-8	Reserved	Leave default
Bit 15	<b>tmr_cmp3_dis</b> — This bit disables the Event Timer Comparator 3 interrupt and status bit.	1 = Disables Event Timer Compare 3 function. 0 = Enables Event Timer Compare 3 function (default).
Bits 7-0	<b>tmr_cmp3[23:16]</b> — These bits represent the 8 most significant bits of 24-bit Event Timer 3 absolute time compare value, tmr_cmp3[23:0].	Default is 0xFF.

## 5.24 Tmr\_Cmp3\_B - Register 20

The Tmr\_CMP3\_B Register 20 stores the least significant 16 bits of the 24-bit compare value. Writing to Register 20 causes an internal load of the full 24-bit comparator value (see Section 5.23, “Tmr\_Cmp3\_A - Register 1F”) and activates the mode presently set into the tmr\_cmp3\_dis control bit.



**Table 5-25. Register 20 Description**

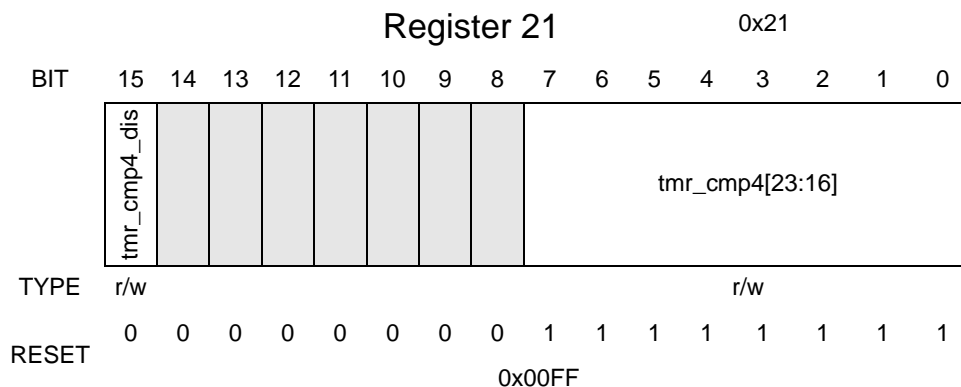
Name	Description	Operation
Bits 15-0	<b>tmr_cmp3[15:0]</b> — These bits represent the 16 least significant bits of 24-bit Event Timer 3 absolute time compare value, tmr_cmp3[23:0].	Default is 0xFFFF.

## 5.25 Tmr\_Cmp4\_A -Register 21

The Tmr\_Cmp4\_A Register 21 contains the disable bit for Timer Comparator 4 and stores the most significant 8 bits of the 24-bit compare value (the lower 16 bits of the compare value are stored in Tmr\_CMP4\_B Register 22). With regard to using the timer comparator:

- It is suggested that the timer be disabled (writing **tmr\_cmp4\_dis** to 1) during system initialization as the default mode out of reset is timer enabled
- The value in Register 21 will not be loaded in the comparator and the affect of the timer disable bit will not active until Register 22 is written. The 24-bit comparator value must be loaded into the comparator register in parallel, and as a result, the load is caused by a write Register 22

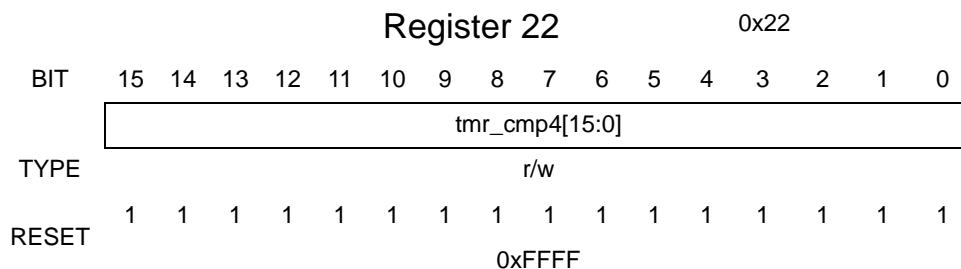
Writing of Registers 21 and 22 can be done as a recursive SPI register write transaction or two separate singular transactions. Writing to Register 22 as a separate operation will always use the present value of Register 21 to load the comparator value and set the state of the disable bit.


**Table 5-26. Register 21 Description**

Name	Description	Operation
Bits 14-8	Reserved	Leave default
Bit 15	<b>tmr_cmp4_dis</b> — This bit disables the Event Timer Comparator 4 interrupt and status bit.	1 = Disables Event Timer Compare 4 function. 0 = Enables Event Timer Compare 4 function (default).
Bits 7-0	<b>tmr_cmp4[23:16]</b> — These bits represent the 8 most significant bits of 24-bit Event Timer 4 absolute time compare value, tmr_cmp4[23:0].	Default is 0xFF.

## 5.26 Tmr\_Cmp4\_B - Register 22

The Tmr\_CMP4\_B Register 22 stores the least significant 16 bits of the 24-bit compare value. Writing to Register 22 causes an internal load of the full 24-bit comparator value (see [Section 5.25, “Tmr\\_Cmp4\\_A - Register 21”](#)) and activates the mode presently set into the tmr\_cmp4\_dis control bit.


**Table 5-27. Register 22 Description**

Name	Description	Operation
Bits 15-0	<b>tmr_cmp4[15:0]</b> — These bits represent the 16 least significant bits of 24-bit Event Timer 4 absolute time compare value, tmr_cmp4[23:0].	Default is 0xFFFF.

## 5.27 TC2\_Prime - Register 23

When the MC1321x is used in Stream Mode (Register 7, Bit 5 = 1), the 16-bit TC2\_Prime Register 23 is used in place of the 24-bit Tmr\_Cmp2 register(s) to initiate timer-triggered sequences. For an interrupt to be generated, tmr2\_mask must be set (value = 1) and the interrupt is generated via the tmr2\_irq status. Note that the tmr\_cmp2\_dis field should be disabled (value = 0) to use tc2\_prime[15:0].

		Register 23																0x23
BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		tc2_prime[15:0]																
TYPE		r/w																
RESET		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		0xFFFF																

**Table 5-28. Register 23 Description**

Name	Description	Operation
Bits 15-0	<b>tc2_prime[15:0]</b> — When use_strm_mode, Register 7, Bit 5 = 1, these bits are compared to the lower 16 bits of the Event Timer and initiate timer-triggered sequences when the value equal to current time. <b>Note:</b> tmr_trig_en, Register 6, Bit 7 must be set to 1. When use_strm_mode = 0, the TC2_Prime register is not used.	Default is 0xFFFF.

## 5.28 IRQ\_Status - Register 24

The IRQ\_Status Register 24 contains status bits that can, in turn, cause an interrupt request when enabled. Reading the register clears the status bits and releases an associated interrupt request on  $\overline{\text{IRQ}}$ . Note use of interrupt status Bit 14, Bit 13, Bit 7 and Bit 6 which are multiplexed and have special functionality.

		Register 24																0x24
BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		pll_lock_irq	ram_addr_err	arb_busy_err	strm_data_err		attn_irq	doze_irq	tmr1_irq	rx_rcvd_irq	tx_sent_irq	cca_irq	tmr3_irq	tmr4_irq	tmr2_irq	cca	crc_valid	
TYPE		r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
RESET		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		0x0000																

Table 5-29. Register 24 Description

Name	Description	Operation
Bit 11	Reserved	Leave default
Bit 15	<b>pll_lock_irq</b> — The Local Oscillator 1 Lock Detect Interrupt bit indicates whether the LO1 PLL is in or out of lock.	1 = LO1 PLL out of lock. Will cause an interrupt if <code>pll_lock_mask</code> is enabled. 0 = LO1 PLL is locked.  <p style="text-align: center;"><b>NOTE</b></p> If <code>pll_lock_irq</code> is not cleared, any subsequent active operation will be aborted.
Bit 14	<b>ram_addr_err or tx_done_irq</b> — The Packet RAM Address Error Interrupt or Stream Mode TX Done Interrupt bit is a multiplexed interrupt status bit.	When SPI bit <code>use_strm_mode</code> , Register 7, Bit 5 = 0, Register 24, Bit 14 represents 'ram_addr_err'. ram_addr_err definition: a recursive SPI read or write operation to Packet RAM exceeded maximum RAM address.  When SPI bit <code>use_strm_mode</code> , Register 7, Bit 5 = 1, Register 24, Bit 14 represents 'tx_done_irq'. tx_done_irq definition: TX Stream Mode reception complete and transceiver has returned to Idle.
Bit 13	<b>arb_busy_err or rx_done_irq</b> — The Packet RAM Arbiter Busy Error Interrupt or Steam Mode RX Done Interrupt bit is a multiplexed interrupt status bit.	When SPI bit <code>use_strm_mode</code> , Register 7, Bit 5 = 0, Register 24, Bit 13 represents 'arb_busy_err'. arb_busy_err definition: a SPI read or write operation to Packet RAM attempted during packet reception or transmission, respectively.  When SPI bit <code>use_strm_mode</code> , Register 7, Bit 5 = 1, Register 24, Bit 13 represents 'rx_done_irq'. rx_done_irq definition: RX Stream Mode reception complete and transceiver has returned to Idle Mode.
Bit 12	<b>strm_data_err</b> — The Stream Mode Data Error Interrupt bit indicates Stream Mode data read or write error	For RX Stream Mode, this bit gets set if a new RX word is updated to the SPI before the previous word is read.  For TX Stream Mode, the current TX word transmission is complete prior to next TX word being written to SPI.

**Table 5-29. Register 24 Description (continued)**

Name	Description	Operation
Bit 10	<b>attn_irq</b> — The Attention Interrupt bit indicates the $\overline{\text{ATTN}}$ pin has been asserted or Power-up complete condition from a reset condition.	The bit being set indicates the $\overline{\text{ATTN}}$ signal has been asserted low or that the MC1321x has reached a Power-up complete condition after software reset (CE released) or a hardware reset ( $\overline{\text{RST}}$ released).
Bit 9	<b>doze_irq</b> — The Doze Timer Interrupt bit.	This bit gets set when the 'tmr_cmp2[23:0]' field matches the current Event Timer value while in Doze Mode. The MC1321x returns to Idle state from Doze Mode.
Bit 8	<b>tmr1_irq</b> — The Timer Compare 1 Interrupt bit.	This bit gets set when the 'tmr_cmp1[23:0]' field matches the current Event Timer value.
Bit 7	<b>rx_rcvd_irq</b> or <b>rx_strm_irq</b> — The RX Packet Received Interrupt or RX Stream Data Ready Interrupt bit is a multiplexed interrupt status bit	<p>When SPI bit use_strm_mode, Register 7, Bit 5 = 0, Register 24, Bit 7 represents 'rx_rcvd_irq'. rx_rcvd_irq definition: RX Packet Mode reception complete and transceiver has returned to Idle Mode.</p> <p>When SPI bit use_strm_mode, Register 7, Bit 5 = 1, Register 24, Bit 7 represents 'rx_strm_irq'. rx_strm_irq definition:</p> <ol style="list-style-type: none"> <li>1) First occurrence - RX Packet Length is available to be read.</li> <li>2) Subsequent occurrences - next receive Stream data word is ready to be read.</li> </ol>
Bit 6	<b>tx_sent_irq</b> or <b>tx_strm_irq</b> — The TX Packet Sent Interrupt or TX Stream Data Needed Interrupt bit is a multiplexed interrupt status bit.	<p>When SPI bit use_strm_mode, Register 7, Bit 5 = 0, Register 24, Bit 6 represents 'tx_sent_irq'. tx_sent_irq definition: TX Packet Mode operation complete and transceiver has returned to Idle Mode.</p> <p>When SPI bit use_strm_mode, Register 7, Bit 5 = 1, Register 24, Bit 7 represents 'tx_strm_irq'. tx_strm_irq definition: Transceiver is ready for next Stream transmit data word to be written.</p>
Bit 5	<b>cca_irq</b> — The Clear Channel Assessment Ready Interrupt bit.	This bit is set when the 'Clear Channel Assessment' operation is complete.
Bit 4	<b>tmr3_irq</b> — The Timer Compare 3 Interrupt bit.	This bit gets set when the 'tmr_cmp3[23:0]' field matches the current Event Timer value.
Bit 3	<b>tmr4_irq</b> — The Timer Compare 4 Interrupt bit.	This bit gets set when the 'tmr_cmp4[23:0]' field matches the current Event Timer value.
Bit 2	<b>tmr2_irq</b> — The Timer Compare 2 Interrupt bit.	This bit gets set when the 'tmr_cmp2[23:0]' field matches the current Event Timer value, or alternately if 'tc2_prime[15:0]' field matches the current Event Timer[15:0] when enabled.

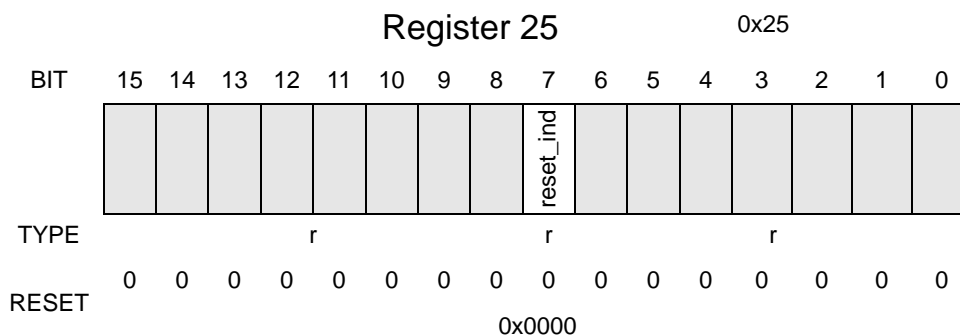


**Table 5-29. Register 24 Description (continued)**

Name	Description	Operation
Bit 1	<b>cca</b> — The Clear Channel Assessment bit indicates channel busy or channel idle.	1 = channel busy detected 0 = channel idle detected  <b>Note:</b> For <code>cca_type[1:0]</code> , Register 6, Bits 5:4 = 10, Energy Detect Mode, CCA is not calculated and <code>cca</code> is held low.
Bit 0	<b>crc_valid</b> — The RX CRC Result bit denotes if the CRC is correct or not.	1 = RX CRC correct 0 = RX CRC incorrect (default)

## 5.29 RST\_Ind - Register 25

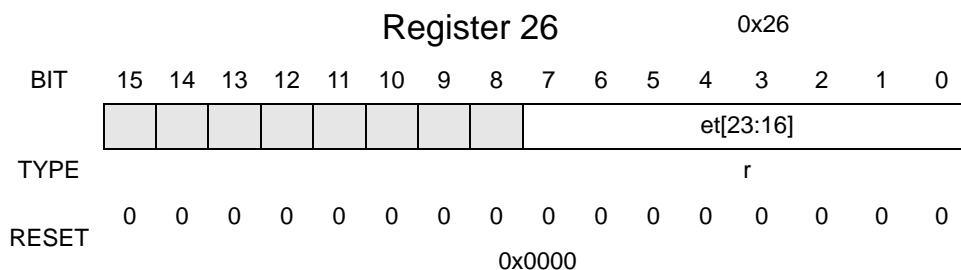
The RST\_Ind Register 25 contains the reset indicator bit. Bit `reset_ind` is cleared during a reset and gets set if Register 25 is read after a reset and remains set. This bit is useful for the MCU to determine if the transceiver has returned from a Hibernate condition via an `ATTN` signal or recovered from a reset condition.


**Table 5-30. Register 25 Description**

Name	Description	Operation
Bits 15-8, 6-0	Reserved	
Bit 7	<b>reset_ind</b> — The reset indicator bit shows whether Register 25 was read since the last $\overline{\text{RST}}$ assertion or program reset.	1 = Register 25 has been read since the last $\overline{\text{RST}}$ or Program Reset event. 0 = Register 25 has not been read since the last $\overline{\text{RST}}$ or Program Reset event. <b>Note:</b> Reading Register 25 will set Bit 7.

### 5.30 Current\_Time\_A - Register 26

The Current\_Time\_A Register 26 is read to get the 8 most significant bits of the current value of the 24-bit counter of the Event Timer.



**Table 5-31. Register 26 Description**

Name	Description	Operation
Bits 15-8	Reserved	
Bits 7 - 0	<b>et [23:16]</b> — These bits are the 8 most significant bits of the current time in the Event Timer counter.	

### 5.31 Current\_Time\_B - Register 27

The Current\_Time\_B Register 27 is read to get the 16 least significant bits of the current value of the 24-bit counter of the Event Timer.

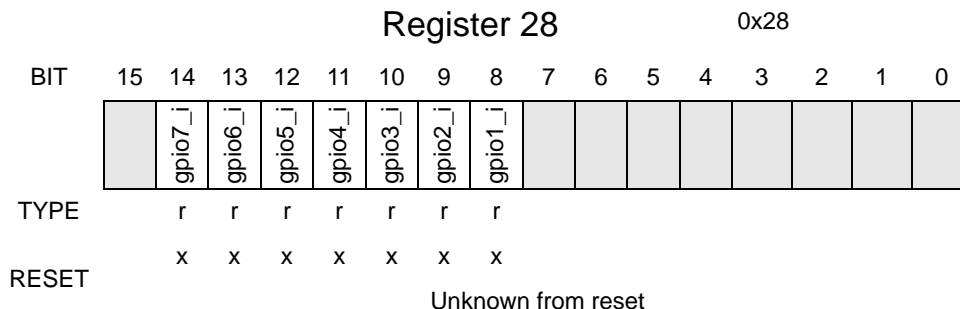


**Table 5-32. Register 27 Description**

Name	Description	Operation
Bits 15-0	<b>et[15:0]</b> — These bits represent the 16 least significant bits of the current time of the Event Timer counter.	

## 5.32 GPIO\_Data\_In - Register 28

The GPIO\_Data\_In Register 28 is read to determine the state of any GPIO that are configured as an input.



**Table 5-33. Register 28 Description**

Name	Description	Operation
Bits 15, 7-0	Reserved	
Bit 14	<b>gpio7_i</b> — This bit is the input value of GPIO7.	With gpio7_oen = 0 and gpio7_ien = 1; GPIO7 is configured as an input whose value can be read from gpio7_i.
Bit 13	<b>gpio6_i</b> — This bit is the input value of GPIO6.	With gpio6_oen = 0 and gpio6_ien = 1; GPIO6 is configured as an input whose value can be read from gpio6_i.
Bit 12	<b>gpio5_i</b> — This bit is the input value of GPIO5.	With gpio5_oen = 0 and gpio5_ien = 1; GPIO5 is configured as an input whose value can be read from gpio5_i.
Bit 11	<b>gpio4_i</b> — This bit is the input value of GPIO4.	With gpio4_oen = 0 and gpio4_ien = 1; GPIO4 is configured as an input whose value can be read from gpio4_i.
Bit 10	<b>gpio3_i</b> — This bit is the input value of GPIO3.	With gpio3_oen = 0 and gpio3_ien = 1; GPIO3 is configured as an input whose value can be read from gpio3_i.
Bit 9	<b>gpio2_i</b> — This bit is the input value of GPIO2.	With gpio2_oen = 0 and gpio2_ien = 1; GPIO2 is configured as an input whose value can be read from gpio2_i.
Bit 8	<b>gpio1_i</b> — This bit is the input value of GPIO1.	With gpio1_oen = 0 and gpio1_ien = 1; GPIO1 is configured as an input whose value can be read from gpio1_i.

### 5.33 Chip\_ID - Register 2C

The Chip\_ID Register 2C is read to get the 9-bit chip version code contained in the chip\_id[8:0] field. Valid version numbers include:

- 0x6000
- 0x6400
- 0x6800 (latest version)

**NOTE**

The latest version of the MC1321x uses transceiver Chip\_ID 0x6800. This device must have Modem SPI Registers 0x31 and 0x34 initialized (over-programmed) from default conditions for proper transceiver operation. See [Section 5.1, “Overview”](#) and the individual register descriptions.



**Table 5-34. Register 2C Description**

Name	Description	Operation
Bits 6-0	Reserved	
Bits 15- 7	<b>chip_id [8:0]</b> — These bits are the 9-bit chip version code for the transceiver.	Version dependent.

### 5.34 RX\_Status - Register 2D

The RX\_Status Register 2D has two fields, the first of which represents the average results of the CCA algorithm selected by cca\_type[1:0], Register 6, Bits 5-4. The second field gives the receive packet length parsed from the packet header; the value is latched after an RX Start Frame Delimiter is detected.

During a RX Stream Mode sequence, loading a valid FLI in rx\_pkt\_latch[6:0] causes the first rx\_strm\_irq status and interrupt in the sequence. Reading RX\_Status Register 2D for that first rx\_strm\_irq interrupt will clear the status and the interrupt. As a result, reading IRQ\_Status Register 24 is not required (see [Section 7.3.4.3, “Stream Transmit Mode”](#)).


**Table 5-35. Register 2D Description**

Name	Description	Operation
Bit 7	Reserved	
Bits 15-8	<b>cca_final [7:0]</b> — Average CCA energy.	These bits represent the average result of the CCA algorithm selected by cca_type[1:0], Register 6, Bits 5-4.
Bits 6- 0	<b>rx_pkt_latch [6:0]</b> — RX packet length	These bits give the RX packet length parsed from the packet header. The value is latched when an RX Start Frame Delimiter is detected.

### 5.35 Timestamp\_A - Register 2E

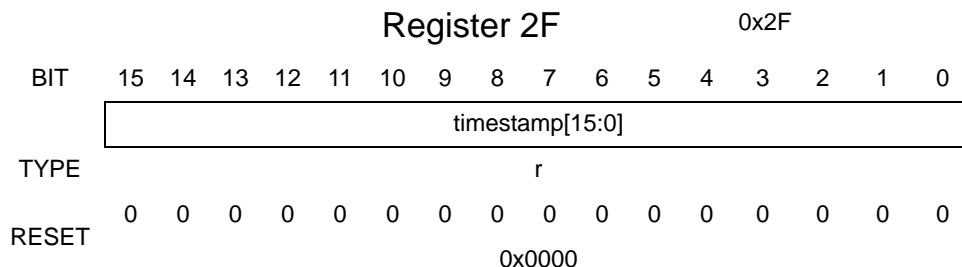
The Timestamp\_A Register 2E stores the most significant 8 bits of the value in the Event Timer counter (et[23:0]) when the beginning of the most recent receive packet occurred. The value is latched immediately following reception of the FLI field and at the beginning of the payload data.


**Table 5-36. Register 2E Description**

Name	Description	Operation
Bits 15-8	Reserved	
Bits 7 - 0	<b>timestamp [23:16]</b> — 8 most significant bits of the latched 24-bit timestamp value for the beginning of the receive packet.	

### 5.36 Timestamp\_B - Register 2F

The Timestamp\_B Register 2F stores the least significant 16 bits of the value in the Event Timer counter (et[23:0]) when the beginning of the most recent receive packet occurred. The value is latched immediately following reception of the FLI field and at the beginning of the payload data.

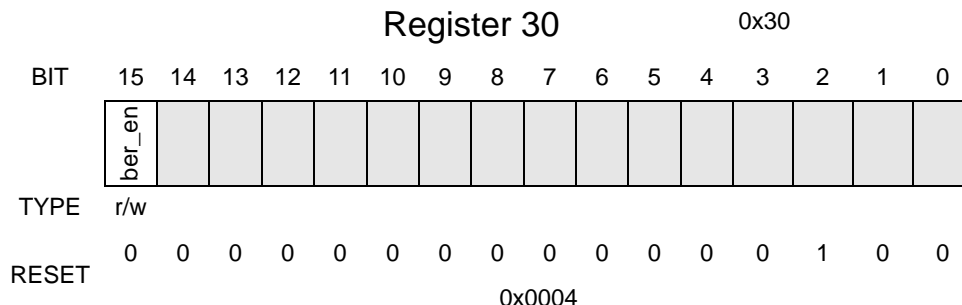


**Table 5-37. Register 2F Description**

Name	Description	Operation
Bits 15-0	<b>timestamp[15:0]</b> — 16 least significant bits of the latched 24-bit timestamp value for the beginning of the receive packet.	

### 5.37 BER\_Enable - Register 30

The BER\_Enable Register 30 contains the bit error rate test enable bit which allows the transceiver to be put into continuous Receive or Transmit Mode. This feature is useful sequences in RF test configurations (reference application note AN2976). The default condition is with the Continuous Mode disabled.



**Table 5-38. Register 30 Description**

Name	Description	Operation
Bits 14-0	Reserved	
Bit 15	<b>ber_en</b> — The bit error rate test enable bit allows the transceiver to be put into continuous Receive or Transmit Mode. Continuous TX Mode is useful for current measurements or for looking at TX spectrum. Continuous RX Mode is useful for looking at current.	1 = When a TX operation or RX operation is enabled, the transceiver stays in that operation until aborted. 0 = Continuous operation disabled; normal operation.

## 5.38 PSM\_Mode - Register 31

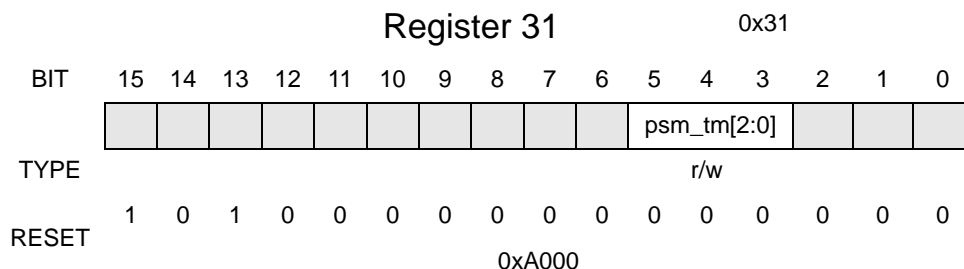
The PSM\_Mode Register 31 contains the phase shift modulator Test Mode field. The psm\_tm[2:0] 3-bit field is only used in two modes:

- When psm\_tm[2:0] = 0b000, there is normal TX modulation and normal operation.
- When psm\_tm[2:0] = 0b001, the modulator is disabled and the transmitter puts out an unmodulated signal. When the unmodulated signal is combined with continuous TX operation (see BER\_Enable Register 30), the unmodulated spectrum of the transmitter can be observed (see application note AN2976).
- Default is normal modulated operation.

### NOTE

The PSM\_Mode Register 31 must be over-written for proper transceiver operation for devices that read Chip\_ID Register 2C as 0x6800 (See Section 5.33, “Chip\_ID - Register 2C”):

- For normal operation, the over-write value is 0xA0C0 (psm\_tm[2:0] = 0b000).
- For test mode, the over-write value is 0xA0C8 (psm\_tm[2:0] = 0b001).



**Table 5-39. Register 31 Description**

Name	Description	Operation
Bits 15-6, 2-0	Reserved	
Bits 5-3	<b>psm_tm[2:0]</b> — Phase shift modulator Test Mode. Use only the two stated conditions.	psm_tm[2:0] = 0b000 - normal modulated operation. psm_tm[2:0] = 0b001 - modulator disabled.

## 5.39 Reserved - Register 34

The reserved Register 34 is listed only in that it must be over-written for latest devices.

### NOTE

The reserved Register 34 must be over-written for proper transceiver operation for devices that read Chip\_ID Register 2C as 0x6800 (See [Section 5.33, “Chip\\_ID - Register 2C”](#)):

- Over-write value is 0xFEC6

		Register 34															0x34
BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE		r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET		1	1	1	1	1	1	0	1	1	0	0	0	0	1	0	
		0xFEC2															

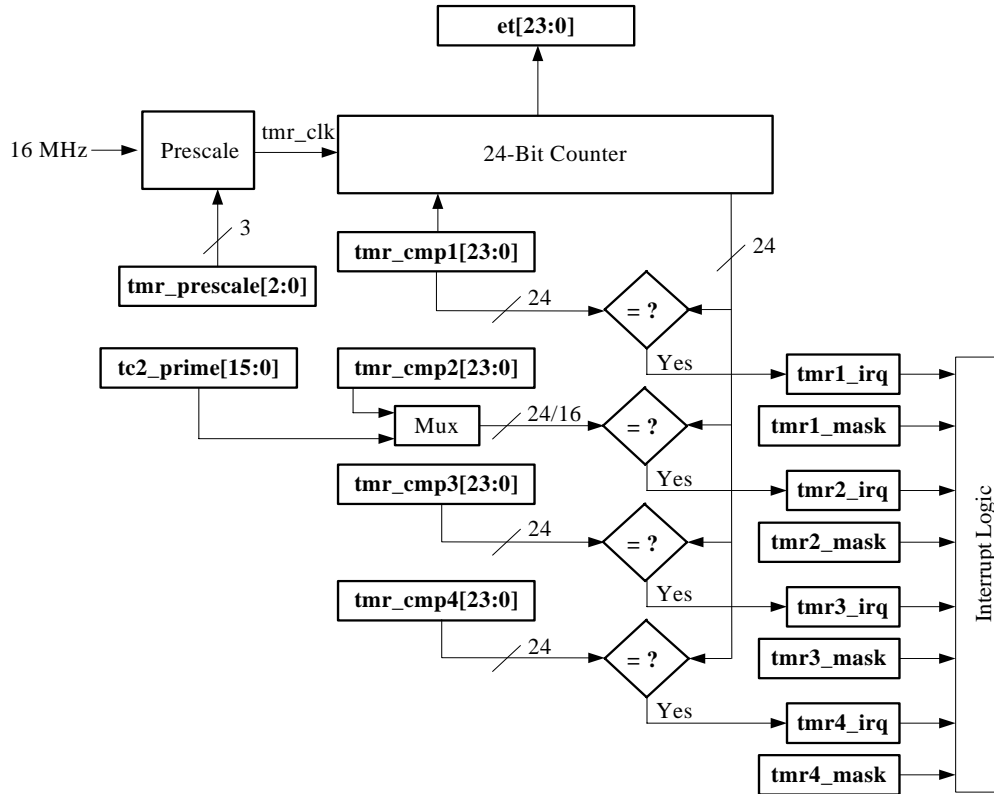
**Table 5-40. Register 31 Description**

Name	Description	Operation
Bits 15-0	Reserved	Default = 0xFEC2



# Chapter 6 Modem Timer Information

## 6.1 Event Timer Block



**Figure 6-1. Event Timer Block Diagram**

The MC1321x contains an internal Event Timer block that manages system timing. A simplified block diagram is shown in [Figure 6-1](#). The Event Timer consists of a prescaler and a 24-bit counter which increment whenever the crystal clock is operating. Interrupts to the MCU may be generated when the “current time” of the counter (et[23:0]) matches several pre-determined values set in registers via SPI write operations. The current time is accessible at any time via a SPI read operation, as well as, programmable via a SPI write operation. The Event Timer provides the following functions:

- Timer to generate current system time
- Interrupt generation at pre-determined system times
- Exit from Doze Mode at pre-determined system time
- Latches “timestamp” value during packet reception
- Initiates timer-triggered sequences

## 6.2 Event Timer Time Base

The Event Timer’s base clock (tmr\_clk) is derived from a programmable prescaler which is clocked by the 16 MHz crystal source. The prescaler provides counter input frequencies from 2 MHz down to 15.625 kHz, which sets the granularity and resolution of the current time. The prescaler, and thus the Event Timer only increment when the crystal oscillator is active. The field tmr\_prescale[2:0] Control\_C Register 9, Bits 2-0 (Section 5.2, “Register Model and Description Details”) establishes the tmr\_clk frequency as shown in Table 6-1.

**Table 6-1. Event Timer Prescaler Settings**

Register 9, Bits 2-0 tmr_prescale [2:0]	Event Timer Time Base	Maximum Event Timer Duration
000	2 MHz	8.389 seconds
001	1 MHz	16.777 seconds
010	500 kHz	33.554 seconds
011 (default)	250 kHz	67.109 seconds
100	125 kHz	134.218 seconds
101	62.5 kHz	268.436 seconds
110	31.25 kHz	536.871 seconds
111	15.625 kHz	1073.742 seconds

The 24-bit counter automatically rolls over upon reaching its maximum value, and the corresponding maximum possible Event Timer durations are also provided in Table 6-1.

## 6.3 Setting Current Time

“Current Time” is defined as the value of the Event Timer internal counter. The current time is programmable, but does not have to be programmed. In the reset condition, the MC1321x current time is set to zero. Current time advances from zero at the tmr\_clk clock rate and rolls over to zero after reaching its maximum value.

Programming “current time” is accomplished by using three SPI registers:

1. Tmr\_Cmp1\_A Register 1B, Bits 7-0, tmr\_cmp1[23:16]
2. Tmr\_Cmp1\_B Register 1C, Bits 15-0, tmr\_cmp1[15:0]
3. Control\_B Register 07, Bit 15, tmr\_load

When field tmr\_load is programmed to high, the value of “current time” is set to the value in tmr\_cmp1[23:0]. Thus, tmr\_cmp1[23:0] is first programmed to the desired current time value, then tmr\_load is programmed to 1, which initiates the timer load. The change to the “current time” value occurs within two crystal clock cycles, after which normal incrementing resumes on the next rising tmr\_clk edge. So, tmr\_load is not required to be programmed to zero for the Event Timer to resume normal operation. However, loading the Event Timer is a positive edge-triggered event, so tmr\_load must be programmed low prior to the next attempt to load the Event Timer.

## 6.4 Reading Current Time

The current value of the Event Timer can be read via the SPI using `et[23:16]`, `Current_Time_A` Register 26, Bits 7-0, and `et[15:0]`, `Current_Time_B` Register 27, Bits 15-0. The “current time” may be obtained using two single SPI reads, or one recursive 2-word SPI read (or as part of a longer recursive read operation as well). It is important to realize that the Event Timer may increment during these recursive SPI read operations, or between successive SPI reads if single SPI reads are used. During such an access, the MC1321x latches the “current time” to protect the host from obtaining an incorrect value. The “current time” least significant 16 bits (LSB) are latched when the most significant 8 bits (MSB) SPI location is read. The LSB is unlatched after the “current time” LSB location is read. This guarantees a stable value until the host completes a read of both words constituting the “current time” before it is allowed to update.

The preferred procedure to obtain the “current time” value from the MC1321x is to perform a 2-word recursive read of the “current time” starting at the MSB address.

## 6.5 Latching the Timestamp

The MC1321x has the ability create a Timestamp or to latch a copy of the “current time” while continuing to increment its internal counter. This timestamp value latched within the Event Timer corresponds to the beginning of a receive packet where the actual payload data begins after the FLI has been received. The `timestamp[23:0]` (Register 2E, Bits 7-0 and Register 2F, bits 15-0) value is read from the MC1321x by the host. When `timestamp[23:0]` is latched, its value corresponds to the “current time” value coincident with the reception of `rx_pkt_latch[6:0]`, `RX_Pkt_Latch` Register 2D, Bits 6-0. The timestamp remains latched until another packet is received, at which point the `timestamp[23:0]` value is updated and re-latched.

## 6.6 Event Timer Comparators

The MC1321x incorporates four full 24-bit programmable fields that compare to the Event Timer’s “current time”. The intent of these compares is to enable the host to schedule events relative to the “current time”. When a match between the “current time” and any one of the four timer compare values occurs, a corresponding flag is sent to internal interrupt logic. This causes the appropriate bit in the `IRQ_Status` Register 24 to be set, and depending on the interrupt mask control bit, generate an interrupt event on the IRQ pin.

### 6.6.1 Timer Compare Fields

There are four 24-bit timer compare fields:

1. `tmr_cmp1[23:0]`, `Tmr_Cmp1_A` Register 1B, Bits 7-0, and `Tmr_Cmp1_B` Register 1C, Bits 15-0.
2. `tmr_cmp2[23:0]`, `Tmr_Cmp2_A` Register 1D, Bits 7-0, and `Tmr_Cmp2_B` Register 1E, Bits 15-0.
3. `tmr_cmp3[23:0]`, `Tmr_Cmp3_A` Register 1F, Bits 7-0, and `Tmr_Cmp3_B` Register 20, Bits 15-0.
4. `tmr_cmp4[23:0]`, `Tmr_Cmp4_A` Register 21, Bits 7-0, and `Tmr_Cmp4_B` Register 22, Bits 15-0.

And a special case 16-bit timer compare field for Stream Data Mode:

1. `tc2_prime[15:0]`, `TC2_Prime` Register 23, Bits 15-0.

## 6.6.2 Timer Disable Bits

Each timer comparator has a disable bit that enables or disables the compare function. The disable bit is written to a “1” to disable the corresponding comparator and the default condition is the timer enabled (reset to “0”):

1. `tmr_cmp1_dis`, `Tmr_Cmp1_A` Register 1B, Bit 15.
2. `tmr_cmp2_dis`, `Tmr_Cmp2_A` Register 1D, Bit 15.
3. `tmr_cmp3_dis`, `Tmr_Cmp3_A` Register 1F, Bit 15.
4. `tmr_cmp4_dis`, `Tmr_Cmp4_A` Register 21, Bit 15.

If a timer comparator is disabled using its associated bit, the corresponding status bit (`tmrx_irq`) will also be cleared if set and will negate an associated interrupt.

## 6.6.3 Timer Status Flags

When enabled, all four fields can be continuously compared to the current value of the Event Timer counter. When a match occurs, the following corresponding internal status flags assert:

1. `tmr1_irq`, `IRQ_Status` Register 24, Bit 8.
2. `tmr2_irq`, `IRQ_Status` Register 24, Bit 2.
3. `tmr3_irq`, `IRQ_Status` Register 24, Bit 4.
4. `tmr4_irq`, `IRQ_Status` Register 24, Bit 3.

The status bit remains set until a read access of the `IRQ_Status` register occurs or if the timer comparator disable bit is set to disable an active comparator.

## 6.6.4 Timer Interrupt Masks

When a comparator match occurs and the internal status flag asserts, the following interrupt masks can enable an interrupt on the IRQ pin:

1. `tmr1_mask`, `IRQ_Mask` Register 05, Bit 0.
2. `tmr2_mask`, `IRQ_Mask` Register 05, Bit 1.
3. `tmr3_mask`, `IRQ_Mask` Register 05, Bit 2.
4. `tmr4_mask`, `IRQ_Mask` Register 05, Bit 3.

If the interrupt mask is set to “1” (enabled), the timer compare status will cause an interrupt and the interrupt signal will stay active until the status bit is cleared via an `IRQ_Status` read.

## 6.6.5 Setting Compare Values

Since the primary timer compare fields are 24-bit values, they are each shared between two sequential SPI register addresses. The timer compare value can be changed using two single SPI writes, or one recursive 2-word SPI write (or as part of a longer recursive write operation as well).

## NOTE

It is important to realize that not all bits of the timer compare value are updated simultaneously within the SPI. To prevent the Event Timer from generating a false match to a partially updated timer compare value, the compare hardware is inhibited temporarily. The inhibit feature initiates when the address of the MSB location of the timer compare field is decoded on a SPI write, and ends when a write to the LSB field is completed. Thus, once an SPI write to the MSB location starts, the comparator is disabled until an SPI write to the LSB location is completed. The preferred procedure for software to change a timer compare value within the MC1321x is to perform a 2-word recursive write of the timer compare field starting at the MSB address.

## 6.7 Intended Event Timer Usage

It is intended that the system utilize the “current time” value and the timer compare functions of the Event Timer to schedule system events, including:

- Generating time-based interrupts
- Exiting Doze Mode
- Triggering transceiver operations

## NOTE

- The timer\_compare functions exit reset with the timer function enabled but with the interrupts masked off. Users should disable all timers and clear the IRQ\_Status Register via a read as part of system initialization after reset.
- **Due to a device anomaly, the tmr\_cmp3[23:0] is always running and doing a comparison with the “current time”.** This only causes a problem if the MC1321x is in receive mode and has not yet started to receive a packet. In this situation, the RX will be aborted without any status being set. This may be dealt with in several ways:
  - Never let the timer counter reach the compare value in Comp 3 register.
  - Enable tmr\_cmp3[23:0] always (normal mode) and enable the interrupt. If the Tmr\_cmp3 interrupt occurs and the RX state was enabled; restart the RX mode as required.
  - The “out-of-idle” status output can be used to detect the condition where the device is expected to be in RX mode, and the RX has been aborted without the status being set.

### 6.7.1 Generating Time-Based Interrupts

Generating time-based interrupts is accomplished by setting timer compare values relative to the “current time”, allowing the Event Timer counter to increment until a timer compare match is generated, and using this match to generate an interrupt to the host. The general procedure is as follows:

1. Disable the timer compare. This clears the status flag if already set.

2. Enable the timer compare interrupt mask.
3. Read the “current time” value from et[23:0].
4. Add an offset to this value to equal desired “future time”.
5. Program the appropriate timer\_compare value to “future time”.
6. Program the appropriate tmr\_cmpx\_dis bit to enable the compare.
7. Allow a timer compare match to set the status register bit and generate an interrupt. The appropriate internal status register bit is always set upon a timer\_compare match. An external interrupt is generated when the corresponding SPI interrupt mask bit, Register 5, Bits 3, 2, 1, or 0, is set.
8. Program the appropriate tmr\_cmpx\_dis bit to disable the compare function. If this is not done, the compare function will continue to run and generate another interrupt every time the counter rolls over and again matches the comparator.

### 6.7.2 Using tmr\_cmp2[23:0] to Exit Doze Mode

The Event Timer provides a timer-based mechanism to bring the MC1321x out of Doze Mode. The MC1321x is put into Doze Mode when doze\_en, Control\_B Register 07, Bit 0, is programmed high. While in Doze Mode, a match between “current time” and field tmr\_cmp2[23:0] causes the MC1321x to exit Doze Mode and return to Idle Mode.

The general procedure is as follows:

1. Read the “current time” value from et[23:0].
2. Add an offset to this value to equal desired “future time” to exit Doze Mode.
3. Program field tmr\_cmp2[23:0] to value “future time”.
4. Program doze\_mask, Register 05, Bit 4, to 1.
5. Program doze\_en, Register 7, Bit 0, to 1. The MC1321x then enters Doze Mode. (Note that the control bit tmr\_cmp2\_dis has no effect on this mode).
6. When “current time” equals tmr\_cmp2[23:0], the MC1321x exits Doze Mode, and doze\_irq, Register 24, Bit 9, gets set. An external interrupt is also generated because doze\_mask is set.

#### NOTE

The MC1321x can always be taken out of Doze Mode by asserting  $\overline{ATTN}$  or  $\overline{RST}$ . Also, if acom\_en IRQ\_Mask Register 05, Bit 8 is set before entering Doze Mode, the Event Timer logic is disabled for additional power savings and only  $\overline{ATTN}$  or  $\overline{RST}$  will cause exit of Doze Mode.

### 6.7.3 Timer-Triggered Transceiver Events

An Event Timer can be used to initiate the MC1321x transceiver operations such as transmit and receive. The desired operation can be scheduled to commence at a future time greater than the “current time” by using the MC1321x timer-triggered operation capability. Timer-triggered operations are invoked by using either by tmr\_cmp2 [23:0] for Packet Mode operations or tc2\_prime[15:0] for Stream Mode operations. A time greater than the “current time” is programmed into the appropriate compare field and tmr\_trig\_en, Control\_A Register 6, Bit 7 is programmed high. When the “current time” advances to match the value set

in the compare field, the selected operation sequence will commence automatically without intervention from the host. This allows the host to arm the MC1321x to execute a desired operation at a future time, and go off to perform other necessary system functions.

### 6.7.3.1 Packet Mode Timer\_Triggered TX or RX Events

In Packet Mode only `tmr_cmp2` [23:0] can be used to initiate a transceiver event. The general procedure is as follows:

1. Desired frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. Stream Mode `rx_strm`, `tx_strm`, and `use_strm_mode` control bits must be cleared.
4. Read the “current time” value from `et`[23:0].
5. Add an offset to this value to equal desired “future time” to initiate selected operation.
6. Program field `tmr_cmp2`[23:0] to value “future time”.
7. Program `tmr_cmp2_dis` to 0 to enable the compare function.
8. If desired, program `tmr2_mask`, `IRQ_Mask` Register 05, Bit 1 high to enable an interrupt when the timer compare function completes and starts the transceiver.
9. For a TX operation only, load `tx_pkt_length`[6:0] and payload data into `tx_pkt_RAM`[15:0].
10. Program `tmr_trig_en`, `Control_A` Register 6, Bit 7 high to enable a timer-based operation.
11. Program the MC1321x/193 for the desired transceiver operation via `xcvr_seq`[1:0].
12. Assert the RXTXEN pin and hold high.
13. When “current time” equals `tmr_cmp2`[23:0], the MC1321x initiates the selected transceiver operation. When `tmr2_irq`, `IRQ_Status` Register 24, Bit 2 is set to 1, an external interrupt is generated if the interrupt mask bit (`tmr2_mask`) was set high.

#### NOTE

`tmr_trig_en` is level sensitive. It is not necessary to program it to 0 prior to the next timer triggered operation.

14. Once started, the transceiver operation commences in a normal manner.

### 6.7.3.2 Stream Mode Timer\_Triggered TX or RX Events

In Stream Mode, `tc2_prime`[15:0] is used in place of `tmr_cmp2` [23:0] to initiate a transceiver event. The `tmr_cmp2_dis` bit must be enabled as allows `tc2_prime`[15:0] to function. The general procedure is as follows:

1. Desired frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. Packet Mode `xcvr_seq`[1:0] field should be cleared.
4. Read the “current time” value from `et`[23:0].
5. Add an offset to this value to equal desired “future time” to initiate selected operation.  
Note that only the lowest 16 bits are used with `tc2_prime`[15:0].

## Modem Timer Information

6. Program field `tc2_prime[15:0]` to value “future time”.
7. If desired, program `tmr2_mask`, `IRQ_Mask Register 05`, Bit 1 high to enable an interrupt when the timer compare function completes and starts the transceiver.
8. For a TX operation only, load `tx_pkt_length[6:0]` and first word of payload data into `tx_pkt_RAM[15:0]`.
9. Enable `tmr_cmp2_dis` by programming to low.
10. Program `tmr_trig_en`, `Control_A Register 6`, Bit 7 high to enable a timer-based operation.
11. Program `use_strm_mode` bit high for Stream Data Mode.
12. Program either `rx_strm` or `tx_strm` for desired operation.
13. Assert the `RXTXEN` pin and hold high.
14. When “current time” equals `tc2_prime[15:0]`, the MC1321x initiates the selected transceiver operation. When `tmr2_irq`, `IRQ_Status Register 24`, Bit 2 is set to 1, an external interrupt is generated if the interrupt mask bit (`tmr2_mask`) was set high.

### NOTE

`tmr_trig_en` is level sensitive. It is not necessary to program it to 0 prior to the next timer triggered operation.

15. Once started, the transceiver operation commences in a normal manner.



# Chapter 7

## Modem Modes of Operation

### 7.1 Modem Operational Modes Summary

The MC1321x has a number of passive operational modes that allow for low-current operation as well as modes where the transceiver is active. These modes are summarized, along with the transition times, in [Table 7-1](#). [Figure 7-1](#), [Figure 7-2](#), [Figure 7-3](#) and [Figure 7-4](#) show the state diagrams for different operational modes.

**Table 7-1. Modem Mode Definitions and Transition Times**

Mode	Definition	Transition Time
Off	$\overline{RST}$ asserted. All IC functions Off, Leakage only. Digital outputs are tri-stated including IRQ. Any RAM buffer data is lost.	10 - 25 ms to Idle
Hibernate	Crystal Reference Oscillator Off. SPI not functional. IC Responds to $\overline{ATTN}$ . Data is retained.	8 - 20 ms to Idle
Doze	Crystal Reference Oscillator On but CLKO is available only if Register 7, Bit 9 = 1 for frequencies of 1 MHz or less. (SPI not functional.) Responds to $\overline{ATTN}$ and can be programmed to enter Idle Mode through an internal timer comparator.	$(300+1/CLKO)$ $\mu$ s to Idle
Idle	Crystal Reference Oscillator On with CLKO output available. SPI active.	
Receive	Crystal Reference Oscillator On. Receiver On.	144 $\mu$ s from Idle
Transmit	Crystal Reference Oscillator On. Transmitter On.	144 $\mu$ s from Idle
CCA/Energy Detect	Crystal Reference Oscillator On. Receiver On.	144 $\mu$ s from Idle

Idle Mode is normally the state from which other states are derived. Low power states include the Off, Hibernate, and Doze modes. The Off state is the lowest power and is caused by the hardware reset. Transition from the Off to Idle Mode occurs when  $\overline{RST}$  is negated to high. Once in Idle Mode, the SPI is active and used to control the MC1321x. Transition to Hibernate or Doze modes is enabled via the SPI.

There are active states of Idle, Transmit (TX), Receive (RX), and Clear Channel Assessment (CCA) modes. The transition from the Idle state to the TX or RX is first determined by the desired data mode of the user, i.e., Packet Mode or Stream Mode. If Packet Data Mode is used, writing to the `xcvr_seq` field can select RX or TX in Packet Mode. Packet RX and TX can also be modified as timer-based sequences using `Tmr_Cmp2`. [Figure 7-1](#) and [Figure 7-2](#) show state diagrams for Packet Mode transceiver operations without and with timer-initiated sequences.

If Stream Data Mode is used, then the configuration bits of `tx_strm`, `rx_strm`, and `use_strm_mode` control transition to the TX and RX sequences. `Use_strm_mode` is a top level control bit and enables a sequence as selected by either `tx_strm` or `rx_strm` (only one bit should be set to 1 at a time). Also, when using Stream

Mode, the `xcvr_seq` field should be set to 0x0 so the selected TX or RX sequence starts from the Idle condition. As with the Packet Mode, the Stream Mode can be modified to have timer-initiated sequences, only for this case TC2\_Prime is used as the timer comparator. Figure 7-3 and Figure 7-4 show state diagrams for Stream Mode transceiver operations without and with timer-initiated sequences.

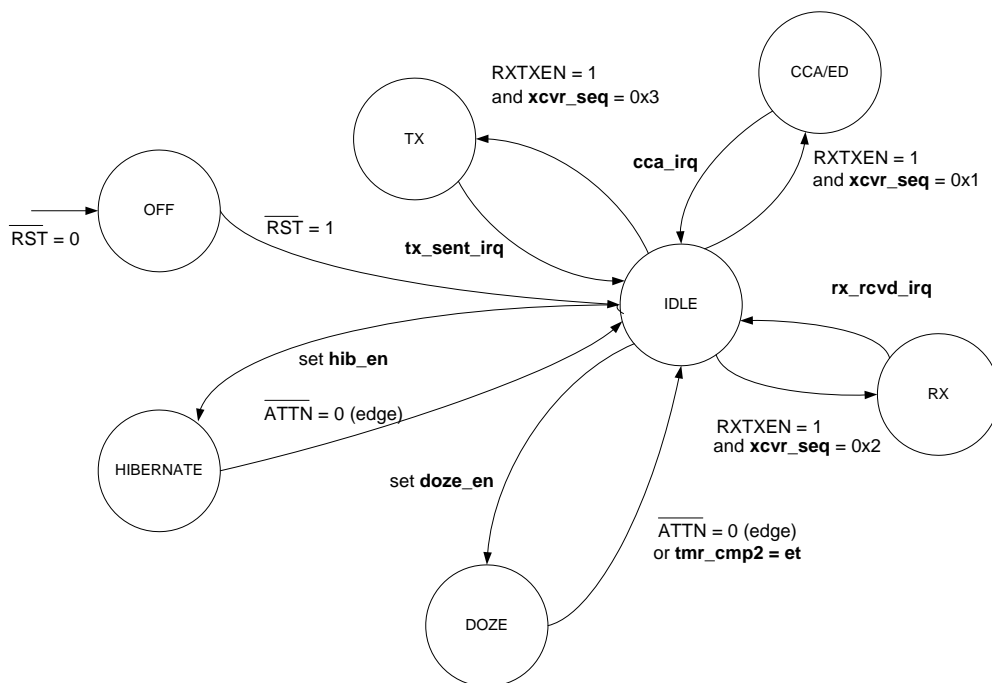


Figure 7-1. State Diagram for Packet Mode Without Timer Enabled States

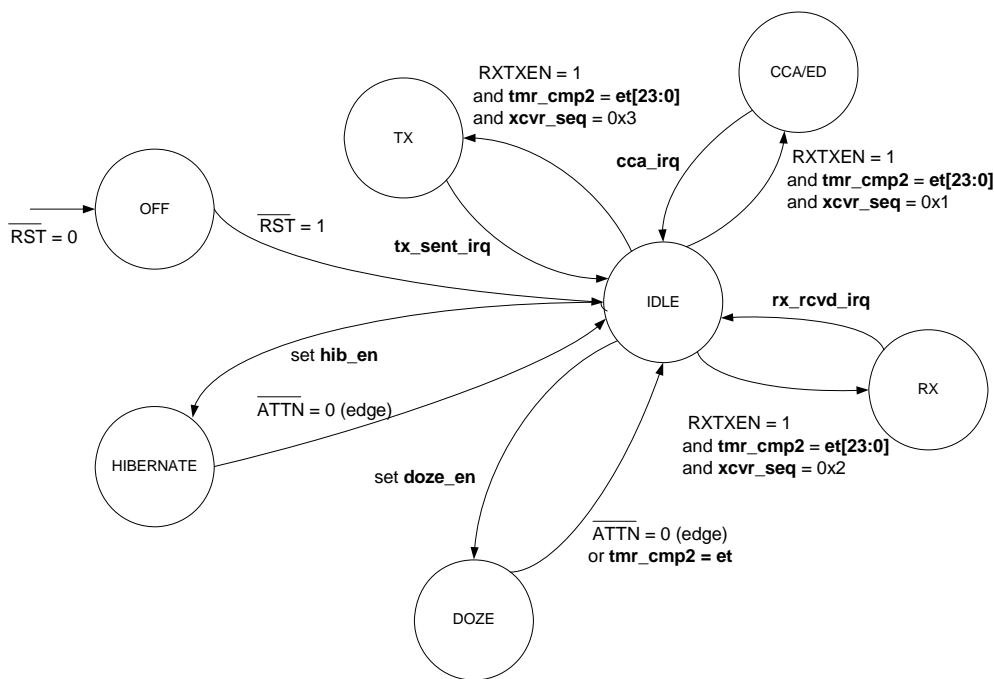


Figure 7-2. State Diagram for Packet Mode With Tmr\_Cmp2 Enabled States (`tmr_trig_en = 1`)

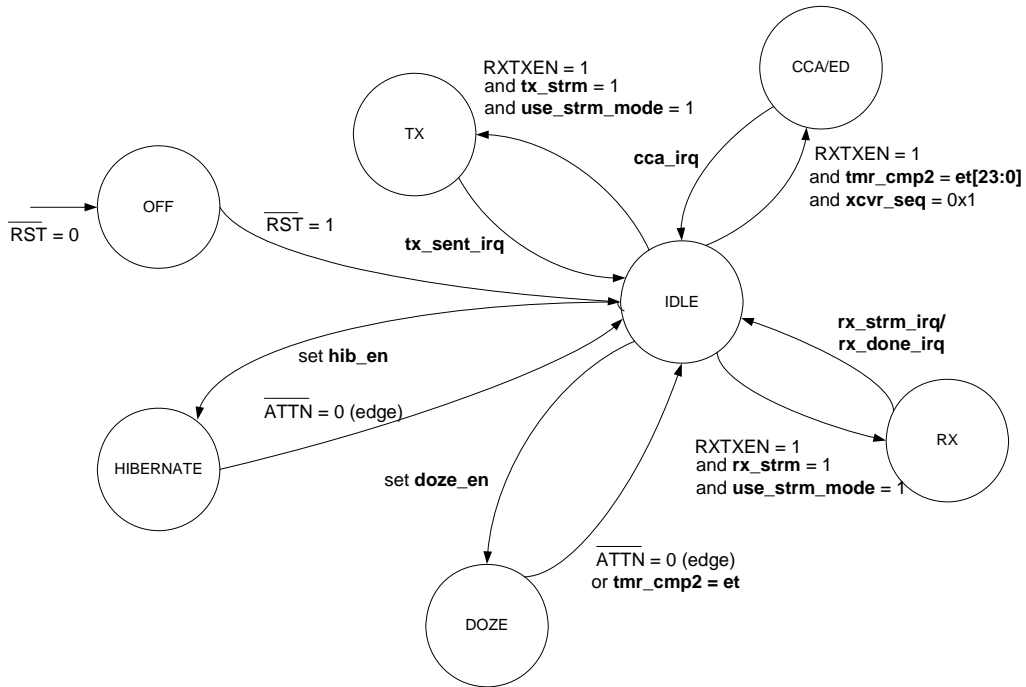


Figure 7-3. State Diagram for Stream Mode Without Timer Enabled States

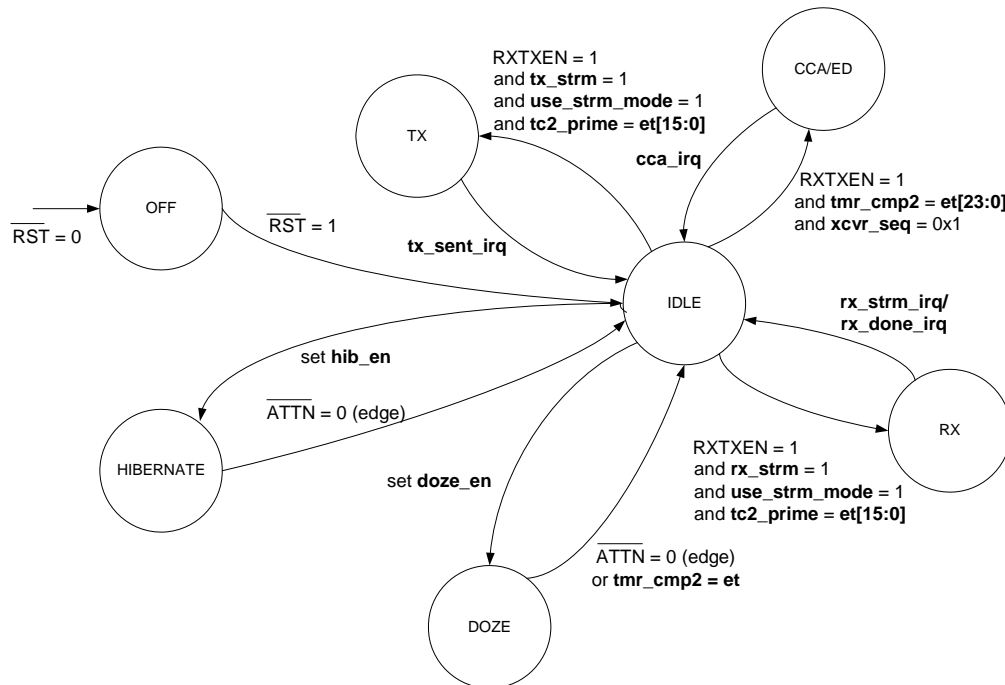


Figure 7-4. State Diagram for Stream Mode With TC2\_Prime Enabled State ( $\text{tmr\_trig\_en} = 1$ )

## 7.2 Low Power Modes

The MC1321x supports several low-power modes where the transceiver circuitry is not active. Each mode has a different advantage, these modes are described in the following sections.

### 7.2.1 Off Mode

The Off or Reset condition has the absolutely lowest power, and is controlled by the  $\overline{\text{RST}}$  input. As long as  $\overline{\text{RST}}$  is asserted low the MC1321x remains in the Off Mode. All functions are disabled and no RAM data is retained. Current draw is attributed to leakage only.

To exit Off Mode,  $\overline{\text{RST}}$  is negated high. The MC1321x then moves to Idle Mode within 25 milliseconds.

### 7.2.2 Hibernate Mode

Although the Off or Reset condition has the lowest possible power, the Hibernate Mode has the next lowest power. All hardware blocks are deactivated (including the SPI interface) and no timers are running. Internal voltage regulation is dropped to less than 1 Vdc. Hibernate Mode has the advantage of retention of all RAM data (which does not occur in the Off Mode) and of the SPI configuration prior to entering Hibernate Mode.

Hibernate Mode is entered from Idle by programming `hib_en`, Control\_B Register 07, Bit 1, to “1”. Hibernate is then entered 128 CLKO cycles after `hib_en` is set. The normal way to exit from Hibernate Mode is to assert  $\overline{\text{ATTN}}$  which will cause the MC1321x to go to Idle Mode. The MC1321x then moves to Idle Mode within 20 milliseconds. Asserting  $\overline{\text{RST}}$  which will force the Off condition.

On entering Hibernate Mode, 128 clock cycles are available at CLKO before the clock is disabled. These 128 CLKO cycles allow a host that uses CLKO as a source clock to attain a low power state prior to losing clock. After the 128 CLKO cycles, the transceiver transitions to the low power state.

Upon exiting Hibernate, CLKO will restart (if enabled) with the same frequency as programmed before before entering Hibernate.

### 7.2.3 Doze Mode

Doze Mode has variations of normal Doze Mode and a subset called Acoma state.

#### 7.2.3.1 Normal Doze Mode

Doze Mode is an additional low power state specifically designed to work in concert with the Event Timer. Most internal hardware blocks are de-activated (including the SPI interface) and internal regulation is reduced, but the reference oscillator and Event Timer are active. Internal RAM data and SPI configuration are retained similar to Hibernate Mode.

In Doze Mode, CLKO can optionally be made available by setting `clko_doze_en`, Control\_B Register 07, Bit 9, with the disadvantage of increased power consumption. The CLKO frequency must be set for 1 MHz or lower. If `clko_doze_en` = 0, then CLKO is disabled 128 clock cycles after entering Doze Mode.

Doze Mode is entered from Idle by programming `doze_en`, Control\_B Register 07, Bit 0, to “1”. Doze Mode is then entered 128 CLKO cycles after `doze_en` is set. After the 128 CLKO cycles, the transceiver transitions to the low power state

The intended primary way to exit Doze Mode is through a “wake-up” timer and return to Idle at a pre-determined time. This will occur when the Event Timer equals the value in `tmr_cmr2[23:0]`, `Tmr_Cmp2`, Registers 1D and 1E. When the match occurs, The MC1321x exits Doze, sets status bit `doze_irq`, IRQ\_Status Register 24, Bit 9, and returns to Idle. An interrupt request will be generated if the `doze_mask` IRQ\_Mask Register 05, Bit 4, has been set.

If CLKO was enabled before Doze Mode and disabled during Doze Mode, the CLKO will automatically re-start after exiting Doze with the exception of two frequencies. The two lowest frequencies of 16.393+ kHz and 32.786+ kHz will not restart directly when exiting Doze Mode. To restart CLKO for these frequencies, the `clk_en`, Control\_C Register 09, Bit 2, must be cleared and set again.

Doze Mode can be exited at any time similar to Hibernate by asserting  $\overline{ATTN}$  or  $\overline{RST}$ . If Doze is exited by asserting  $\overline{ATTN}$  and the Event Timer was activated and waiting on a timeout to waken, the timer should be disabled or the timeout will still happen and generate a status and possible interrupt.

### 7.2.3.2 Acoma Doze Mode

A subset of Doze Mode without timer wake-up is the Acoma state that has the advantage of lowest power with data retention while allowing CLKO to run. This mode disables the Event Timer and prescaler, but allows the clock to run and have CLKO available. Timers are not available so only  $\overline{ATTN}$  will return the device to Idle or a  $\overline{RST}$  can be used to exit. Acoma Mode is entered by setting `acoma_en`, IRQ\_Mask Register 05, Bit 8 = 1.

## 7.3 Active Modes

There are four active modes for the MC1321x which include, Idle, Transmit (TX), Receive (RX) and Clear Channel Assessment (CCA)/Energy Detect (ED).

### 7.3.1 Idle Mode

Idle Mode is the default mode after leaving one of the low-power modes and is the basic active state from which all other activity is initiated. In Idle Mode, the receiver hardware and transmitter hardware are shut down waiting for a command. The command can instruct the MC1321x to transition to Receive Mode, Transmit Mode, CCA Mode, or to one of the low-power modes. The CCA / ED Mode (in its variations) is called by writing to the `xcvr_seq[1:0]` field, Control\_A Register 06, Bits 1 - 0.

The transitions to RX Mode or TX Mode, however, are dependent upon the desired data mode. For example, Packet Mode or Stream Mode. In Packet Data Mode, the transition to RX Mode or TX Mode is determined by writing the appropriate value to the `xcvr_seq[1:0]` field. Alternately, Stream Data Mode uses control bits `use_strm_mode`, `rx_strm`, and `tx_strm` to cause transition to an RX or TX Mode (in this case, the `xcvr_seq[1:0]` field should not be written).

Once CCA, Receive, or Transmit is entered, the MC1321x will transition back to the Idle Mode upon completion of the selected operation. For Packet Mode, at the end of the operation the `xcvr_seq[1:0]` field

will not be cleared to an Idle value although the transceiver returns to the Idle condition. In this case, a read from the `xcvr_seq[1:0]` field will return the code of the last programmed operation.

In Idle Mode, the crystal oscillator is active, CLKO is available (if enabled), and the SPI is active.

### 7.3.2 Controlling Transition to Other Active Modes from Idle

Reviewing the state diagrams in [Figure 7-1](#), [Figure 7-2](#), [Figure 7-3](#), and [Figure 7-4](#), shows that the input signal RXTXEN must be asserted to allow transition from Idle to other active states. The recommended procedure is that RXTXEN is taken low while setting-up the desired function (writing required registers) and then after SPI transactions, the MCU raises RXTXEN to a high state enabling the transition. For timed functions (using either `tmr_cmp2` or `tc2_prime`), the same procedure holds with the exception that the transition will be delayed until the timer function completes.

### 7.3.3 Packet Mode Data Transfer TX and RX Operation

Packet Mode assumes use of the onboard buffer RAMs and has the advantage of using less of the MCU resources. The Idle Mode is the condition from which RX and TX modes are initiated. Writing to the `xcvr_seq[1:0]` field arms the transition to the desired mode (`use_strm_mode = 0`, `tx_strm = 0`, and `rx_strm = 0`). However, the RXTXEN signal must also be high for the transition to occur and if the Event Timer is enabled, the transition will be synchronized to the timer compare event. Once Receive or Transmit is entered, the MC1321x will transition back to the Idle Mode upon completion of the selected operation.

[Table 7-2](#) shows the transceiver sequence field modes.

**Table 7-2. Transceiver Sequence Field (`xcvr_seq[1:0]`)**

Mode	Value	Description
Idle	00	Idle Mode - default state after exiting low-power modes
CCA / Energy Detect	01	CCA / Energy detect - special case of receive used to monitor channel energy
Packet Receive	10	Packet Receive
Packet Transmit	11	Packet Transmit

The selected mode is controlled by:

1. `xcvr_seq[1:0]` field - Shown in [Table 7-2](#).
2. RXTXEN signal - The transition to any other active mode from Idle will not occur unless RXTXEN is asserted high.
3. `tmr_trig_en`, Control\_A Register 06, Bit 7 - When `tmr_trig_en` is set to “1”, the transition to the selected active mode will be based on a `tmr_cmp2[23:0]` compare function as described in the Event Timer section. When `tmr_trig_en` is cleared to “0”, the transition to the selected active mode is based only on programming of `xcvr_seq[1:0]`. For both cases, RXTXEN must be high and overrides.
4. For Packet Mode, `tx_strm`, `rx_strm` and `use_strm_mode` control bits must always be cleared to zero.

### 7.3.3.1 Packet Receive Mode

Receive Mode is the state where the transceiver is waiting for an incoming data frame. The advantage of Packet Receive Mode is that it allows the MC1321x to receive the whole packet without intervention from the microcontroller. The entire packet payload is stored in RX Packet RAM and the microcontroller fetches the data after determining the length and validity of the RX packet. The disadvantage of Packet Mode is that there is a significant latency to reading the RX data via a SPI recursive access, processing the frame data and providing a response if required; the Streaming Mode is better suited to faster response time.

The MC1321x waits for preamble followed by a Start of Frame Delimiter. From there, the Frame Length Indicator is used to determine length of the frame and calculate CRC. The receive function provides the following frame information/data:

1. The frame payload data - accessed through rx\_pkt\_ram[15:0] RX\_Pkt\_RAM Register 01.
2. CRC valid status - reported by crc\_valid, IRQ\_Status Register 24, Bit 0.
3. Payload data length - reported by rx\_pkt\_latch[6:0], RX\_Pkt\_Latch Register 2D, Bits 6 - 0.
4. Link quality indicator (LQI) - this is a measure of the received energy that occurs during the received frame. Once a preamble is detected, the received energy is measured over a 64  $\mu$ s period and stored in cca\_final[7:0], RX\_Pkt\_Latch Register 2D, Bits 15 - 8.

#### NOTE

After a frame is received, the application must determine the validity of the packet. Due to noise, it is possible for an invalid packet to be reported with either of the following conditions:

- a.) A valid CRC and a frame length of 0,1, or 2.
- b.) Invalid CRC and invalid frame length.

The application software needs to verify that:

- a.) The CRC is valid.
- b.) The frame length is valid with a value of 3 or greater.

The following is a typical sequence for packet receive operation (not using a timer-based start):

#### NOTE

This sequence shows use of the TTXEN signal to control a sequence. The RXTXEN signal can be tied high and left high. The sequence will then start based on writing of the field xcvr\_seq[1:0].

1. The RX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. Control bits cleared (no Stream Mode and no timer):
  - a) tmr\_trig\_en = 0.
  - b) tx\_strm = 0.
  - c) rx\_strm = 0.
  - d) use\_strm\_mode = 0.

4. rx\_rcvd\_mask, Control\_A Register 06, Bit 8 is programmed to “1” to enable an interrupt request when the RX packet has been received.
5. Transceiver sequence is programmed to xcvr\_seq[1:0] = 0x2 for receive.
6. RXTXEN must be asserted and held high.
7. When a packet is successfully received, the following are reported:
  - a) rx\_pkt\_latch[6:0], RX\_Pkt\_Latch Register 2D, Bits 6 - 0 - reports the length of the packet payload including 2 bytes of CRC data.
  - b) crc\_valid, IRQ\_Status Register 24, Bit 0 - reports the results of the CRC check, where a “1” indicates valid CRC.
  - c) cca\_final[7:0], RX\_Pkt\_Latch Register 2D, Bits 15 - 8 - reports Link Quality Indicator.
  - d) rx\_rcvd\_irq, IRQ\_Status Register 24, Bit 7- reports the completion of packet reception, where a “1” indicate complete status. Also, an interrupt is generated due to the valid status.
8. In response of the interrupt request from the MC1321x, the microcontroller does the following:
  - a) Determines the validity of the frame by reading and checking rx\_rcvd\_irq and crc\_valid. Determines a valid length for the frame by reading rx\_pkt\_latch[6:0].
  - b) Reads the payload data from RX Packet RAM using a recursive read from rx\_pkt\_ram[15:0] RX\_Pkt\_RAM Register 01.

### 7.3.3.2 Aborting a Packet Receive Sequence

It may be required to abort a packet receive sequence. The RX sequence can be aborted by either negating RXTXEN to low or by writing xcvr\_seq[1:0] to 0x0. If either of these conditions happen, the transceiver returns to Idle Mode and no additional status bit is set.

### 7.3.3.3 Packet Transmit Mode

The advantage of Packet Transmit Mode is that it allows the MC1321x to send the whole packet without intervention from the microcontroller. The entire packet payload is pre-loaded in TX Packet RAM, the MC1321x sends the frame, and then the transmit complete status is given to the MCU.

#### **NOTE**

This sequence shows use of the TTXEN signal to control a sequence. The RXTXEN signal can be tied high and left high. The sequence will then start based on writing of the field xcvr\_seq[1:0].

The following is a typical sequence for packet transmit operation (not using a timer-based start):

1. The TX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. Control bits cleared (no Stream Mode and no timer):
  - a) tmr\_trig\_en = 0.
  - b) tx\_strm = 0.
  - c) rx\_strm = 0.
  - d) use\_strm\_mode = 0.



4. tx\_sent\_mask, Control\_A Register 06, Bit 9 is programmed to “1” to enable an interrupt request when the TX packet has been sent.
5. The MCU loads the value of the number of data bytes plus two (for FCS) into tx\_pkt\_length[6:0] TX\_Pkt\_Ctl Register 03, Bits 6 - 0.
6. The MCU then pre-loads the number of actual data bytes into tx\_pkt\_ram[15:0] TX\_Pkt\_RAM register 02 with a recursive SPI write. An odd number of data bytes requires stuffing a dummy byte due to the 16-bit SPI data format.
7. Transceiver sequence is programmed to xcvr\_seq[1:0] = 0x3 for transmit.
8. RXTXEN must be asserted and held high.
9. When the packet is successfully transmitted, tx\_sent\_irq reports the completion of packet transmission, where a “1” indicates a complete status. Also, an interrupt is generated due to the valid status.
10. In response of the interrupt request from the MC1321x, the microcontroller reads the status to clear the interrupt and check successful transmission.

### 7.3.4 Stream Mode Data Transfer TX and RX Operation

Stream Mode does not use the onboard buffer RAMs and presents the data to the SPI buffer on a word-by-word basis. The Idle Mode is the condition from which RX and TX modes are initiated. In Stream Mode, control bits tx\_strm, rx\_strm and use\_strm\_mode control transition to RX or TX modes (xcvr\_seq[1:0] must equal 00 for Idle). However, the RXTXEN signal must also be high for the transition to occur and if the Event Timer is enabled, the transition will be synchronized to the TC2\_Prime compare event. Once Receive or Transmit is entered, the MC1321x will transition back to the Idle Mode upon completion of the selected operation.

The selected mode is controlled by:

1. For Stream Mode, xcvr\_seq[1:0] field must be cleared to zero.
2. use\_strm\_mode, Control\_B Register 07, Bit 5 - Set to “1” first to enable Stream Mode.
3. rx\_strm, Control\_A Register 06, Bit 11, and tx\_strm, Control\_A Register 06, Bit 12 - These bits are used to initiate RX or TX operation once Stream Mode is selected.
  - a) rx\_strm is set to 1 to initiate a RX sequence (tx\_strm stays at 0).
  - b) tx\_strm is set to 1 to initiate a TX sequence (rx\_strm stays at 0).
4. RXTXEN signal - The transition to any other active mode from Idle will not occur unless RXTXEN is asserted high. This signal may be held high to allow the transitions to occur based on SPI programming and/or Event Timer activity.
5. tmr\_trig\_en, Control\_A Register 06, Bit 7 - When tmr\_trig\_en is set to “1”, the transition to the selected active mode will be based on a tc2\_prime[15:0] compare function as described in the Event Timer section. For Stream Mode, tc2\_prime[15:0] takes the place of tmr\_cmp2[23:0] making the compare function based on a 16-bit value as opposed to a 24-bit value. Also, tc2\_prime[15:0] uses the tmr2\_irq when the transceiver is in Stream Mode. When tmr\_trig\_en is cleared to “0”, the transition to the selected active mode is based only on programming of the stream control bits. For both cases, RXTXEN must be set high.

The Stream Mode requires that the host MCU either supply a data word every 64 microseconds for a TX sequence or read a data word every 64 microseconds for a RX sequence (with the exception shown being in the note below). If this timing requirement is violated, a `strm_data_err` status will be issued causing an interrupt request if enabled. The use of the interrupts is important to supporting stream data mode.

For TX Mode once the sequence has been initiated, an interrupt (`tx_strm_irq` bit) is generated for every new data word required for the packet (excluding the CRC data). The MCU is required to write the data word to TX\_Pkt\_RAM Register 02 within the 64 microsecond interval. Writing the TX word will clear the interrupt request and thus saves an access to the IRQ\_Status Register 24 to clear the  $\overline{IRQ}$ . An interrupt will also be generated to signal the completion of the TX operation via the `tx_done_irq` bit.

The RX Mode is slightly more complex. Once a RX sequence has been initiated and a RX packet is being received, the first interrupt generated (`rx_strm_irq` bit) is for the MCU to read the RX Packet Length from Register 2D. Reading Register 2D will clear the interrupt and saves a read of the IRQ\_Status Register 24. There will be a following interrupt (`rx_strm_irq` bit) for each received data word and the interrupt can be cleared by reading the data from RX\_Pkt\_RAM Register 01. The data must be read within the 64 microsecond period, and no interrupt will be generated for the received CRC data. An interrupt will also be generated for the completion of the RX operation via the `rx_done_irq` bit.

**NOTE**

- There is one exception to the 64 microsecond response time. For the stream receive sequence and an odd byte length packet, the last data transfer only has 8 bits (one byte) of valid data and the data will only be available for 32 microseconds. If the packet length is even, the full 64 microseconds is available
- If the timing of the last data transfer on a stream receive is violated, a `strm_data_err` status will not be issued and the user software must take this into account

These are described in more detail in the following sections.

**7.3.4.1 Stream Receive Mode**

The advantage of Stream Receive Mode is that it allows the microcontroller to fetch data from the MC1321x as soon as the data arrives. As a result the MCU can begin processing frame information as it arrives and provide a quicker turn-around time if a response is required. The disadvantage of Stream Mode is that there is a significant amount of overhead required from the MCU to process incoming data on a word-by-word basis.

**NOTE**

Similar to Packet Receive Mode, after a frame is received, the application must determine the validity of the packet. Due to noise, it is possible for an invalid packet to be reported with either of the following conditions:

- a.) A valid CRC and a frame length of 0,1, or 2.
- b.) Invalid CRC and invalid frame length.

The application software needs to verify that:

- a.) The CRC is valid.

b.) The frame length is valid with a value of 3 or greater.

### NOTE

This following shows use of the RXTXEN signal to control a sequence. The RXTXEN signal can be tied high and left high. The sequence will then start based on writing of the bit rx\_strm.

The following is a typical sequence for stream receive operation not using a timer-based start (note that xcvr\_seq[1:0] field = 00):

1. The RX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. RX control bits must be initiated (Stream Mode and no timer):
  - a) tmr\_trig\_en = 0.
  - b) use\_strm\_mode = 1
  - c) tx\_strm = 0.
4. Program gpio\_alt\_en = 1 to enable GPIO1 as an “out-of-idle” indicator and GPIO2 as a CRC “valid” indicator.
5. rx\_rcvd\_mask, Control\_A Register 06, Bit 8 is programmed to “1” to enable an interrupt request when a word of RX payload data is ready to be read.
6. rx\_done\_mask, Control\_B Register 07, Bit 6 is programmed to “1” to enable an interrupt request when the RX packet has been fully received.
7. rx\_strm, Control\_A Register 06, Bit 11 is programmed to “1” to enable the stream RX sequence.
8. RXTXEN must be asserted and held high.
9. The receiver will await preamble followed by a SFD. After receiving these, the next byte is the FLI and gets stored in rx\_pkt\_latch[6:0]. The rx\_strm\_irq, IRQ\_Status Register 24, Bit 7 will get set and generates an interrupt request on  $\overline{\text{IRQ}}$ .
10. The MCU should read RX\_Status Register 2D to fetch the rx\_pkt\_latch[6:0] to clear the  $\overline{\text{IRQ}}$ . Note that LQI is valid at this time and can be fetched with the same read because cca\_final[7:0] is the LQI value and is RX\_Status Register 2D, Bits 15-8.

### NOTE

The MCU must respond to the interrupt within 64 microseconds and use the FLI data to determine the number of data fetches that will be required for the packet data.

11. As the payload data arrives, it gets stored in a 2-byte wide buffer and the rx\_strm\_irq gets set to indicate data is ready and also generates an interrupt request.
12. The microcontroller must respond to the interrupt by reading the RX data from rx\_pkt\_ram[15:0], RX\_Pkt\_RAM Register 01. Reading the data clears the interrupt request.

### NOTE

The MCU must respond to the interrupt request within 64 microseconds. Steps 10 and 11 get repeated as required for the payload data. No interrupt is generated and no data is available for the FCS word.

13. When a packet is successfully received, the following are reported:
  - a) `crc_valid`, IRQ\_Status Register 24, Bit 0 - reports the results of the CRC check, where a “1” indicates valid CRC. GPIO2 also reports valid CRC.
  - b) `cca_final[7:0]`, RX\_Pkt\_Latch Register 2D, Bits 15 - 8 - reports Link Quality Indicator.
  - c) `rx_done_irq`, IRQ\_Status Register 24, Bit 13 - reports the completion of packet reception, where a “1” indicates complete status. Also, an interrupt is generated due to the valid status.
14. In response of the interrupt request from the MC1321x, the microcontroller checks status to clear the interrupt and check the validity of the RX packet.

In between reading of the payload data, the MCU can be processing the data as it arrives. In this manner once the frame has been validated, the MCU can respond with a reply (if required) with a minimum amount of latency.

### 7.3.4.2 Aborting a Stream Receive Mode Sequence

It may be required to abort a stream receive sequence. The sequence should be aborted by negating RXTXEN to low. The RX sequence will be terminated and the transceiver will return to the Idle condition. The `rx_done_irq` status will be set and generate an interrupt (if enabled).

Because of the Streaming Data Mode, it is possible for a `strm_data_err` status/interrupt to occur up to 64  $\mu$ sec after the transceiver returns to Idle. Driver software must deal with this extraneous interrupt by not reacting to it other than to clear the status by reading the IRQ\_Status Register 24. One means of detecting that the interrupt is not valid is to monitor the Idle state of the transceiver through use of GPIO1 (out of idle) indicator, such that if the interrupt occurs while the transceiver is in idle, then the interrupt status must be cleared and ignored.

### 7.3.4.3 Stream Transmit Mode

The advantage of Stream Transmit Mode is that it allows the microcontroller to start a transmission without the latency of pre-loading the TX data into the TX Packet RAM. The disadvantage of Stream Mode is that there is a significant amount of overhead required from the MCU to process outgoing data on a word-by-word basis.

#### NOTE

This following shows use of the RXTXEN signal to control a sequence. The RXTXEN signal can be tied high and left high. The sequence will then start based on writing of the bit `tx_strm`.

The following is a typical sequence for stream transmit operation not using a timer-based start (note that `xcvr_seq[1:0]` field = 00):

1. The TX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. TX control bits must be programmed (Stream Mode and no timer):
  - a) `tmr_trig_en` = 0.
  - b) `use_strm_mode` = 1.

- c) rx\_strm = 0.
4. Program gpio\_alt\_en = 1 to enable GPIO1 as an “out-of\_idle” indicator and GPIO2 as a CRC “valid” indicator.
  5. tx\_sent\_mask, Control\_A Register 06, Bit 9 is programmed to “1” to enable an interrupt request when a word of TX payload data needs to be written.
  6. tx\_done\_mask, Control\_B Register 07, Bit 7 is programmed to “1” to enable an interrupt request when the TX packet has been fully sent.
  7. The MCU loads the value of the number of data bytes plus two (for FCS) into tx\_pkt\_length[6:0] TX\_Pkt\_Ctl Register 03, Bits 6 - 0.
  8. The MCU preloads the first data word into TX\_Pkt\_RAM Register 02
  9. tx\_strm, Control\_A Register 06, Bit 12 is programmed to “1” to enable the stream TX sequence.
  10. RXTXEN must be asserted and held high.
  11. The transmitter will turn-on, send preamble, the SFD, the FLI, and the first data word. When the transceiver is ready to accept another data word, the tx\_sent\_irq, IRQ\_Status Register 24, Bit 6, will go valid and generate an interrupt request asking for the next word of payload data.
  12. The microcontroller must respond to the interrupt by writing the TX data to tx\_pkt\_ram[15:0] TX\_Pkt\_RAM Register 02.

#### NOTE

Steps 11 and 12 get repeated as required for the payload data. A word must be transferred within 64  $\mu$ s to keep the packet contiguous. No data transfer is required for the FCS data because it is generated by the onboard transceiver.

13. When the payload data packet has been successfully transmitted followed by the CRC bytes, the MC1321x will generate a tx\_done\_irq, IRQ\_Status Register 24, Bit 14 that reports the completion of packet transmission, where a “1” indicate complete status. Also, an interrupt is generated due to the valid status.
14. In response of the interrupt request from the MC1321x, the microcontroller checks status to clear the interrupt and check the validity of the TX complete.

### 7.3.5 Clear Channel Assessment (CCA) Modes (including Link Quality Indication)

A special case of receive function called Clear Channel Assessment (CCA) modes is available to measure received energy from the selected channel. The CCA function exists as two algorithms:

1. Clear channel assessment - measuring channel energy and comparing to a preset threshold.
2. Energy detect (ED) - measuring channel energy and giving an indication of measured strength.

The energy detect algorithm is also used for Link Quality Indication (LQI) during a normal RX operation. The LQI is reported as part of the RX operation.

The CCA modes are associated with the following register fields:

1. `cca_type[1:0]`, Control\_A Register 06, Bits 5 - 4, determines channel energy assessment algorithm where value 0x1 selects CCA and value 0x2 selects energy detect.
2. `cca_vt[7:0]`, CCA\_Thresh Register 04, Bits 15 - 8, sets the threshold level for the CCA function.
3. The average power of the signal is displayed in field `cca_final[7:0]`, RX\_Pkt\_Latch Register 2D, Bits 15 - 8. This field is used for CCA, ED, and as LQI during an RX operation.
4. `power_comp[7:0]`, CCA\_Thresh Register 04, Bits 7-0, provides an offset that is added to the measured value of the average energy from a CCA/ED function or LQI value from an RX function.
5. Status bit `cca`, IRQ\_Status Register 24, Bit 1, is used only for the CCA algorithm and is set to “1” when a busy channel is detected.
6. Status bit `cca_irq_status`, IRQ\_Status Register 24, Bit 5, is set to “1” when the power measurement is complete for CCA or ED. If `cca_mask`, Control\_A Register 06, Bit 10, is set an interrupt request will be also generated when the `cca_irq_status` is set.
7. For CCA Mode, it is preferred that `tx_strm`, `rx_strm` and `use_strm_mode` control bits should be cleared to zero.

### 7.3.5.1 Clear Channel Assessment Function (use\_strm is zero)

The CCA function measures the average energy of the channel and compares it to a preset threshold as required by the 802.15.4 Standard. In CCA Mode, the receiver first warms-up from Idle in 144  $\mu$ s. The average value of the signal power, as measured over the next 128  $\mu$ s (8 symbol periods), is calculated and stored in `cca_final[7:0]`.

To determine the decimal equivalent of the value stored in `cca_final[7:0]`, one must convert the hex value to its decimal value, divide by two, and change the sign; where this calculated value is equivalent to the received signal strength in dBm:

$$\text{Signal strength in dBm} = - (\text{dec} (\text{cca\_final}[7:0] / 2))$$

The value stored in `cca_final[7:0]` is also affected by an offset stored in `power_comp[7:0]`, CCA\_Thresh Register 04, Bits 7-0. The default value of `power_comp[7:0] = 0x8D` and is added to the measured value during the CCA/ED/LQI function to give the stored value in `cca_final[7:0]`.

The compensation number added to the internal measured value is `power_comp[7:0] / 2` and the resulting addition is shown in Figure 7-5.

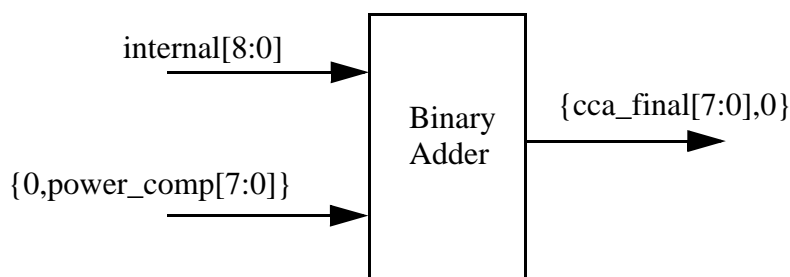


Figure 7-5. Compensation Factor Added to Internal CCA Value to Produce Corrected Final Value

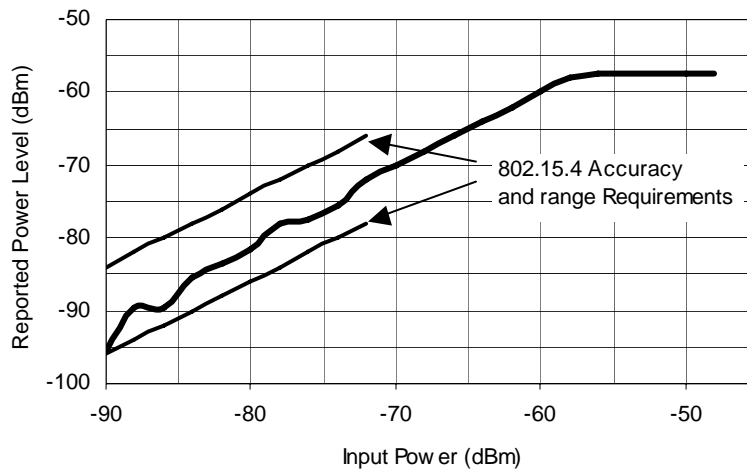
The resulting formula for dBm means that the power\_comp[7:0] value must be incremented or decremented by 4 to cause a 1 dBm change, which equates to a 1/4 dBm resolution for the power\_comp correction.

The power\_comp[7:0] default value is 0x8D and the value must be incremented to set a lower final CCA value. As an example, if one were putting in -30 dBm signal and the cca\_final[7:0] was reporting -26 dBm, then the correction value in power\_comp[7:0] should be increased by (4 \* delta) or (4 \* 4) in this case. The new compensated value of power\_comp[7:0] = 0x8D + 0x10 = 0x9D.

**NOTE**

As stated above, the default value for power\_comp[7:0] = 0x8D. For the MC1321x device an offset of 3.5 dBm is needed to center the reported CCA/LQI value over temperature. As a result, it is suggested that a value of 0x9B be programmed into power\_comp[7:0] for best accuracy.

Since the AGC is set to a fixed gain during the CCA procedure, input signals above -65 dBm will not be reflected correctly due to saturation. This is not a problem since the purpose of this measurement is to detect a signal above a threshold that is not to exceed -75 dBm. See the 802.15.4 Standard for details.



**Figure 7-6. CCA Reported Power Level vs. Input Power**

The value contained in cca\_final[7:0], is compared to the preset threshold of cca\_vt[7:0], CCA\_Thresh Register 04, Bits 15 - 8. If cca\_final[7:0] is equal to or less than the threshold, then cca is set to 1, indicating a busy channel. If cca\_final[7:0] is greater than the threshold, cca remains at 0. Once the CCA operation is complete, cca\_irq is asserted.

The value of cca\_vt[7:0] is calculated:

$$\text{Threshold value} = \text{hex} ( | (\text{Threshold Power in dBm}) * 2 | )$$

A suggested threshold is -82 dBm or 0xA4 = cca\_vt[7:0].

The following is a typical sequence for CCA operation (not using a timer-based start):

### NOTE

The control bit `use_strm_mode`, Control\_B Register 07, Bit 5 should be set to “0”.

1. The RX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. The CCA threshold must be programmed (`cca_vt[7:0] = 0xA4` as an example).
4. `cca_mask`, Control\_A Register 06, Bit 10 is programmed to “1” to enable an interrupt request when the CCA operation is complete.
5. `cca_type[1:0]`, Control\_A Register 06, Bits 5 - 4, is programmed to “01” to select the CCA algorithm.
6. Transceiver sequence is programmed to `xcvr_seq[1:0] = 0x1` for CCA Mode.
7. RXTXEN must be asserted and held high.
8. When the measurement is complete, the following are reported:
  - a) `cca_final[7:0]`, RX\_Pkt\_Latch Register 2D, Bits 15 - 8 - reports the average power level
  - b) `cca`, IRQ\_Status Register 24, Bit 1, is set to “1” when a busy channel is detected.
  - c) `cca_irq`, IRQ\_Status Register 24, Bit 5, is set to “1” to indicate complete status. Also, an interrupt is generated due to the valid status.
9. In response of the interrupt request from the MC1321x, the microcontroller does the following:
  - a) Determines the busy status of the channel by reading and checking `cca_irq` and `cca`.
  - b) If required the power level can be determined by reading `cca_final[7:0]`.
10. Negate TTXEN to low.

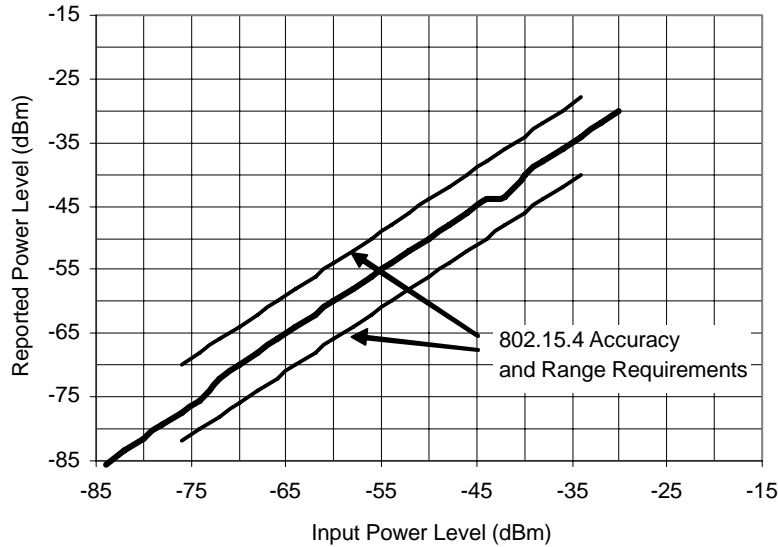
#### 7.3.5.2 Energy Detect Function (`use_strm` is zero)

With the energy detect algorithm, the exact same procedure results as the CCA operation without the threshold comparison. The receiver warms-up from Idle in 144  $\mu$ s and the received power is measured over the next 128  $\mu$ s (8 symbol periods). The average power is calculated and stored in `cca_final[7:0]`.

### NOTE

For the graph below, the required 802.15.4 Standard accuracy and range limits are shown. A 3.5 dBm offset has been programmed into the ED/LQI reporting level to center the level over temperature in the graph.





**Figure 7-7. ED and LQI Reported Power vs. Input Power**

Status bit `cca` is unaffected by energy detect. Once the energy detect operation is complete, `cca_irq` is asserted.

The following is a typical sequence for ED operation (not using a timer-based start):

#### NOTE

The control bit `use_strm_mode`, Control\_B Register 07, Bit 5 should be set to “0”.

1. The RX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. `cca_mask`, Control\_A Register 06, Bit 10 is programmed to “1” to enable an interrupt request when the CCA operation is complete.
4. `cca_type[1:0]`, Control\_A Register 06, Bits 5 - 4, is programmed to “10” to select the ED algorithm.
5. Transceiver sequence is programmed to `xcvr_seq[1:0] = 0x1` for CCA Mode.
6. RXTXEN must be asserted and held high.
7. When the measurement is complete, the following are reported:
  - a) `cca_final[7:0]`, RX\_Pkt\_Latch Register 2D, Bits 15 - 8 - reports the average power level
  - b) `cca_irq`, IRQ\_Status Register 24, Bit 5, is set to “1” to indicate complete status. Also, an interrupt is generated due to the valid status.
8. In response of the interrupt request from the MC1321x, the microcontroller can determine the power level by reading `cca_final[7:0]`.
9. Negate RXTXEN to low.

### 7.3.5.3 CCA / ED While in Stream Mode (use\_strm is one)

It is recommended that a CCA or ED operation be performed while use\_strm control bit is not active. However, for software timing considerations, there are times that it may be desirable to leave use\_stream active and a CCA/ED can be done through an alternate procedure.

The CCA / ED operation is done in a manner similar to that described in the previous two Sections. However, before asserting RXTXEN (starting the sequence), the MCU should write Register 38 to the value 0x01FF. This register value should only be used for the CCA or ED operation and not for an RX operation. The default value of 0x0008 should be written to Register 38 be initiating an RX operation.

### 7.3.5.4 Link Quality Indication

Link Quality Indication is a measure of the signal quality during an actual receive operation. Its value is stored in field cca\_final[7:0], RX\_Pkt\_Latch Register 2D, Bits 15 - 8. In Stream Mode, note that LQI is valid and can be fetched with the same read as when fetching rx\_pkt\_latch[6:0].

The format for the LQI is the same as CCA:

Signal strength in dBm = - (dec (cca\_final[7:0] / 2))

#### NOTE

In Figure 7-7, the required 802.15.4 Standard accuracy and range limits are shown. A 3.5 dBm offset has been programmed into the LQI reporting level to center the level over temperature. Typical values of LQI returned from an RX operation are from about -95 dBm to about -18 dBm giving a cca\_final[7:0] range of decimal values 190 (0xBE) to 36 (0x24). When used in an 802.15.4 application, the LQI must be scaled and limited to have a range of decimal 0 to 255. Values of LQI above approximately -30 to -25 dBm can be non-linear and should not be used as an indication of absolute received power.

## 7.4 Frequency of Operation

The MC1321x is designed to operate in the 2.4 GHz band, covering 16 channels and using 5 MHz of spacing between each channel. The MC1321x uses two local oscillators (LO). The first LO synthesizer is the main LO for the receiver and the carrier generator for the transmitter. This block is comprised of a Fractional-N (Frac-N) PLL frequency synthesizer.

The fractional and integer components of the Frac-N must be programmed properly to perform a transceiver operation on a particular channel. The integer setting is LO1\_Int\_Div Register 0F, Bits 7-0 (lo1\_idiv[7:0]) and the fractional setting is LO1\_Num Register 10 (lo1\_num[15:0]). For programming a desired frequency  $f_{\text{channel}}$ :

Where  $([N+1].\text{frac})_{\text{dec}}$  is the RFrac-N divider (in decimal) consisting of integer “N” and fraction “frac” -

$$f_{\text{channel}} = ([N+1].\text{frac})_{\text{dec}} * 16 \text{ MHz}$$

After finding  $([N+1].\text{frac})_{\text{dec}}$  the register settings become:

- Integer  $lo1\_div[7:0] = N_{hex}$
- Fractional  $lo1\_num[15:0]$  is found by converting the decimal fraction to a 16-bit hexadecimal fraction -

$$frac_{dec} = lo1\_num_{dec} / 65536 \text{ or}$$

$$lo1\_num_{dec} = 65536 * frac_{dec} \text{ then,}$$

$$lo1\_num[15:0] = lo1\_num_{hex}$$

As an example use Channel 23 (desired frequency is 2465 MHz):

$$([N+1].frac)_{dec} = 2465 \text{ MHz} / 16 \text{ MHz} = 154.0625$$

- $N+1 = 154_{dec}$  and  $N = 153_{dec}$ , therefore,  $lo1\_div[7:0] = 0x99$
- $lo1\_num_{dec} = 65536 * 0.0625 = 4096_{dec}$ , therefore,  $lo1\_num[15:0] = 0x1000$

The channels and the respective required bit values of the integer settings LO1\_Int\_Div Register 0F, Bits 7-0 ( $lo1\_div[7:0]$ ) and the fractional settings LO1\_Num Register 10 ( $lo1\_num[15:0]$ ) are shown in [Table 5-18](#).

## 7.5 Transmit Power Adjustment

The PA output power can be controlled via programming of the PA\_Lvl Register 12[7:0], whose fields include  $pa\_lvl\_course[1:0]$ ,  $pa\_lvl\_fine[1:0]$ ,  $pa\_dr\_coarse[1:0]$ , and  $pa\_dr\_fine[1:0]$ . The programmable range of differential power is typically -27 dBm to +3 dBm. [Table 7-3](#) shows the device power output versus SPI register settings for PA\_Lvl Register 12[7:0].

### NOTE

Register 12[7:0] value of 0xBC is the default setting and yields a nominal power out of -1 dBm to 0 dBm. Also, common practice is to allow Register 12[3:0] to stay at its default value of 0xC and only adjust the  $pa\_lvl\_course$  and  $pa\_lvl\_fine$  fields as required; this allows a range of about -16 dBm to +1.4 dBm.

**Table 7-3. MC1321x Power Output vs. SPI Settings (Register 12)**

PA Power Adjust Reg 12[7:0] (Hex)	Typical Differential Power at Output Contact (dBm)	Typical PA Current (mA)	Comments
00	-28.7	20.9	
04	-22.0	21.7	
08	-18.5	22.9	
0C	-16.2	25.0	Reg 12[3:0] default
1C	-15.9	25.1	Reg 12[3:0] default
2C	-15.3	25.1	Reg 12[3:0] default
3C	-14.8	25.1	Reg 12[3:0] default

**Table 7-3. MC1321x Power Output vs. SPI Settings (Register 12) (continued)**

PA Power Adjust Reg 12[7:0] (Hex)	Typical Differential Power at Output Contact (dBm)	Typical PA Current (mA)	Comments
4C	-8.5	25.9	Reg 12[3:0] default
5C	-7.6	26.0	Reg 12[3:0] default
6C	-7.2	26.1	Reg 12[3:0] default
7C	-7.0	26.2	Reg 12[3:0] default
8C	-1.7	28.0	Reg 12[3:0] default
9C	-1.6	28.3	Reg 12[3:0] default
AC	-0.77	28.6	Reg 12[3:0] default
BC (default)	-0.66	28.8	Reg 12[7:0] default
CC	0.62	30.6	Reg 12[3:0] default
DC	1.19	31.9	Reg 12[3:0] default
EC	1.23	34.9	Reg 12[3:0] default
FC	1.42	35.3	Reg 12[3:0] default
FD	2.2	35.0	
FE	2.9	35.0	
FF	3.4	35.0	

## 7.6 2.4GHz PLL Out-of-Lock Interrupt

Successful wireless data transmission and reception is predicated on the proper channel frequency being maintained internally by the modem. Sophisticated control circuitry and design techniques assure that the internal 2.4GHz local oscillator stays on the selected channel frequency. In the unusual event that the PLL loses lock, the host is notified via the 'Out-of-Lock Interrupt'. If lock indication circuitry indicates an out-of-lock condition, status bit `pll_lock_irq`, `IRQ_Status` Register 24, Bit 15, is set and any CCA, RX or TX operation in progress is automatically terminated. The modem returns immediately to the IDLE state to await interrupt service routine handling. Also, the  $\overline{\text{IRQ}}$  is asserted provided the mask bit `pll_lock_mask`, `IRQ_Mask` Register 05, Bit 9, is set.

### NOTE

Freescale recommends that software enable the `pll_lock_mask` during CCA, RX, and TX operations. The `pll_lock_irq` status bit is set by an out-of-lock condition and MUST be cleared by an `IRQ_Status` Register read before another CCA, RX, or TX operation can be enabled. As already stated, an out-of-lock condition aborts the present operation and if the `pll_lock_irq` status is not cleared, any subsequent CCA, RX, or TX operation requested is immediately aborted.

Examples of where this could be troublesome is in Packet Mode RX or a CCA operation. If the RX or CCA is aborted due to an out-of-lock condition, no rx\_done\_irq status or cca\_irq status is set and a corresponding IRQ signal is not asserted, and as a result, no interrupt is generated unless the pll\_lock\_irq is enabled to generate an interrupt.



# Chapter 8

## Modem Interrupt Description

### 8.1 Modem Interrupts

Interrupts provide a way for the MC1321x to inform the host microcontroller (MCU) of onboard events without requiring the MCU to constantly query MC1321x status.

For a given event, the interrupt flow is as follows.

- The source interrupt mask is enabled
- The source function is enabled
- The source event occurs causing the source status flag to be set and the  $\overline{\text{IRQ}}$  pin to be asserted low
- The  $\overline{\text{IRQ}}$  pin stays asserted until the  $\overline{\text{IRQ}}$ \_Status Register is read to determine the source of the interrupt. Reading the IRQ\_Status Register clears the status bits and releases the  $\overline{\text{IRQ}}$  signal to be negated high

When multiple source events have occurred, the MCU must use the IRQ\_Status register contents to determine all the present events that caused an interrupt, prioritize them, and respond to all of the them. This is done through the interrupt service routine of the MCU.

#### 8.1.1 Modem Interrupt Sources

Table 8-1 lists the interrupt status bits, mask bits, and source description.

**Table 8-1. Modem Interrupt Sources**

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism <sup>1</sup>
1	pll_lock_irq	pll_lock_mask	PLL out of lock.	Read IRQ_Status Reg <sup>2</sup>
2	ram_addr_err / tx_done_irq	ram_addr_mask / tx_done_mask	1. RAM address error - When in Packet Data Mode, a recursive access to Packet RAM has exceeded the maximum RAM address. The ram_addr_mask controls the interrupt generation for this mode. 2. TX Done - When in Streaming Data Mode, the TX operation is complete and transceiver has returned to Idle Mode. The tx_done_mask controls interrupt generation in this mode.	Read IRQ_Status Reg (for either mode)

**Table 8-1. Modem Interrupt Sources (continued)**

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism <sup>1</sup>
3	arb_busy_err / rx_done_irq	arb_busy_mask / rx_done_mask	<p>1. Arbitration busy error - When in Packet Data Mode, a SPI access to Packet RAM was attempted during packet reception or transmission. The arb_busy_mask controls the interrupt generation for this mode.</p> <p>2. RX Done - When in Streaming Data Mode, the RX reception is complete and the transceiver has returned to Idle Mode. The rx_done_mask controls interrupt generation in this mode.</p>	Read IRQ_Status Reg (for either mode)
4	strm_data_err	strm_data_mask	<p>1. For RX Stream Data Mode, a new received data word has been received before the previous RX data word was read (this is a data overrun condition). An interrupt will not be generated on the last transfer of a packet if it is not read.; if this occurs the Rx Stream Data Ready status will still be set.</p> <p>2. For TX Stream Data Mode, the current TX data word transmission was complete before next TX data word was written (this is a data underrun condition).</p>	Read IRQ_Status Reg
5	attn_irq	attn_mask	The $\overline{\text{ATTN}}$ signal has been asserted or the modem has reached a Power-up complete condition after a reset.	Read IRQ_Status Reg
6	doze_irq	doze_mask	While in Doze Mode, a tmr_cmp2 match has occurred and the MC1321x will return to Idle Mode.	Read IRQ_Status Reg
7	rx_rcvd_irq/ rx_strm_irq	rx_rcvd_mask	<p>1. Rx_rcvd_irq - When in Packet Data Mode, the current RX packet has been received, data in Packet RAM is ready to be read, and the transceiver has returned to Idle Mode.</p> <p>2. Rx_strm_irq - When in Streaming Data Mode, a data word is ready to be read:</p> <p>a) First occurrence - RX Packet Length is available to be read.</p> <p>b) Subsequent occurrences - next RX stream data word is ready to be read</p>	<p>1. Read IRQ_Status Reg</p> <p>2a) First occurrence - read RX Packet Length from Register 2D</p> <p>2b) Subsequent occurrences - read RX data word from Register 01</p>
8	tx_sent_irq/ tx_strm_irq	tx_sent_mask	<p>1. Tx_sent_irq - When in Packet Data Mode, the current TX packet in Packet RAM has been completely transmitted, and the transceiver has returned to Idle Mode.</p> <p>2. Tx_strm_irq - When Streaming Data Mode, transceiver is ready for next TX data word to be written.</p>	<p>1. Read IRQ_Status Reg</p> <p>2. Write TX data to Register 02</p>
9	cca_irq	cca_mask	The Clear Channel Assessment operation has been completed.	Read IRQ_Status Reg



**Table 8-1. Modem Interrupt Sources (continued)**

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism <sup>1</sup>
10	tmr1_irq	tmr1_mask	Tmr_cmp1 match has been made.	Read IRQ_Status Reg or set tmr_cmp1_dis bit
11	tmr2_irq	tmr2_mask	Tmr_cmp2 or tc2_prime match has been made. (Not functional when Tmr_cmp2 is used to exit Doze Mode)	Read IRQ_Status Reg or set tmr_cmp2_dis bit
12	tmr3_irq	tmr3_mask	Tmr_cmp3 match has been made.	Read IRQ_Status Reg or set tmr_cmp3_dis bit
13	tmr4_irq	tmr4_mask	Tmr_cmp4 match has been made.	Read IRQ_Status Reg or set tmr_cmp4_dis bit

<sup>1</sup> Although some status bits can be cleared by other means, reading IRQ\_Status register will always clear all status bits.

<sup>2</sup> If a pll\_lock\_irq status is set, the status must be cleared before a subsequent CCA, RX or TX operation is allowed (the requested operation will immediately abort).

## 8.1.2 Output Pin $\overline{\text{IRQ}}$

The  $\overline{\text{IRQ}}$  signal is an open drain output that is asserted low when an interrupt request is pending. The signal is released to high by reading the IRQ\_Status register via an SPI transaction.  $\overline{\text{IRQ}}$  is an open drain output that requires a passive pullup and it also can be programmed for drive strength.

### 8.1.2.1 Programming $\overline{\text{IRQ}}$ Pullup

A passive pullup is required on  $\overline{\text{IRQ}}$  and may be done via two methods:

1. Use the onboard (nominal 40 Kohm) pullup resistor - Set irqb\_pup\_en bit, GPIO\_Data\_Out Register 0C, Bit 7, to activate. This is the default mode.
2. Use an external resistor (value should be greater than 4 kilohms).

### 8.1.2.2 Setting $\overline{\text{IRQ}}$ Output Drive Strength

$\overline{\text{IRQ}}$  output drive strength is programmed by writing to irqb\_drv[1:0], GPIO\_Data\_Out Register 0C. There are 4 levels of drive strength with field value 00 for lowest and value 11 for greatest. The default value is 11.

#### NOTE

It is suggested the user leave  $\overline{\text{IRQ}}$  at greatest drive strength for best performance.

## 8.2 PLL\_lock\_irq Status Bit and Operation

As described in [Section 5.28, “IRQ\\_Status - Register 24”](#), pll\_lock\_irq status bit indicates the LO1 PLL has come out of lock during a TX, RX, or CCA (ED) transceiver operation. If the LO1 unlocks during the transceiver operation, the device returns to Idle mode, and the pll\_lock\_irq status bit gets set as expected. The application software must read the IRQ\_Status Register 24 (clearing the pll\_lock\_irq status bit) before

attempting any further transceiver active operations (TX, RX or CCA). If the status bit is not cleared, any subsequent active operation will abort immediately. This condition occurs because the LO1 unlock causes an operation abort and the status bit must be cleared or any follow-on operation will also abort.

The best practice is to enable the `pll_lock_irq` interrupt so that  $\overline{\text{IRQ}}$  will be asserted if an unlock occurs.

- If the `pll_lock_irq` interrupt is not enabled and an LO1 unlock occurs, no `rx_rcvd_irq` status will be set for an RX operation nor will a `cca_irq` status be set for a CCA operation because the operation was aborted. As a result, an interrupt cannot be generated, and any follow-on operation will be aborted.
- If the `pll_lock_irq` interrupt is not enabled for a TX operation and an unlock occurs, the `tx_sent_irq` status will be set when the TX aborts, so an interrupt can be generated. However, any follow-on operation will still be aborted if the `IRQ_Status` Register is not read.

### 8.3 Attn\_irq Status Bit and Interrupt Operation

As described in [Section 5.28, “IRQ\\_Status - Register 24”](#), `attn_irq` status bit indicates:

- The transceiver has achieved Idle status (full power-up) after the release of the  $\overline{\text{RST}}$  signal. The default condition out of reset leaves the `attn_irq` interrupt request enabled, and upon the transceiver reaching Idle, the `attn_irq` status is set and the `IRQ` signal is asserted
- Signal `ATTN` has been asserted (normally to release the transceiver from Hibernate or Doze mode) and the transceiver has exited the low power mode. The  $\overline{\text{IRQ}}$  signal will be asserted if the interrupt has not been masked

### 8.4 Interrupts from Exiting Low Power Modes

The modem has three low power modes and interrupt generation differs somewhat for each mode.

#### 8.4.1 Exiting Off Mode (Reset)

The transceiver is put in reset and stays in reset (Off Mode) through the assertion of  $\overline{\text{RST}}$ . The initialization done at reset enables `attn_mask` which allows an interrupt request when `attn_irq` is set. One condition that sets `attn_irq` is when the transceiver exits reset after  $\overline{\text{RST}}$  is released high. As a result, an interrupt request will always be generated by `attn_irq` status when reset is exited.

#### 8.4.2 Exiting Hibernate Mode

Hibernate is normally only exited through assertion of  $\overline{\text{ATTN}}$  (obviously  $\overline{\text{RST}}$  can still override). The `attn_irq` status will be set by the assertion of  $\overline{\text{ATTN}}$ . If an interrupt is desired to signify the event, the `attn_mask` bit must be set before entering Hibernate. The interrupt request will then be generated due to the `attn_irq` being set true upon exit from Hibernate.

### 8.4.3 Exiting Doze Mode(s)

Doze can be exited via assertion of  $\overline{\text{ATTN}}$  or through use of `tmr_cmp2` (again reset can override). Asserting  $\overline{\text{ATTN}}$  will always cause Doze to be exited even if the timer option is enabled. If an interrupt is desired, set `attn_mask` before entering Doze which will cause the interrupt when the `attn_irq` status is set upon exiting Doze due to  $\overline{\text{ATTN}}$ .

Alternately, Doze has the option of using `tmr_cmp2` to exit, except for Acoma Mode which cannot use the timer. When `tmr_cmp2` match occurs the `doze_irq` status will be set. An interrupt request will also occur if `doze_mask` bit has been enabled.



## Chapter 9

# Modem Miscellaneous Functions

### 9.1 Reset Function

The MC1321x can be placed in one of two reset conditions either through hardware input  $\overline{M\_RSTB}$  or by writing to Reset Register 00.

#### 9.1.1 Input Pin $\overline{M\_RSTB}$

Asserting input pin  $\overline{M\_RSTB}$  low places the transceiver in a complete reset condition (Off Mode and power down), and the device stays in this reset mode until  $\overline{M\_RSTB}$  is released high. After  $\overline{M\_RSTB}$  is released, the transceiver will transition to the Idle Mode within 10-25 milliseconds, causing an  $\overline{ATTN}$  interrupt request and allowing CLK0 to start at 32.768+ kHz (both of which are default conditions).

#### 9.1.2 Software Reset (Writing to Register 00)

Writing to Reset Register 00 causes a reset condition where the digital logic is reset, but the transceiver is not powered down. The device is forced to the Idle Mode and the SPI registers are all reset and forced to their default condition although all data in the Packet RAMs is retained. The reset is held as long as  $\overline{CE}$  remains asserted and is released when  $\overline{CE}$  is negated high.

#### 9.1.3 Reset Indicator Bit (RST\_Ind Register 25, Bit 7)

It is useful to determine if the transceiver has powered-up from a reset condition or from a low power state that was released via the  $\overline{ATTN}$  signal. The reset indicator bit (reset\_ind, RST\_Ind Register 25, Bit 7) is cleared during a reset operation but not during a low power mode such as Doze or Hibernate. The reset\_ind bit gets set by the first read of Register 25 after a reset operation and stays set until another reset operation.

When exiting reset, an interrupt is generated by attn\_irq, IRQ\_Status Register 24, Bit 10, (the default condition is with the interrupt mask enabled). This same interrupt can be enabled for exiting Hibernate or Doze via an  $\overline{ATTN}$  assertion. As a result, the reset\_ind bit can determine if the power-up condition is from the reset condition or a Doze or Hibernate condition.

After exiting reset and responding to the attn\_irq interrupt, users should read Register 25 which in turn sets the reset\_ind bit. Thereafter, if the transceiver is put into Doze or Hibernate and then later awakened by an  $\overline{ATTN}$  assertion, the attn\_irq interrupt is also used, but the reset\_ind is set signifying that the chip was not reset and does not need re-initialized.

## 9.2 General Purpose Input/Output

The MC1321x has seven (7) general purpose input/output (GPIO) pins (GPIO1 through GPIO7). Features include:

- CMOS logic levels with +/- 1 mA load current
- Programmable as inputs or outputs
- During reset outputs are disabled and exit reset as inputs
- Not capable of generating an interrupt
- Once programmed as an output, a GPIO keeps its state if the transceiver transitions to Doze or Hibernate Mode

### 9.2.1 Configuring GPIO Direction

The GPIO are configured using GPIO\_Dir Register 0B. Each I/O has a `gpiox_oen` output enable bit and a `gpiox_ien` input enable bit. Exiting reset, the default condition for these enable bits is that the `gpiox_ien` bits are set to 1 which enables the pins as inputs and `gpiox_oen` are cleared.

#### NOTE

If any bit is programmed to be an input and output simultaneously, the input condition overrides.

### 9.2.2 Setting GPIO Output Drive Strength

If any GPIO are programmed as outputs, their drive strength is programmable. GPIO1 through GPIO4 are programmed as a group for drive strength by writing to control field `gpio1234_drv[1:0]`, GPIO\_Dir Register 0B, and GPIO5 through GPIO7 are programmed as a group by writing to `gpio567_drv[1:0]`, GPIO\_Data\_Out Register 0C. There are 4 levels of drive strength with field value 00 for lowest and value 11 for greatest.

### 9.2.3 Programming GPIO Output Value

GPIO\_Data\_Out Register 0C has a `gpiox_o` bit for each GPIO pin that establishes the corresponding output's state when that I/O is programmed as an output. Setting a `gpiox_o` to 1 sets the output high.

### 9.2.4 Reading GPIO Input State

GPIO\_Data\_In Register 28 has a `gpiox_i` bit for each GPIO pin. When a GPIO is programmed as an input, its state can be determined by reading the corresponding `gpiox_i` bit in the register.

## 9.2.5 GPIO1 and GPIO2 as Stream Mode Status Indicators

To support easier and quicker status indication for the MCU, GPIO1 and GPIO2 can be programmed for special alternative functionality. If the `gpio_alt_en` bit of `Control_C Register 09` is set to 1 then:

1. GPIO1 becomes an “Out of Idle” indicator (active high) - GPIO1 will always reflect the status of the internal state machine. If the MC1321x is in a TX or RX or CCA/ED sequence, the GPIO1 will be high. Once the sequence ends, the GPIO1 returns to a low state and shows that the transceiver has returned to Idle. In Doze or Hibernate Mode GPIO1 stays low.
2. GPIO2 becomes a “Valid CRC” or “Valid CCA Result” indicator (active high) -
  - a) For a RX sequence, the GPIO2 will show the CRC is valid once the RX operation is complete and GPIO1 goes to low to indicate a return to Idle condition. The transition of GPIO1 from high to low latches the CRC result on GPIO2 and that status will not change until the next transceiver sequence. GPIO2 will not be valid if an error condition such as a PLL out-of-lock condition occurs.
  - b) For a CCA sequence, the GPIO2 will show if the CCA is valid once the CCA operation is complete and GPIO1 goes to low to indicate a return to Idle condition. The transition of GPIO1 from high to low latches the CCA result on GPIO2 and that status will not change until the next transceiver sequence. GPIO2 will not be valid if an error condition such as a PLL out-of-lock condition occurs.

These two signals can be used by the MCU to monitor transceiver status without interrogating the onboard status registers.

### NOTE

GPIO1 and GPIO2 should also be programmed as outputs for this function.

## 9.2.6 GPIO in Off, Hibernate, and Doze Modes

The GPIO can affect excess leakage current during low power modes. The best practice is to tie unused GPIO pins to ground through a resistor. In Hibernate and Doze, the state of the signals are retained and a GPIO programmed as an output will remain an output. Programming outputs low uses the absolute lowest power.

The Off condition can present a problem. In the Off Mode the GPIO resort to inputs and can float if not tied down which can cause excessive leakage current (the reason for the off condition is lowest current). This is why the best practice is to tie unused inputs low. Also, if a GPIO is used to interface to another device, a pullup or pull-down resistor should be in place to hold the GPIO pin to a known state.

## 9.3 Crystal Oscillator

The crystal oscillator for the MC1321x uses the following external pins:

1. XTAL1 - reference oscillator input.
2. XTAL2 - reference oscillator output. Note that this pin should not be loaded to be used as a reference source or to measure frequency; instead use CLK0 to measure or supply 16 MHz.

The external crystal circuit is shown in [Figure 9-1](#).

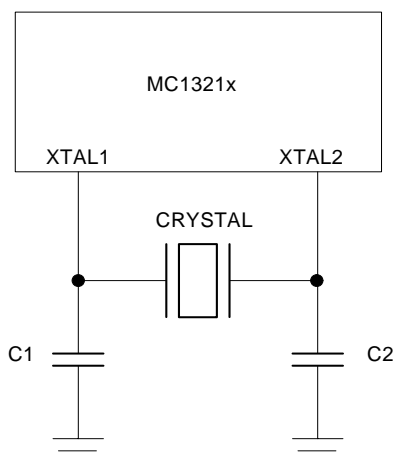


Figure 9-1. Crystal Oscillator Circuit

### 9.3.1 Crystal Requirements

The MC1321x requires that only a 16 MHz crystal with a  $<9$  pF load capacitance can be used. The load capacitance limitation is required due to internal oscillator circuit and the ability to trim the oscillator as described in the next section. A tight frequency tolerance on the crystal may also be required due to the 802.15.4 Standard which demands that frequency tolerances be kept within  $\pm 40$  ppm. This requirement is for the oscillator circuit, not just the crystal. This is covered in detail in the appropriate MC1321x Data Sheet.

### 9.3.2 Crystal Trim Operation

The MC1321x uses the 16 MHz crystal oscillator with warp capability as the reference oscillator for the system. The warp capability is done by the MC1321x and is controlled by programming CLKO\_Ctl Register 0A, Bits 15-8 (xtal\_trim[7:0]). The trimming procedure varies the frequency by a few hertz per step, depending on the type of crystal. The high end of the frequency spectrum is set when xtal\_trim[7:0] is set to zero. As xtal\_trim[7:0] is increased, the frequency is decreased. Accuracy of this feature can be observed by varying xtal\_trim[7:0] and using a spectrum analyzer or frequency counter to track the change in frequency of the crystal signal. The reference oscillator frequency can be measured at the CLKO contact by programming CLKO\_Ctl Register 0A, Bits 2-0, to value 000. The crystal frequency should not be monitored at IC pins 26 or 27 (XTAL1 or XTAL2) because this will load the oscillator and alter the frequency.



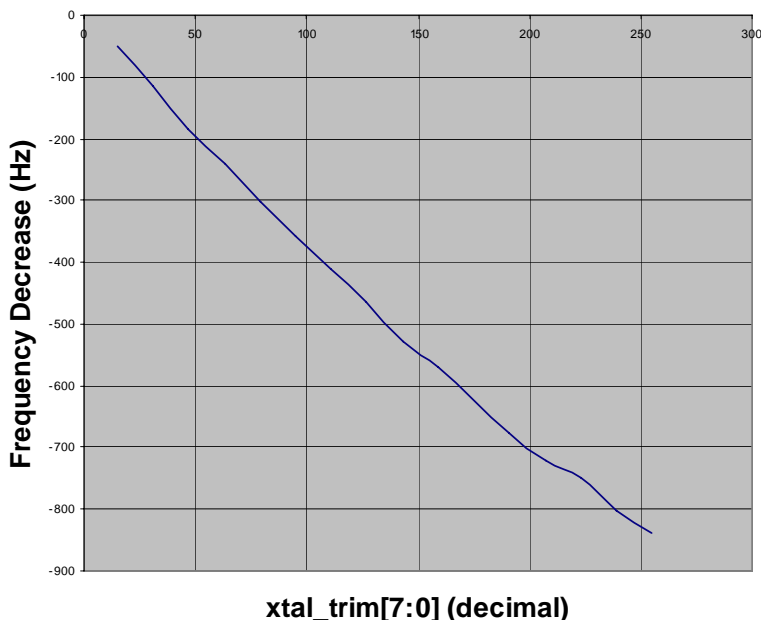


Figure 9-2. Crystal Frequency Variation vs. xtal\_trim[7:0]

Figure 9-2 shows typical oscillator frequency decrease versus the value programmed in xtal\_trim[7:0].

## 9.4 Output Clock Pin CLKO

The MC1321x can supply a clock output useful as a frequency source for a microcontroller, frequency test point, or reference for other uses. The clock output is available on signal CLKO and can be turned on or off as a power saving measure (default is CLKO active). CLKO is controlled by a number of control fields.

### 9.4.1 Enable CLKO (clko\_en, Control\_C Register 09, Bit 5)

Setting clko\_en, Control\_C Register 9, Bit 5, to 1 enables the CLKO signal. The default condition out of reset is that the clock out is enabled at the default frequency of 32.768+ kHz set by field clko\_rate[2:0].

### 9.4.2 Setting CLKO frequency (clko\_rate[2:0], CLKO\_Ctl Register 0A, Bits 2-0)

The 3-bit field clko\_rate[2:0], CLKO\_Ctl Register 0A, Bits 2-0, selects the output frequency based on the programmed value. Frequencies from 16 MHz to 16 kHz are available. Default frequency is 32.768+ kHz with a field value of clko\_rate[2:0] = 110. Table 5-13 lists the CLKO frequencies versus clko\_rate[2:0] program value.

### 9.4.3 Enable CLKO During Doze Mode (clk<sub>o</sub>\_doze\_en, Control\_B Register 07, Bit 9)

Bit clk<sub>o</sub>\_doze\_en, Control\_B register 07, Bit 9, is used to control CLKO during Doze Mode. If clk<sub>o</sub>\_doze\_en is set to 1 before entering Doze Mode, CLKO will continue to toggle while the MC1321x is in Doze Mode. The CLKO frequency must be set for 1 MHz or lower. Default out of reset is clk<sub>o</sub>\_doze\_en = 0 as a power-saving measure.

If clk<sub>o</sub>\_doze\_en = 0 and CLKO was enabled, CLKO will stop toggling 128 reference clock (16 MHz) cycles after the doze\_en bit is programmed to 1. CLKO will automatically re-start after exiting Doze.

### 9.4.4 Setting CLKO Output Drive Strength (clk<sub>o</sub>\_drv[1:0], GPIO\_Data\_Out Register 0C, Bits 11-10)

The CLKO output drive strength can be programmed to 4 different levels by writing to clk<sub>o</sub>\_drv[1:0], GPIO\_Data\_Out Register 0C, Bits 11-10. The default value is the lowest drive value of 00. Note that for higher frequencies such as 16 MHz, the CLKO must be programmed for highest drive. Table 9-1 shows output drive strength for maximum frequency and maximum load capacitance.

Table 9-1. CLKO Drive Strength Versus clk<sub>o</sub>\_drv[1:0] Value

Drive Strength (clk <sub>o</sub> _drv[1:0])	Max Freq (MHz)	Max C <sub>load</sub> (pF)
00	1	20
01	8	20
10	16	20
11	16	20 < C <sub>load</sub>

## 9.5 Input Pin $\overline{\text{ATTN}}$

The attention or  $\overline{\text{ATTN}}$  signal is used to exit either Doze Mode or Hibernate Mode.

### NOTE

Doze Mode may also be exited via a tmr\_cmp2 compare event. A transition event from high to low (assertion) on  $\overline{\text{ATTN}}$  will always exit either mode.

The  $\overline{\text{ATTN}}$  assertion low event can also generate an interrupt. The interrupt status bit is attn\_irq, IRQ\_Status Register 24, Bit 10, and the interrupt mask bit is attn\_mask, IRQ\_Mask Register 05, Bit 15.

# Chapter 10

## MCU Modes of Operation

### 10.1 Introduction

The operating modes of the HCS08 are described in this section. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

### 10.2 Features

- Active background mode for code development
- Wait Mode:
  - CPU shuts down to conserve power
  - System clocks running
  - Full voltage regulation maintained
- Stop modes:
  - System clocks stopped; voltage regulator in standby
  - Stop1 — Full power down of internal circuits for maximum power savings
  - Stop2 — Partial power down of internal circuits, RAM contents retained
  - Stop3 — All internal circuits powered for fast recovery

### 10.3 Run Mode

This is the normal operating mode for the HCS08. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at \$FFFE:\$FFFF after reset.

### 10.4 Active Background Mode

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

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low at the rising edge of reset
- When a BACKGROUND command is received through the BKGD pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint

- When encountering a DBG breakpoint

After entering Active Background Mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user's 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 while 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's 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 HCS08 is shipped from the Freescale Semiconductor factory, the FLASH program memory is erased by default unless specifically noted so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The Active Background Mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the Active Background Mode, refer to [Chapter 21, "Development Support"](#).

## 10.5 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 leading 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.

## 10.6 Stop Modes

One of three stop modes is entered upon execution of a STOP instruction when the STOPE bit in the system option register is set. In all stop modes, all internal clocks are halted. If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU will not enter any of the stop modes and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in SPMSC2.

### NOTE

Unused GPIO including signals not pinned-out must be initialized for low power operation.

Table 10-1 summarizes the behavior of the MCU in each of the stop modes.

**Table 10-1. Stop Mode Behavior**

Mode	PDC	PPDC	CPU, Digital Peripherals, FLASH	RAM	ICG	ATD	Regulator	I/O Pins	RTI
Stop1	1	0	Off	Off	Off	Disabled <sup>1</sup>	Off	Reset	Off
Stop2	1	1	Off	Standby	Off	Disabled	Standby	States held	Optionally on
Stop3	0	Don't care	Standby	Standby	Off <sup>2</sup>	Disabled	Standby	States held	Optionally on

<sup>1</sup> Either ATD Stop Mode or Power-Down Mode depending on the state of ATDPU.

<sup>2</sup> Crystal oscillator can be configured to run in Stop3. Please see the ICG registers.

### 10.6.1 Stop1 Mode

The Stop1 Mode provides the lowest possible standby power consumption by causing the internal circuitry of the MCU to be powered down. Stop1 can be entered only if the LVD circuit is not enabled in stop modes (either LVDE or LVDSE not set).

When the MCU is in Stop1 Mode, all internal circuits that are powered from the voltage regulator are turned off. The voltage regulator is in a low-power standby state, as is the ATD.

Exit from Stop1 is performed by asserting either of the wake-up pins on the MCU:  $\overline{\text{RESET}}$  or IRQ. IRQ is always an active low input when the MCU is in Stop1, regardless of how it was configured before entering Stop1.

Entering Stop1 Mode automatically asserts LVD. Stop1 cannot be exited until  $V_{DD} > V_{LVDH/L}$  rising ( $V_{DD}$  must rise above the LVI rearm voltage).

Upon wake-up from Stop1 Mode, the MCU will start up as from a power-on reset (POR). The CPU will take the reset vector.

## 10.6.2 Stop2 Mode

The Stop2 Mode provides very low standby power consumption and maintains the contents of RAM and the current state of all of the I/O pins. Stop2 can be entered only if the LVD circuit is not enabled in stop modes (either LVDE or LVDSE not set).

Before entering Stop2 Mode, the user must save the contents of the I/O port registers, as well as any other memory-mapped registers they want to restore after exit of Stop2, to locations in RAM. Upon exit of Stop2, these values can be restored by user software before pin latches are opened.

When the MCU is in Stop2 Mode, all internal circuits that are powered from the voltage regulator are turned off, except for the RAM. The voltage regulator is in a low-power standby state, as is the ATD. Upon entry into Stop2, the states of the I/O pins are latched. The states are held while in Stop2 Mode and after exiting Stop2 mode until a 1 is written to PPDACK in SPMSC2.

Exit from Stop2 is performed by asserting either of the wake-up pins:  $\overline{\text{RESET}}$  or IRQ, or by an RTI interrupt. IRQ is always an active low input when the MCU is in Stop2, regardless of how it was configured before entering Stop2.

Upon wake-up from Stop2 Mode, the MCU will start up as from a power-on reset (POR) except pin states remain latched. The CPU will take the reset vector. The system and all peripherals will be in their default reset states and must be initialized.

After waking up from Stop2, the PPDF bit in SPMSC2 is set. This flag may be used to direct user code to go to a Stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to PPDACK in SPMSC2.

To maintain I/O state for pins that were configured as general-purpose I/O, the user must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the register bits will assume their reset states when the I/O pin latches are opened and the I/O pins will switch to their reset states.

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.

## 10.6.3 Stop3 Mode

Upon entering the Stop3 Mode, all of the clocks in the MCU, including the oscillator itself, are halted. The ICG is turned off, the ATD is disabled, and the voltage regulator is put in standby. The states of all of the internal registers and logic, as well as the RAM content, are maintained. The I/O pin states are not latched at the pin as in Stop2. Instead they are maintained by virtue of the states of the internal logic driving the pins being maintained.

Exit from Stop3 is performed by asserting  $\overline{\text{RESET}}$ , an asynchronous interrupt pin, or through the real-time interrupt. The asynchronous interrupt pins are the IRQ or KBI pins.

If Stop3 is exited by means of the  $\overline{\text{RESET}}$  pin, then the MCU will be reset and operation will resume after taking the reset vector. Exit by means of an asynchronous interrupt or the real-time interrupt will result in the MCU taking the appropriate interrupt vector.

A separate self-clocked source ( $\approx 1$  kHz) for the real-time interrupt allows a wake up from Stop2 or Stop3 Mode with no external components. When  $\text{RTIS2:RTIS1:RTIS0} = 0:0:0$ , the real-time interrupt function and this 1-kHz source are disabled. Power consumption is lower when the 1-kHz source is disabled, but in that case the real-time interrupt cannot wake the MCU from stop.

### 10.6.4 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 the [Chapter 11, “MCU Memory”](#). If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters Stop Mode so background debug communication is still possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If the user attempts to enter either Stop1 or 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 the device enters background debug mode, all background commands are available. The table below summarizes the behavior of the MCU in stop when entry into the background debug mode is enabled.

**Table 10-2. BDM Enabled Stop Mode Behavior**

Mode	PDC	PPDC	CPU, Digital Peripherals, FLASH	RAM	ICG	ATD	Regulator	I/O Pins	RTI
Stop3	Don't care	Don't care	Standby	Standby	Active	Disabled <sup>1</sup>	Active	States held	Optionally on

<sup>1</sup> Either ATD Stop Mode or Power-Down Mode depending on the state of ATDPU.

### 10.6.5 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 by setting the LVDE and the LVDSE bits in SPMSC1 when the CPU executes a STOP instruction, then the voltage regulator remains active during Stop Mode. If the user attempts to enter either Stop1 or Stop2 with the LVD enabled for stop ( $\text{LVDSE} = 1$ ), the MCU will instead enter Stop3. The table below summarizes the behavior of the MCU in stop when the LVD is enabled.

**Table 10-3. LVD Enabled Stop Mode Behavior**

Mode	PDC	PPDC	CPU, Digital Peripherals, FLASH	RAM	ICG	ATD	Regulator	I/O Pins	RTI
Stop3	Don't care	Don't care	Standby	Standby	Standby	Disabled <sup>1</sup>	Active	States held	Optionally on

<sup>1</sup> Either ATD Stop Mode or Power-Down Mode depending on the state of ATDPU.

## 10.6.6 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any Stop Mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to [Section 10.6.1, “Stop1 Mode”](#), [Section 10.6.2, “Stop2 Mode”](#), and [Section 10.6.3, “Stop3 Mode”](#), for specific information on system behavior in stop modes.

### I/O Pins

- All I/O pin states remain unchanged when the MCU enters Stop3 Mode.
- If the MCU is configured to go into Stop2 Mode, all I/O pins states are latched before entering stop.
- If the MCU is configured to go into Stop1 Mode, all I/O pins are forced to their default reset state upon entry into stop.

### Memory

- All RAM and register contents are preserved while the MCU is in Stop3 Mode.
- All registers will be reset upon wake-up from Stop2, but the contents of RAM are preserved and pin states remain latched until the PPDACK bit is written. The user may save any memory-mapped register data into RAM before entering Stop2 and restore the data upon exit from Stop2.
- All registers will be reset upon wake-up from Stop1 and the contents of RAM are not preserved. The MCU must be initialized as upon reset. The contents of the FLASH memory are non-volatile and are preserved in any of the stop modes.

**ICG** — In Stop3 Mode, the ICG enters its low-power standby state. Either the oscillator or the internal reference may be kept running when the ICG is in standby by setting the appropriate control bit. In both Stop2 and Stop1 modes, the ICG is turned off. Neither the oscillator nor the internal reference can be kept running in Stop2 or Stop1, even if enabled within the ICG module.

**TPM** — When the MCU enters Stop Mode, the clock to the TPM1 and TPM2 modules stop. The modules halt operation. If the MCU is configured to go into Stop2 or Stop1 Mode, the TPM modules will be reset upon wake-up from stop and must be re initialized.

**ATD** — When the MCU enters Stop Mode, the ATD will enter a low-power standby state. No conversion operation will occur while in stop. If the MCU is configured to go into Stop2 or Stop1 Mode, the ATD will be reset upon wake-up from stop and must be re initialized.



**KBI** — During Stop3, the KBI pins that are enabled continue to function as interrupt sources that are capable of waking the MCU from Stop3. The KBI is disabled in Stop1 and Stop2 and must be re initialized after waking up from either of these modes.

**SCI** — When the MCU enters Stop Mode, the clocks to the SCI1 and SCI2 modules stop. The modules halt operation. If the MCU is configured to go into Stop2 or Stop1 Mode, the SCI modules will be reset upon wake-up from stop and must be re initialized.

**SPI** — When the MCU enters Stop Mode, the clocks to the SPI module stop. The module halts operation. If the MCU is configured to go into Stop2 or Stop1 Mode, the SPI module will be reset upon wake-up from stop and must be re initialized.

**IIC** — When the MCU enters Stop Mode, the clocks to the IIC module stops. The module halts operation. If the MCU is configured to go into Stop2 or Stop1 Mode, the IIC module will be reset upon wake-up from stop and must be re-initialized.

**Voltage Regulator** — The voltage regulator enters a low-power standby state when the MCU enters any of the stop modes unless the LVD is enabled in Stop Mode or BDM is enabled.



# Chapter 11

## MCU Memory

### 11.1 HCS08 Memory Map

As shown in [Figure 11-1](#), on-chip memory in the HCS08 MCUs consists of RAM, FLASH program memory for non-volatile data storage, plus I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (\$0000 through \$007F)
- High-page registers (\$1800 through \$182B)
- Non volatile registers (\$FFB0 through \$FFBF)

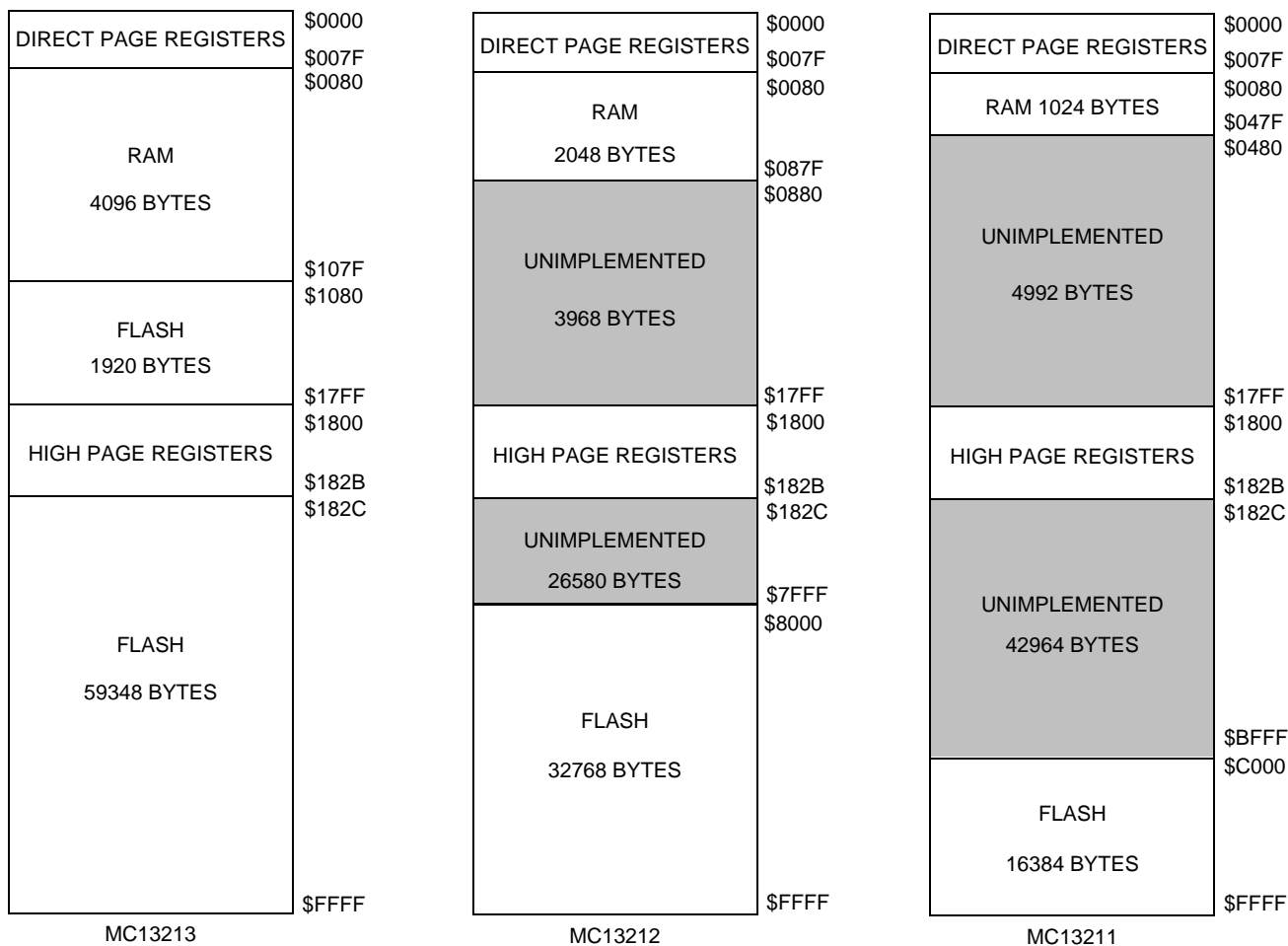


Figure 11-1. MC1321X Memory Maps

Note that the MC13213 has a high ~60k block of flash memory and a second smaller ~2k low block of flash memory. The high block can be fully protected while the low block is left writable for storing parameters and data as non volatile values.

### 11.1.1 Reset and Interrupt Vector Assignments

Table 11-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale-provided equate file for the HCS08. For more details about resets, interrupts, interrupt priority, and local interrupt mask controls, refer to [Chapter 12, “MCU Resets, Interrupts, and System Configuration”](#).

**Table 11-1. Reset and Interrupt Vectors**

Address (High/Low)	Vector	Vector Name
0xFFC0:FFC1 ↑ ↓ 0xFFCA:FFCB	Unused Vector Space (available for user program)	
0xFFCC:FFCD	RTI	Vrti
0xFFCE:FFCF	IIC	Viic1
0xFFD0:FFD1	ATD Conversion	Vatd1
0xFFD2:FFD3	Keyboard	Vkeyboard1
0xFFD4:FFD5	SCI2 Transmit	Vsci2tx
0xFFD6:FFD7	SCI2 Receive	Vsci2rx
0xFFD8:FFD9	SCI2 Error	Vsci2err
0xFFDA:FFDB	SCI1 Transmit	Vsci1tx
0xFFDC:FFDD	SCI1 Receive	Vsci1rx
0xFFDE:FFDF	SCI1 Error	Vsci1err
0xFFE0:FFE1	SPI	Vspi1
0xFFE2:FFE3	TPM2 Overflow	Vtpm2ovf
0xFFE4:FFE5	TPM2 Channel 4	Vtpm2ch4
0xFFE6:FFE7	TPM2 Channel 3	Vtpm2ch3
0xFFE8:FFE9	TPM2 Channel 2	Vtpm2ch2
0xFFEA:FFEB	TPM2 Channel 1	Vtpm2ch1
0xFFEC:FFED	TPM2 Channel 0	Vtpm2ch0
0xFFEE:FFEF	TPM1 Overflow	Vtpm1ovf
0xFFFF0:FFF1	TPM1 Channel 2	Vtpm1ch2
0xFFFF2:FFF3	TPM1 Channel 1	Vtpm1ch1
0xFFFF4:FFF5	TPM1 Channel 0	Vtpm1ch0
0xFFFF6:FFF7	ICG	Vicg
0xFFFF8:FFF9	Low Voltage Detect	Vlvd
0xFFFFA:FFFB	IRQ	Virq

**Table 11-1. Reset and Interrupt Vectors (continued)**

Address (High/Low)	Vector	Vector Name
0xFFFC:FFFD	SWI	Vswi
0xFFFE:FFFF	Reset	Vreset

## 11.2 Register Addresses and Bit Assignments

The registers in the HCS08 are divided into these three groups:

- Direct-page registers are located in the first 128 locations in the memory map, so they are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and variables.
- The non volatile register area consists of a block of 16 locations in FLASH memory at 0xFFB0–0xFFBF.

Non volatile register locations include:

- Three values which are loaded into working registers at reset
- An 8-byte backdoor comparison key which optionally allows a user to gain controlled access to secure memory

Because the non volatile 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 11-2](#) is a summary of all user-accessible direct-page registers and control bits.

The direct page registers in [Table 11-2](#) can use the more efficient direct addressing mode which only requires 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 11-3](#) and [Table 11-4](#) the whole address in column one is shown in bold. In [Table 11-2](#), [Table 11-3](#), and [Table 11-4](#), the register names in column two are shown in bold to set them apart from the bit names to the right. Cells that are not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s.

**Table 11-2. Direct-Page Register Summary (Sheet 1 of 3)**

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	<b>PTAD</b>	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	<b>PTAPE</b>	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x0002	<b>PTASE</b>	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x0003	<b>PTADD</b>	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0004	<b>PTBD</b>	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0005	<b>PTBPE</b>	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x0006	<b>PTBSE</b>	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x0007	<b>PTBDD</b>	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0008	<b>PTCD</b>	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0009	<b>PTCPE</b>	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x000A	<b>PTCSE</b>	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x000B	<b>PTCDD</b>	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x000C	<b>PTDD</b>	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
0x000D	<b>PTDPE</b>	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x000E	<b>PTDSE</b>	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x000F	<b>PTDDD</b>	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
0x0010	<b>PTED</b>	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x0011	<b>PTEPE</b>	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x0012	<b>PTESE</b>	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x0013	<b>PTEDD</b>	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x0014	<b>IRQSC</b>	0	0	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x0015	Reserved	—	—	—	—	—	—	—	—
0x0016	<b>KBI1SC</b>	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBF	KBACK	KBIE	KBIMOD
0x0017	<b>KBI1PE</b>	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x0018	<b>SCI1BDH</b>	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0019	<b>SCI1BDL</b>	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x001A	<b>SCI1C1</b>	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x001B	<b>SCI1C2</b>	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x001C	<b>SCI1S1</b>	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x001D	<b>SCI1S2</b>	0	0	0	0	0	0	0	RAF
0x001E	<b>SCI1C3</b>	R8	T8	TXDIR	0	ORIE	NEIE	FEIE	PEIE
0x001F	<b>SCI1D</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0020	<b>SCI2BDH</b>	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0021	<b>SCI2BDL</b>	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0022	<b>SCI2C1</b>	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0023	<b>SCI2C2</b>	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0024	<b>SCI2S1</b>	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0025	<b>SCI2S2</b>	0	0	0	0	0	0	0	RAF
0x0026	<b>SCI2C3</b>	R8	T8	TXDIR	0	ORIE	NEIE	FEIE	PEIE
0x0027	<b>SCI2D</b>	Bit 7	6	5	4	3	2	1	Bit 0

Table 11-2. Direct-Page Register Summary (Sheet 2 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	<b>SPI1C1</b>	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0029	<b>SPI1C2</b>	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x002A	<b>SPI1BR</b>	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x002B	<b>SPI1S</b>	SPRF	0	SPTEF	MODF	0	0	0	0
0x002C	Reserved	0	0	0	0	0	0	0	0
0x002D	<b>SPI1D</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x002E	Reserved	0	0	0	0	0	0	0	0
0x002F	Reserved	0	0	0	0	0	0	0	0
0x0030	<b>TPM1SC</b>	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0031	<b>TPM1CNTH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0032	<b>TPM1CNTL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0033	<b>TPM1MODH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0034	<b>TPM1MODL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0035	<b>TPM1C0SC</b>	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0036	<b>TPM1C0VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0037	<b>TPM1C0VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0038	<b>TPM1C1SC</b>	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0039	<b>TPM1C1VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x003A	<b>TPM1C1VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x003B	<b>TPM1C2SC</b>	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x003C	<b>TPM1C2VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x003D	<b>TPM1C2VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x003E– 0x003F	Reserved	—	—	—	—	—	—	—	—
0x0040	<b>PTFD</b>	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
0x0041	<b>PTFPE</b>	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
0x0042	<b>PTFSE</b>	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
0x0043	<b>PTFDD</b>	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
0x0044	<b>PTGD</b>	PTGD7	PTGD6	PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
0x0045	<b>PTGPE</b>	PTGPE7	PTGPE6	PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
0x0046	<b>PTGSE</b>	PTGSE7	PTGSE6	PTGSE5	PTGSE4	PTGSE3	PTGSE2	PTGSE1	PTGSE0
0x0047	<b>PTGDD</b>	PTGDD7	PTGDD6	PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
0x0048	<b>ICGC1</b>	HGO	RANGE	REFS	CLKS		OSCSTEN	LOCD	0
0x0049	<b>ICGC2</b>	LOLRE	MFD			LOCRE	RFD		
0x004A	<b>ICGS1</b>	CLKST		REFST	LOLS	LOCK	LOCS	ERCS	ICGIF
0x004B	<b>ICGS2</b>	0	0	0	0	0	0	0	DCOS
0x004C	<b>ICGFLTU</b>	0	0	0	0	FLT			
0x004D	<b>ICGFLTU</b>	FLT							
0x004E	<b>ICGTRM</b>	TRIM							
0x004F	Reserved	0	0	0	0	0	0	0	0

**Table 11-2. Direct-Page Register Summary (Sheet 3 of 3)**

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0050	<b>ATD1C</b>	ATDPU	DJM	RES8	SGN	PRS			
0x0051	<b>ATD1SC</b>	CCF	ATDIE	ATDCO	ATDCH				
0x0052	<b>ATD1RH</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0053	<b>ATD1RL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0054	<b>ATD1PE</b>	ATDPE7	ATDPE6	ATDPE5	ATDPE4	ATDPE3	ATDPE2	ATDPE1	ATDPE0
0x0055– 0x0057	Reserved	—	—	—	—	—	—	—	—
0x0058	<b>IIC1A</b>	ADDR							0
0x0059	<b>IIC1F</b>	MULT		ICR					
0x005A	<b>IIC1C</b>	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x005B	<b>IIC1S</b>	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x005C	<b>IIC1D</b>	DATA							
0x005D– 0x005F	Reserved	—	—	—	—	—	—	—	—
0x0060	<b>TPM2SC</b>	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0061	<b>TPM2CNTH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0062	<b>TPM2CNTL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0063	<b>TPM2MODH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0064	<b>TPM2MODL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0065	<b>TPM2C0SC</b>	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0066	<b>TPM2C0VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0067	<b>TPM2C0VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0068	<b>TPM2C1SC</b>	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0069	<b>TPM2C1VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x006A	<b>TPM2C1VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x006B	<b>TPM2C2SC</b>	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x006C	<b>TPM2C2VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x006D	<b>TPM2C2VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x006E	<b>TPM2C3SC</b>	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x006F	<b>TPM2C3VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0070	<b>TPM2C3VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0071	<b>TPM2C4SC</b>	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	0	0
0x0072	<b>TPM2C4VH</b>	Bit 15	14	13	12	11	10	9	Bit 8
0x0073	<b>TPM2C4VL</b>	Bit 7	6	5	4	3	2	1	Bit 0
0x0074– 0x007F	Reserved	—	—	—	—	—	—	—	—

High-page registers, shown in [Table 11-3](#), are accessed much less often than other I/O and control registers so they have been located outside the direct addressable memory space, starting at 0x1800.



**Table 11-3. High-Page Register Summary**

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	0	ICG	LVD	0
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT	COPE	COPT	STOPE	STOPT—	0	0	BKGDPE	—
0x1803– 0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	REV3	REV2	REV1	REV0	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	SRTISC	RTIF	RTIACK	RTICLKS	RTIE	0	RTIS2	RTIS1	RTIS0
0x1809	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	0
0x180A	SPMSC2	LVWF	LVWACK	LVDV	LVWV	PPDF	PPDACK	PDC	PPDC
0x180B– 0x180F	Reserved	—	—	—	—	—	—	—	—
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBGC	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	DBGT	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	DBGS	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
0x1819– 0x181F	Reserved	—	—	—	—	—	—	—	—
0x1820	FCDIV	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPOPEN	FPDIS	FPS2	FPS1	FPS0	0	0	0
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
0x1827– 0x182B	Reserved	—	—	—	—	—	—	—	—

Non volatile FLASH registers, shown in [Table 11-4](#), are located in the FLASH memory. These registers include an 8-byte backdoor key which optionally can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the non volatile register area of the FLASH memory are transferred into corresponding FPROT and FOPT working registers in the high-page registers to control security and block protection options.

**Table 11-4. Nonvolatile Register Summary**

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFB0	NVBACKKEY	8-Byte Comparison Key							
–									
0xFFB7									
0xFFB8	Reserved	—	—	—	—	—	—	—	—
–		—	—	—	—	—	—	—	—
0xFFBC									
0xFFBD	NVPROT	FPOPEN	FPDIS	FPS2	FPS1	FPS0	0	0	0
0xFFBE	Reserved <sup>1</sup>	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

<sup>1</sup> This location is used to store the factory trim value for the ICG.

Provided the key enable (KEYEN) bit is 1, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory. (A security key cannot be entered directly through background debug commands.) This security key can be disabled completely by programming the KEYEN bit to 0. If the security key is disabled, the only way to disengage security is by mass erasing the FLASH if needed (normally through the background debug interface) and verifying that FLASH is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC01:SEC00) to the unsecured state (1:0).

### 11.3 RAM

The HCS08 includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, Stop2, or Stop3 Mode. At power-on or after wake up from Stop1, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the HCS08, it is usually best to re-initialize 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 the reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale-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 11.5, “Security”](#) for a detailed description of the security feature.

## 11.4 FLASH

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

### 11.4.1 Features

Features of the FLASH memory include:

- FLASH Size
  - MC13211 — 16384 bytes (32 pages of 512 bytes each)
  - MC13212 — 32768 bytes (64 pages of 512 bytes each)
  - MC13213 — a high block of 59348 bytes (115 pages of 512 bytes each plus 1 page of 468 bytes) and a low block of 1920 bytes. The high block can be protected and the low block left unprotected for use as non-volatile parameters.
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

### 11.4.2 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 11.6.1, “FLASH Clock Divider Register \(FCDIV\)”](#)). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ( $1/f_{FCLK}$ ) 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 11-5](#) shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK ( $f_{FCLK}$ ). The time for one cycle of FCLK is  $t_{FCLK} = 1/f_{FCLK}$ . The times are shown as a number of cycles of FCLK and as an absolute time for the case where  $t_{FCLK} = 5 \mu\text{s}$ . Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

**Table 11-5. Program and Erase Times**

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

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

### 11.4.3 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 blocks of FLASH that may be erased. In the 60K version, there are two instances where the size of a block that is accessible to the user is less than 512 bytes: the first page following RAM, and the first page following the high page registers. These pages are overlapped by the RAM and high page registers, respectively.

#### NOTE

Do not program any byte in the FLASH more than once after a successful erase operation. Reprogramming bits in a byte which 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 adhered to, or the command will not be accepted. This minimizes the possibility of any unintended change 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 11-2](#) is a flow chart 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.

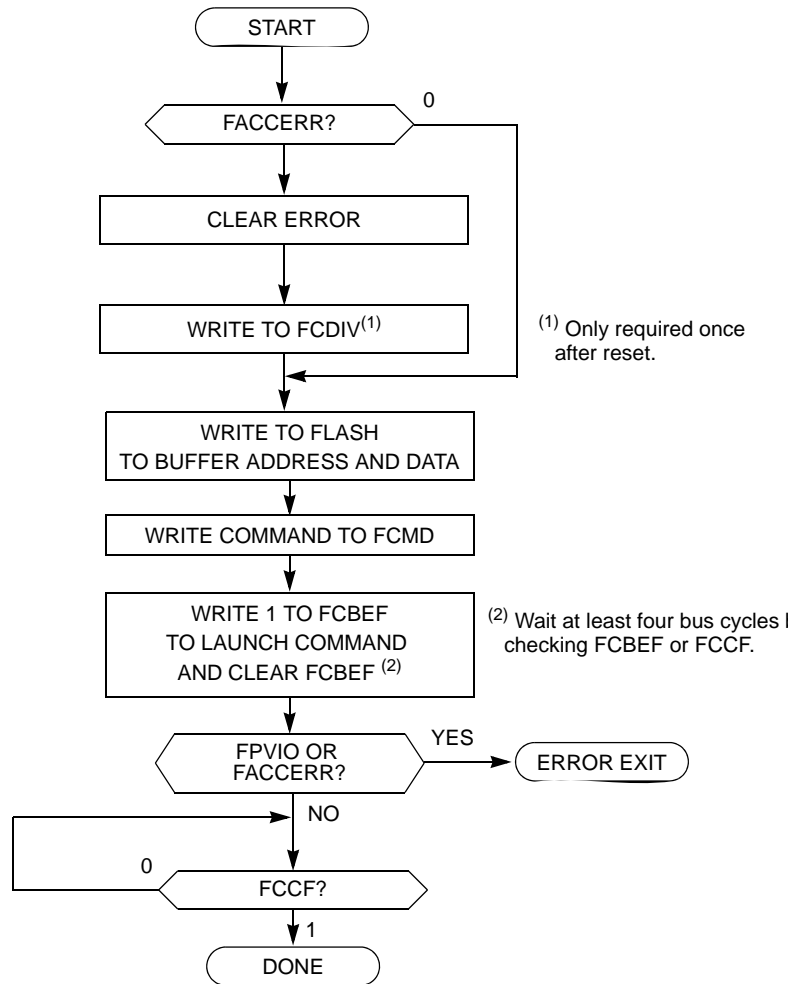


Figure 11-2. FLASH Program and Erase Flow chart

### 11.4.4 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 the following two conditions are met:

1. The next burst program command has been queued before the current program operation has completed.
2. 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.

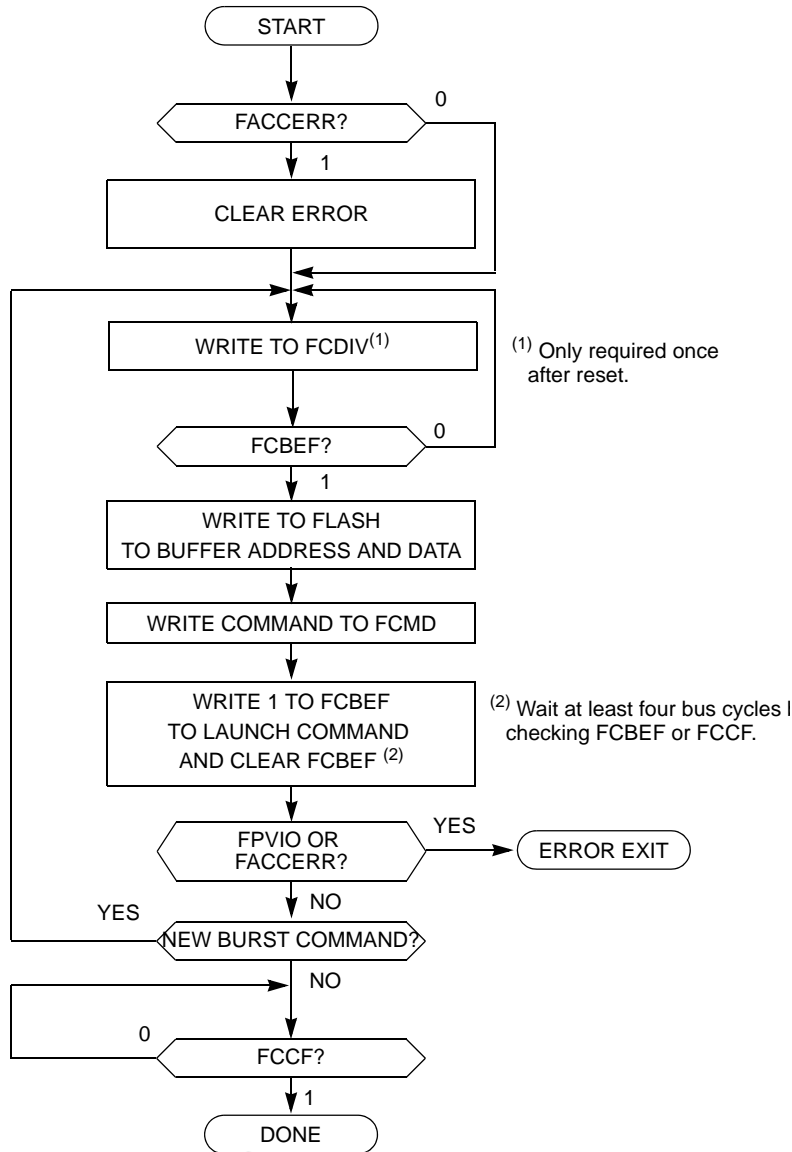


Figure 11-3. FLASH Burst Program Flow chart

### 11.4.5 Access Errors

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

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

- Writing to a FLASH address before the internal FLASH clock frequency has been set by writing to the FCDIV register
- Writing to a FLASH address while FCBEF is not set (A new command cannot be started until the command buffer is empty.)
- Writing a second time to a FLASH address before launching the previous command (There is only one write to FLASH for every command.)
- Writing a second time to FCMD before launching the previous command (There is only one write to FCMD for every command.)
- Writing to any FLASH control register other than FCMD after writing to a FLASH address
- Writing any command code other than the five allowed codes (0x05, 0x20, 0x25, 0x40, or 0x41) to FCMD
- Accessing (read or write) any FLASH control register other than the write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD
- The MCU 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

### 11.4.6 FLASH Block Protection

Block protection prevents program or erase changes for FLASH memory locations in a designated address range. Mass erase is disabled when any block of FLASH is protected. The HCS08 allows a block of memory at the end of FLASH, and/or the entire FLASH memory to be block protected. A disable control bit and a 3-bit control field, allows the user to set the size of this block. A separate control bit allows block protection of the entire FLASH memory array. All seven of these control bits are located in the FPROT register (see [Section 11.6.4, “FLASH Protection Register \(FPROT and NVPROT\)”](#)).

At reset, the high-page register (FPROT) is loaded with the contents of the NVPROT location which is in the non volatile register block of the FLASH memory. The value in FPROT cannot be changed directly from application software so a runaway program cannot alter the block protection settings. If the last 512 bytes of FLASH which includes the NVPROT register is protected, the application program cannot alter the block protection settings (intentionally or unintentionally). The FPROT control bits can be written by background debug commands to allow a way to erase a protected FLASH memory.

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

### 11.4.7 Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the FLASH memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFFD) are redirected, while the reset vector (0xFFFFE:FFFF) is not. When more than 32K is protected, vector redirection must not be enabled.

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

## 11.5 Security

The HCS08 includes circuitry to prevent unauthorized access to the contents of FLASH and RAM memory. When security is engaged, FLASH and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two non volatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the non volatile 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 while the other three combinations engage security. Notice the erased state (1:1) makes the MCU secure. During development, whenever the FLASH is erased, it is good practice to immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

The on-chip debug module cannot be enabled while the MCU is secure. The separate background debug controller can still be used for background memory access commands, but the MCU cannot enter Active Background Mode except by holding BKGD/MS low at the rising edge of reset.

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

1. Writing 1 to KEYACC in the FCNFG register. This makes the FLASH module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a FLASH program or erase command.



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

The security key can be written only from RAM, so it cannot be entered through background commands without the cooperation of a secure user program. The FLASH memory cannot be accessed by read operations while KEYACC is set.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in FLASH memory locations in the non volatile register space so users can program these locations just as they would program any other FLASH memory location. The non volatile 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 performing 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.

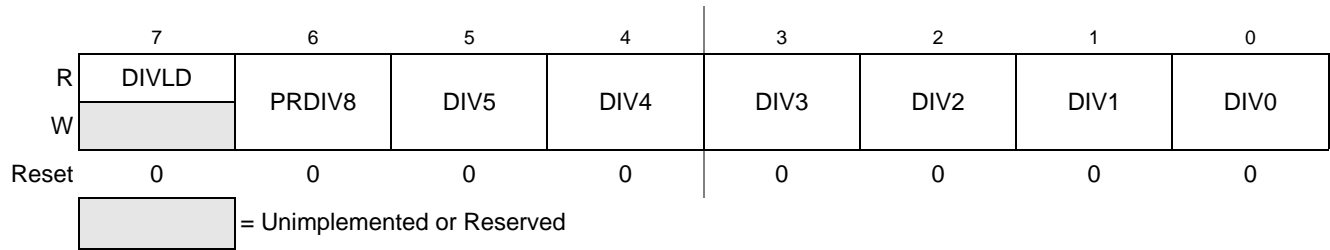
## 11.6 FLASH Registers and Control Bits

The FLASH module has nine 8-bit registers in the high-page register space, three locations in the non volatile register space in FLASH memory that are copied into three corresponding high-page control registers at reset. There is also an 8-byte comparison key in FLASH memory. Refer to [Table 11-3](#) and [Table 11-4](#) for the absolute address assignments for all FLASH registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

### 11.6.1 FLASH Clock Divider Register (FCDIV)

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

Offset



**Figure 11-4. FLASH Clock Divider Register (FCDIV)**

**Table 11-6. FCDIV Field Descriptions**

Field	Description
7 DIVLD	<p><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.</p> <p>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.</p>
6 PRDIV8	<p><b>Prescale (Divide) FLASH Clock by 8</b></p> <p>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.</p>
5 DIV[5:0]	<p><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 DIV5:DIV0 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 11-1</a> and <a href="#">Equation 11-2</a>. <a href="#">Table 11-7</a> shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.</p>

if PRDIV8 = 0 —  $f_{FCLK} = f_{Bus} \div ([DIV5:DIV0] + 1)$  *Eqn. 11-1*

if PRDIV8 = 1 —  $f_{FCLK} = f_{Bus} \div (8 \times ([DIV5:DIV0] + 1))$  *Eqn. 11-2*

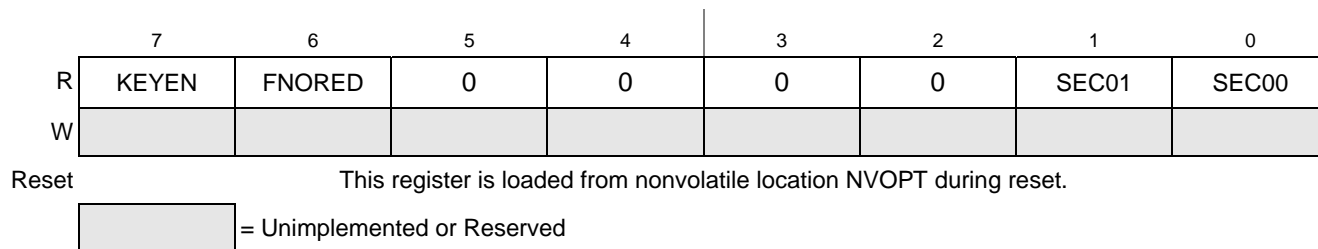
**Table 11-7. FLASH Clock Divider Settings**

$f_{Bus}$	PRDIV8 (Binary)	DIV5:DIV0 (Decimal)	$f_{FCLK}$	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200 kHz	5 μs
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

## 11.6.2 FLASH Options Register (FOPT and NVOPT)

During reset, the contents of the non volatile location NVOPT are copied from FLASH into FOPT. Bits 5 through 2 are not used and always read 0. This register may be read at any time, but writes have no meaning or effect. To change the value in this register, erase and reprogram the NVOPT location in FLASH memory as usual and then issue a new MCU reset.

Offset



**Figure 11-5. FLASH Options Register (FOPT)**

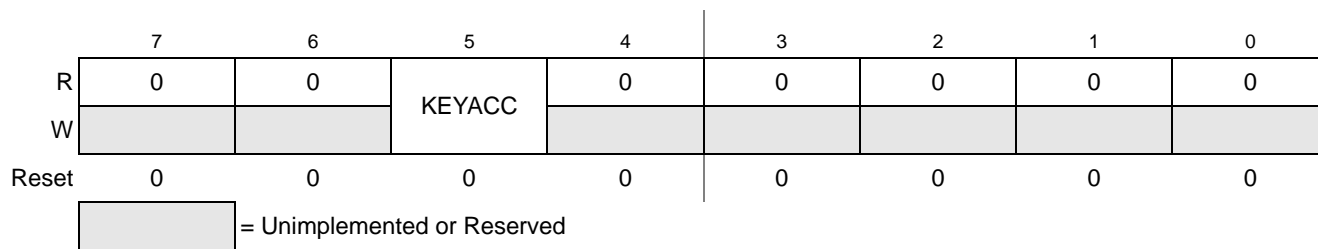
**Table 11-8. FOPT Field Descriptions**

Field	Description
7 KEYEN	<p><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 11.5, “Security”</a>.</p> <p>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.</p>
6 FNORED	<p><b>Vector Redirection Disable</b> — When this bit is 1, vector redirection is disabled.</p> <p>0 Vector redirection enabled.                      1 Vector redirection disabled.</p>
1:0 SEC0[1:0]	<p><b>Security State Code</b> — This 2-bit field determines the security state of the MCU as shown below. 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. For more detailed information about security, refer to <a href="#">Section 11.5, “Security”</a>.</p> <p>00 Secure                      01 Secure                      10 Unsecured                      11 Secure</p> <p>SEC0[1:0] changes to 10 after successful backdoor key entry or a successful blank check of FLASH.</p>

### 11.6.3 FLASH Configuration Register (FCNFG)

Bits 7 through 5 may be read or written at any time. Bits 4 through 0 always read 0 and cannot be written.

Offset



**Figure 11-6. FLASH Configuration Register (FCNFG)**

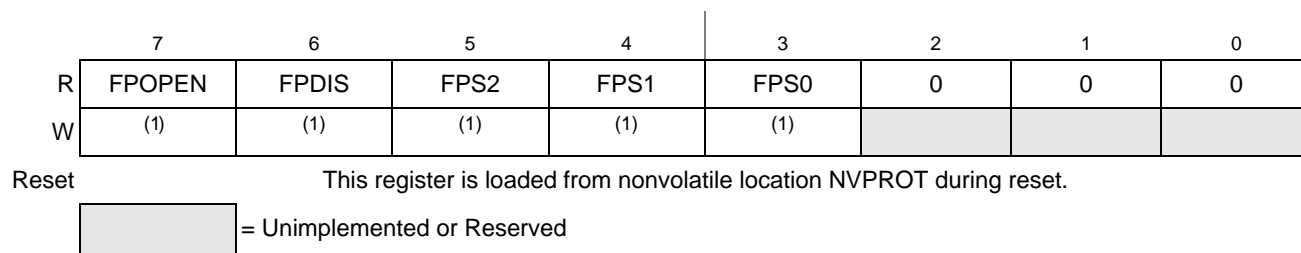
**Table 11-9. FCNFG Field Descriptions**

Field	Description
5 KEYACC	<p><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 11.5, “Security”</a>.</p> <p>0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a FLASH programming or erase command.</p> <p>1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.</p> <p>Reads of the FLASH return invalid data.</p>

### 11.6.4 FLASH Protection Register (FPROT and NVPROT)

During reset, the contents of the non volatile location NVPROT is copied from FLASH into FPROT. Bits 0, 1, and 2 are not used and each always reads as 0. This register may be read at any time, but user program writes have no meaning or effect. Background debug commands can write to FPROT.

Offset



**Figure 11-7. FLASH Protection Register (FPROT)**

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

**Table 11-10. FPROT Field Descriptions**

Field	Description
7 FPOPEN	<b>Open Unprotected FLASH for Program/Erase</b> 0 Entire FLASH memory is block protected (no program or erase allowed). 1 Any FLASH location, not otherwise block protected or secured, may be erased or programmed.
6 FPDIS	<b>FLASH Protection Disable</b> 0 FLASH block specified by FPS2:FPS0 is block protected (program and erase not allowed). 1 No FLASH block is protected.
5:3 FPS[2:0]	<b>FLASH Protect Size Selects</b> — When FPDIS = 0, this 3-bit field determines the size of a protected block of FLASH locations at the high address end of the FLASH (see Table 11-11). Protected FLASH locations cannot be erased or programmed.

**Table 11-11. High Address Protected Block**

FPS2:FPS1:FPS0	Protected Address Range	Protected Block Size	Redirected Vectors <sup>1,2</sup>
0:0:0	0xFE00–0xFFFF	512 bytes	0xFDC0–0xFDFD
0:0:1	0xFC00–0xFFFF	1024 bytes	0xFBC0–0xFBFD
0:1:0	0xF800–0xFFFF	2048 bytes	0xF7C0–0xF7FD
0:1:1	0xF000–0xFFFF	4096 bytes	0xEFC0–0xEFFD
1:0:0	0xE000–0xFFFF	8192 bytes	0xDFC0–0xDFFD
1:0:1	0xC000–0xFFFF	16384 bytes	0xBFC0–0xBFFD <sup>3</sup>
1:1:0	0x8000–0xFFFF	32768 bytes	0x7FC0–0x7FFD <sup>4</sup>
1:1:1	0x182C–0xFFFF	59,348 bytes	No redirection allowed

<sup>1</sup> No redirection if FPOPEN = 0, or FNORED = 1.

<sup>2</sup> Reset vector is not redirected.

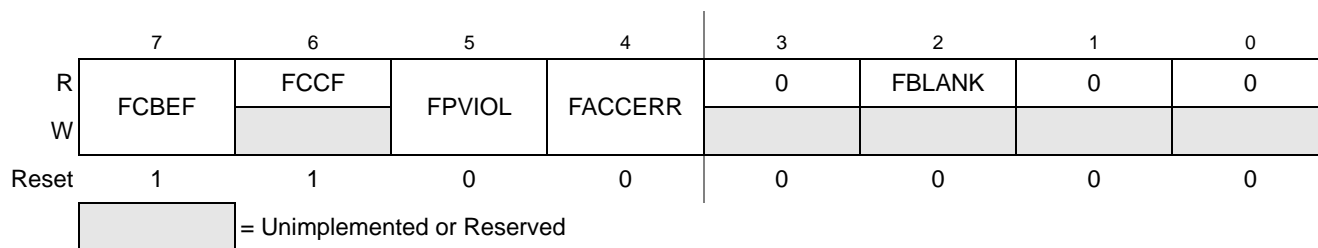
<sup>3</sup> 32K and 60K devices only.

<sup>4</sup> 60K devices only. Note that the low flash block of 1920 bytes at 0x1080–0x17FF is not protected.

### 11.6.5 FLASH Status Register (FSTAT)

Bits 3, 1, and 0 always read 0 and writes have no meaning or effect. The remaining five bits are status bits that can be read at any time. Writes to these bits have special meanings that are covered in the bit descriptions.

Offset


**Figure 11-8. FLASH Status Register (FSTAT)**

**Table 11-12. FSTAT 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 may 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 FCBEF is cleared to register a command that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.
4 FACCERR	<b>Access Error Flag</b> — FACCERR is set automatically when the proper command sequence is not followed 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 11.4.5, “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 has occurred. 1 An access error has occurred.
2 FBLANK	<b>FLASH Verified as All Blank (Erased) Flag</b> — FBLANK is set automatically at the conclusion of a blank check command if the entire FLASH array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the FLASH array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the FLASH array is completely erased (all 0xFF).

### 11.6.6 FLASH Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in [Table 11-14](#). Refer to [Section 11.4.3, “Program and Erase Command Execution”](#) for a detailed discussion of FLASH programming and erase operations.

Offset

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
Reset	0	0	0	0	0	0	0	0

**Figure 11-9. FLASH Command Register (FCMD)**

**Table 11-13. FCMD Field Descriptions**

Field	Description
7:0 FCMD[7:0]	See <a href="#">Table 11-14</a> for a description of FCMD[7:0].

**Table 11-14. FLASH Commands**

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

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

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





# Chapter 12

## MCU Resets, Interrupts, and System Configuration

### 12.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the HCS08. Some interrupt sources from peripheral modules are covered in greater detail within other sections of this data manual. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog and real-time interrupt (RTI), are not part of on-chip peripheral systems with their own sections but are part of the system control logic.

### 12.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation:
  - Power-on detection (POR)
  - Low voltage detection (LVD) with enable
  - External  $\overline{\text{RESET}}$  pin with enable
  - COP watchdog with enable and two timeout choices
  - Illegal opcode
- Illegal address access
  - Serial command from a background debug host
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see [Table 12-1](#))
- Software interrupt instruction (SWI)
- IRQ pin with enable, choice of polarity, level, and/or edge
- Low-voltage detect interrupt with enable
- Three timer interrupts; overflow and two channels from TPM
- Analog comparator interrupt with enable
- Carrier modulator timer interrupt with enable
- Two interrupts from keyboard interrupt with enable
- Real time interrupt with enable, available in stop2 and stop3 with multiple rates based on a separate 1-kHz self-clocked source

## 12.3 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 (\$FFFE:\$FFFF). 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 the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to \$00FF at reset.

The HCS08 has seven sources for reset:

- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect
- Background debug forced reset
- The reset pin ( $\overline{\text{RESET}}$ )
- Clock generator loss of lock and loss of clock reset

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register. Whenever the MCU enters reset, the internal clock generator (ICG) module switches to Self-clocked mode with the frequency of  $f_{\text{Self\_reset}}$  selected. The reset pin is driven low for 34 internal bus cycles where the internal bus frequency is half the ICG frequency. After the 34 cycles are completed, the pin is released and will be pulled up by the internal pullup resistor, unless it is held low externally. After the pin is released, it is sampled after another 38 cycles to determine whether the reset pin is the cause of the MCU reset.

## 12.4 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 timer periodically. If the application program gets lost and fails to reset the COP before it times out, a system reset is generated to force the system back to a known starting point. The COP watchdog is enabled by the COPE bit in SOPT (see [Section 12.7.4, “System Options Register \(SOPT\)”](#) for additional information). The COP timer 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 timer.

After any reset, the COP timer is enabled. This provides a reliable way to detect code that is not executing as intended. If the COP watchdog is not used in an application, it can be disabled by clearing the COPE bit in the write-once SOPT register. Also, the COPT bit can be used to choose one of two timeout periods ( $2^{18}$  or  $2^{13}$  cycles of the bus rate clock). Even if the application will use the reset default settings in COPE and COPT, the user should still write to write-once SOPT during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost.

The write to SRS that services (clears) the COP timer should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

When the MCU is in Active Background Mode, the COP timer is temporarily disabled.

## 12.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing resumes where it 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. The debug module can also generate an SWI under certain circumstances.

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

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence follows 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 may 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 off the stack.

### NOTE

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

When two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 12-1).

### 12.5.1 Interrupt Stack Frame

Figure 12-1 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 which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

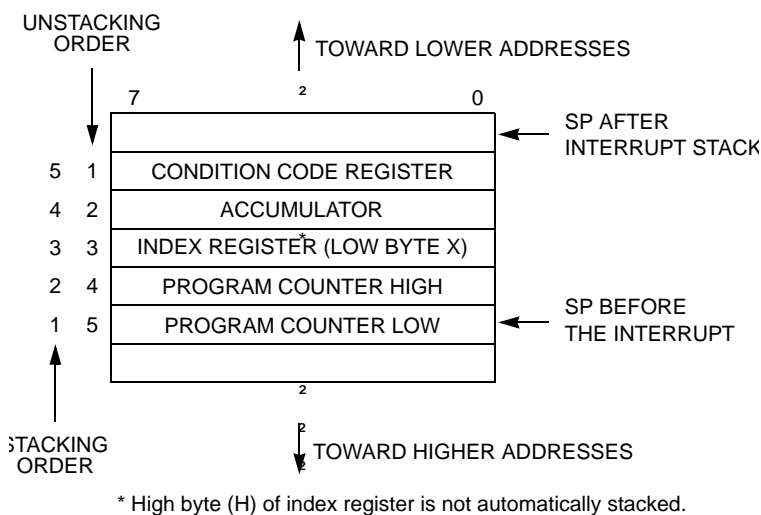


Figure 12-1. Interrupt Stack Frame

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 just recovered from the stack.

The status flag causing the interrupt must be acknowledged (cleared) before returning from the ISR. Typically, the flag should be 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.

### 12.5.2 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQSC status and control register. 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 (if enabled) can wake the MCU.

### 12.5.2.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in the IRQSC register must be 1 for the IRQ pin to act as the interrupt request (IRQ) input. When the pin is configured 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).

When the IRQ pin is configured to detect rising edges, an optional pull-down resistor is available rather than a pullup resistor. 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.

#### NOTE

The voltage measured on the pulled up IRQ pin may be as low as  $V_{DD} - 0.7$  V. The internal gates connected to this pin are pulled all the way to  $V_{DD}$ . All other pins with enabled pullup resistors will have an unloaded measurement of  $V_{DD}$ .

**Table 12-1. Vector Summary**

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description	
Lower	26 through 31	\$FFC0/FFC1 through \$FFCA/FFCB	Unused Vector Space (available for user program)					
	25	\$FFCC/FFCD	Vrti	System control	RTIF	RTIE	Real-time interrupt	
	24	\$FFCE/FFCF	Viiic1	IIC	IICIS	IICIE	IIC control	
	23	\$FFD0/FFD1	Vatd1	ATD	COCO	AIEN	AD conversion complete	
	22	\$FFD2/FFD3	Vkeyboard1	KBI	KBF	KBIE	Keyboard pins	
	21	\$FFD4/FFD5	Vsci2tx	SCI2	TDRE TC	TIE TCIE	SCI2 transmit	
	20	\$FFD6/FFD7	Vsci2rx	SCI2	IDLE RDRF	ILIE RIE	SCI2 receive	
	19	\$FFD8/FFD9	Vsci2err	SCI2	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI2 error	
	18	\$FFDA/FFDB	Vsci1tx	SCI1	TDRE TC	TIE TCIE	SCI1 transmit	
	17	\$FFDC/FFDD	Vsci1rx	SCI1	IDLE RDRF	ILIE RIE	SCI1 receive	
	16	\$FFDE/FFDF	Vsci1err	SCI1	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI1 error	

**Table 12-1. Vector Summary (continued)**

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
	15	\$FFE0/FFE1	Vspi1	SPI	SPIF MODF SPTEF	SPIE SPIE SPTIE	SPI
	14	\$FFE2/FFE3	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
	13	\$FFE4/FFE5	Vtpm2ch4	TPM2	CH4F	CH4IE	TPM2 channel 4
	12	\$FFE6/FFE7	Vtpm2ch3	TPM2	CH3F	CH3IE	TPM2 channel 3
	11	\$FFE8/FFE9	Vtpm2ch2	TPM2	CH2F	CH2IE	TPM2 channel 2
	10	\$FFEA/FFEB	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
	9	\$FFEC/FFED	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
	8	\$FFEE/FFEF	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
	7	\$FFF0/FFF1	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2
	6	\$FFF2/FFF3	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
	5	\$FFF4/FFF5	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
	4	\$FFF6/FFF7	Vicg	ICG	ICGIF (LOLS/LOCS)	LOLRE/LOCRE	ICG
	3	\$FFF8/FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect
	2	\$FFFA/FFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	\$FFFC/FFFD	Vswi	Core	SWI Instruction	—	Software interrupt
Higher	0	\$FFFE/FFFF	Vreset	System control	COP LVD RESET pin Illegal opcode	COPE LVDRE — —	Watchdog timer Low-voltage detect External pin Illegal opcode

## 12.6 Low-Voltage Detect (LVD) System

The HCS08 includes a system to protect against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system comprises 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 SPMSC2. The LVD is disabled upon entering any of the stop modes unless the LVDSE bit is set. If LVDSE and LVDE are both set, then the MCU cannot enter Stop1 or Stop2, and the current consumption in Stop3 with the LVD enabled will be greater.

### 12.6.1 Power-On Reset Operation

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

### 12.6.2 LVD Reset Operation

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

### 12.6.3 LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF will be set and an LVD interrupt will occur.

### 12.6.4 Low-Voltage Warning (LVW)

The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching, but is still above, the LVD voltage. The LVW does not have an interrupt associated with it. There are two user selectable trip voltages for the LVW, one high ( $V_{LVWH}$ ) and one low ( $V_{LVWL}$ ). The trip voltage is selected by LVWV in SPMSC2.

## 12.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts based on a multiple of the source clock's period. The RTI has two source clock choices, the external clock input (ICGERCLK) to the ICG or the RTI's own internal clock. The RTI can be used in run, wait, Stop2 and Stop3 modes. It is not available in Stop1 mode.

In run and wait modes, only the external clock can be used as the RTI's clock source. In Stop2 Mode, only the internal RTI clock can be used. In Stop3, either the external clock or internal RTI clock can be used. When using the external oscillator in Stop3 Mode, it must be enabled in stop (OSCSTEN = 1) and configured for low bandwidth operation (RANGE = 0).

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS2:RTIS1:RTIS0) used to select one of seven RTI periods. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The module can be disabled by writing 0:0:0 to RTIS2:RTIS1:RTIS0 in which case the clock source input is disabled and no interrupts will be generated. See [Section 12.7.6, “System Real-Time Interrupt Status and Control Register \(SRTISC\)”](#), for detailed information about this register.

Reset, Interrupt, and System Control Registers and Control Bits

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

Refer to the direct-page register summary in [Chapter 11, “MCU Memory”](#), for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are covered in greater detail in [Chapter 10, “MCU Modes of Operation”](#).

### 12.7.1 Interrupt Pin Request Status and Control Register (IRQSC)

This direct page register includes two unimplemented bits which always read 0, four read/write bits, one read-only status bit, and one write-only bit. These bits are used to configure the IRQ function, report status, and acknowledge IRQ events.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
Write:						IRQACK		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

**Figure 12-2. Interrupt Request Status and Control Register (IRQSC)**

#### IRQEDG — Interrupt Request (IRQ) Edge Select

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 the IRQ pin is enabled as the IRQ input and is configured to detect rising edges, the optional pullup resistor is re-configured as an optional pull-down resistor.

- 1 = IRQ is rising edge or rising edge/high-level sensitive.
- 0 = IRQ is falling edge or falling edge/low-level sensitive.

#### IRQPE — IRQ Pin Enable

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. Also, when this bit is set, either an internal pull-up or an internal pull-down resistor is enabled depending on the state of the IRQMOD bit.

- 1 = IRQ pin function is enabled.
- 0 = IRQ pin function is disabled.

#### IRQF — IRQ Flag

This read-only status bit indicates when an interrupt request event has occurred.

- 1 = IRQ event detected.
- 0 = No IRQ request.



### IRQACK — IRQ Acknowledge

This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.

### IRQIE — IRQ Interrupt Enable

This read/write control bit determines whether IRQ events generate a hardware interrupt request.

- 1 = Hardware interrupt requested whenever IRQF = 1.
- 0 = Hardware interrupt requests from IRQF disabled (use polling).

### IRQMOD — IRQ Detection Mode

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 [Section 8.1, “Modem Interrupts”](#) for more details.

- 1 = IRQ event on falling edges and low levels or on rising edges and high levels.
- 0 = IRQ event on falling edges or rising edges only.

## 12.7.2 System Reset Status Register (SRS)

This register includes six read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by 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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POR	PIN	COP	ILOP	0	ICG	LVD	0
Write:	Writing any value to SIMRS address clears COP watchdog timer.							
Power-on reset:	1	0	0	0	0	0	1	0
Low-voltage reset:	U	0	0	0	0	0	1	0
Any other reset:	0	1	(1)	(1)	0	(1)	0	0

U = Unaffected by reset

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

**Figure 12-3. System Reset Status (SRS)**

### POR — Power-On Reset

Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold.

- 1 = POR caused reset.
- 0 = Reset not caused by POR.

### PIN — External Reset Pin

Reset was caused by an active-low level on the external reset pin.

- 1 = Reset came from external reset pin.
- 0 = Reset not caused by external reset pin.

### COP — Computer Operating Properly (COP) Watchdog

Reset was caused by the COP watchdog timer timing out. This reset source may be blocked by COPE = 0.

- 1 = Reset caused by COP timeout.
- 0 = Reset not caused by COP timeout.

### ILOP — Illegal Opcode

Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if Active Background Mode is disabled by ENBDM = 0 in the BDCSC register.

- 1 = Reset caused by an illegal opcode.
- 0 = Reset not caused by an illegal opcode.

### ICG — ILAD — Illegal Address Access

Reset was caused by an attempt to access a designated illegal address.

- 1 = Reset caused by an illegal address access
- 0 = Reset not caused by an illegal address access

Illegal address areas in the 9S08GB60 are:

- \$0440–\$17FF — Gap from end of RAM to start of high-page registers
- \$1834–\$BFFF — Gap from end of high-page registers to start of FLASH memory

Unused and reserved locations in register areas are not considered designated illegal addresses and do not trigger illegal address resets.

### Internal Clock Generation Module Reset

Reset was caused by an ICG module reset.

- 1 = Reset caused by ICG module.
- 0 = Reset not caused by ICG module.

### LVD — Low Voltage Detect

If the LVD reset is enabled (LVDE = LVDRE = 1) and the supply drops below the LVD trip voltage, an LVD reset occurs. The LVD function is disabled when the MCU enters stop. To maintain LVD operation in stop, the LVDSE bit must be set.

- 1 = Reset caused by LVD trip or POR.
- 0 = Reset not caused by LVD trip or POR.

### 12.7.3 System Background Debug Force Reset Register (SBD FR)

This register contains a single write-only control bit. A serial background 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 \$00.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	0
Write:								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 12-4. System Background Debug Force Reset Register (SBD FR)**

#### BDFR — Background Debug Force Reset

A serial 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.

### 12.7.4 System Options Register (SOPT)

This register may be read at any time. Bits 3 and 2 are unimplemented and always read 0. This is a write-once register so only the first write after reset is honored. Any subsequent attempt to write to SOPT (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT should be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	COPE	COPT	STOPE		0	0	BKGDPE	
Write:								
Reset:	1	1	0	1	0	0	1	1

= Unimplemented or Reserved

**Figure 12-5. System Integration Module Options Register (SOPT)**

#### COPE — COP Watchdog Enable

This write-once bit defaults to 1 after reset.

1 = COP watchdog timer enabled (force reset on timeout).

0 = COP watchdog timer disabled.

**COPT — COP Watchdog Timeout**

This write-once bit defaults to 1 after reset.

- 1 = Long timeout period selected ( $2^{18}$  cycles of BUSCLK).
- 0 = Short timeout period selected ( $2^{13}$  cycles of BUSCLK).

**STOPE — Stop Mode Enable**

This write-once bit defaults to 0 after reset, which disables Stop Mode. If Stop Mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced.

- 1 = Stop Mode enabled.
- 0 = Stop Mode disabled.

**STOPT — Stop Recovery Time Select**

This write-once bit is not used and ignored in the HCS08 because this MCU always defaults to the self-clocked VCO clock source on recovery from stop mode. Since there is little start-up time for the self-clocked clock source, stop recovery is always fast in the HCS08.

- 1 = Long stop recovery to allow crystal oscillator start-up (does not apply in HCS08).
- 0 = Short stop recovery (assumes oscillator is running as MCU recovers from stop).

**BKGDPE — Background Debug Mode Pin Enable**

The BKGDPE bit enables the PTG0/BKGD/MS pin to function as BKGD/MS. When the bit is clear, the pin will function as PTG0, which is an output-only general-purpose I/O. This pin always defaults to BKGD/MS function after any reset.

- 1 = BKGD pin enabled.
- 0 = BKGD pin disabled.

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

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	REV3	REV2	REV1	REV0	ID11	ID10	ID9	ID8
Reset:	0 <sup>1</sup>	0 <sup>(1)</sup>	0 <sup>(1)</sup>	0 <sup>(1)</sup>	0	0	0	0
Read:	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
Reset:	0	0	0	0	0	0	1	0

= Unimplemented or Reserved

<sup>1</sup> The revision number that is hard coded into these bits reflects the current silicon revision level.

**Figure 12-6. System Device Identification Register (SDIDH, SDIDL)**

### REV[3:0] — Revision Number

The high-order 4 bits of address \$1806 are hard coded to reflect the current mask set revision number (0–F).


### ID[11:0] — Part Identification Number

Each derivative in the HCS08 Family has a unique identification number. The HCS08 is hard coded to the value \$002.

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

This register contains one read-only status flag, one write-only acknowledge bit, three read/write delay selects, and three unimplemented bits, which always read 0.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RTIF	0	RTICLKS	RTIE	0	RTIS2	RTIS1	RTIS0
Write:		RTIACK						
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 12-7. System RTI Status and Control Register (SRTISC)**

### RTIF — Real-Time Interrupt Flag

This read-only status bit indicates the periodic wakeup timer has timed out.

- 1 = Periodic wakeup timer timed out.
- 0 = Periodic wakeup timer not timed out.

### RTIACK — Real-Time Interrupt Acknowledge

This write-only bit is used to acknowledge real-time interrupt request (write 1 to clear RTIF). Writing 0 has no meaning or effect. Reads always return 0.

### RTICLKS — Real-Time Interrupt Clock Select

This read/write bit selects the clock source for the real-time interrupt.

- 1 = Real-time interrupt request clock source is external clock.
- 0 = Real-time interrupt request clock source is internal oscillator.

### RTIE — Real-Time Interrupt Enable

This read-write bit enables real-time interrupts.

- 1 = Real-time interrupts enabled.
- 0 = Real-time interrupts disabled.

### RTIS2:RTIS1:RTIS0 — Real-Time Interrupt Period Selects

These read/write bits select the wakeup period for the RTI. The clock source for the real-time interrupt is its own clock source, which oscillates with a period of approximately  $1/f_{ext}$ , and it is independent of other MCU clock sources. Using an external clock source, the delays will be crystal frequency divided by value in RTIS2:RTIS1:RTIS0.

**Table 12-2. Real-Time Interrupt Period**

RTIS2:RTIS1:RTIS0	Internal Clock Source ( $t_{RTI} = 1 \text{ ms, Nominal}$ )	External Clock Source <sup>1</sup> Period = $t_{ext}$
0:0:0	Disable periodic wakeup timer	Disable periodic wakeup timer
0:0:1	8 ms	$t_{ext} \times 256$
0:1:0	32 ms	$t_{ex} \times 1024$
0:1:1	64 ms	$t_{ex} \times 2048$
1:0:0	128 ms	$t_{ex} \times 4096$
1:0:1	256 ms	$t_{ext} \times 8192$
1:1:0	512 ms	$t_{ext} \times 16384$
1:1:1	1.024 s	$t_{ex} \times 32768$

<sup>1</sup>  $t_{ext}$  is based on the external clock source, resonator, or crystal selected by the ICG configuration.

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVDF	0	LVDIE	LVDRE <sup>1</sup>	LVDSE <sup>(1)</sup>	LVDE <sup>(1)</sup>	0	0
Write:		LVDACK						
Reset:	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

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

**Figure 12-8. System Power Management Status and Control 1 Register (SPMSC1)**

**LVDF** — Low-Voltage Detect Flag

Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event.

**LVDACK** — Low-Voltage Detect Acknowledge

This write-only bit is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return 0.

**LVDIE** — Low-Voltage Detect Interrupt Enable

This read/write bit enables hardware interrupt requests for LVDF.

- 1 = Request a hardware interrupt when LVDF = 1.
- 0 = Hardware interrupt disabled (use polling).

**LVDRE** — Low-Voltage Detect Reset Enable

This read/write bit enables LVDF events to generate a hardware reset (provided LVDE = 1).

- 1 = Force an MCU reset when LVDF = 1.
- 0 = LVDF does not generate hardware resets.

**LVDSE** — Low-Voltage Detect Stop Enable

Provided LVDE = 1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in Stop Mode.

- 1 = Low-voltage detect enabled during Stop Mode.
- 0 = Low-voltage detect disabled during Stop Mode.

**LVDE** — Low-Voltage Detect Enable

This read/write bit enables low-voltage detect logic and qualifies the operation of other bits in this register.

- 1 = LVD logic enabled.
- 0 = LVD logic disabled.

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

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVWF	0	LVDV	LVWV	PPDF	0	PDC	PPDC
Write:		LVWACK				PPDACK		
Power-on reset:	0 <sup>1</sup>	0	0	0	0	0	0	0
LVD reset:	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  $V_{LWV}$ .

**Figure 12-9. System Power Management Status and Control 2 Register (SPMSC2)**

**LVWF** — Low-Voltage Warning Flag

The LVWF bit indicates the low-voltage warning status.

1 = Low voltage warning is present or was present.

0 = Low voltage warning **not** present.

LVWACK — Low-Voltage Warning Acknowledge

The LVWACK bit indicates the low-voltage warning acknowledge.

Writing a 1 to LVWACK clears LVWF to 0 if a low voltage warning is not present.

LVDV — Low-Voltage Detect Voltage Select

The LVDV bit selects the LVD trip point voltage ( $V_{LVD}$ ).

1 = High trip point selected ( $V_{LVD} = V_{LVDH}$ ).

0 = Low trip point selected ( $V_{LVD} = V_{LVDL}$ ).

LVWV — Low-Voltage Warning Voltage Select

The LVWV bit selects the LVW trip point voltage ( $V_{LVW}$ ).

1 = High trip point selected ( $V_{LVW} = V_{LVWH}$ ).

0 = Low trip point selected ( $V_{LVW} = V_{LVWL}$ ).

PPDF — Partial Power Down Flag

The PPDF bit indicates that the MCU has exited the Stop2 Mode.

1 = Stop2 Mode recovery.

0 = Not Stop2 Mode recovery.

PPDACK — Partial Power Down Acknowledge

Writing a 1 to PPDACK clears the PPDF bit.

PDC — Power Down Control

The write-once PDC bit controls entry into the power down (Stop2 and Stop1) modes.

1 = Power down modes are enabled.

0 = Power down modes are disabled.

PPDC — Partial Power Down Control

The write-once PPDC bit controls which power down mode, Stop1 or Stop2, is selected.

1 = Stop2, partial power down, mode enabled if PDC set.

0 = Stop1, full power down, mode enabled if PDC set.



## Chapter 13

# MCU Parallel Input/Output

### 13.1 Introduction

This section explains software controls related to parallel input/output (I/O). The MC9S08GBxxA device on the MC1321x has seven I/O ports which include a total of 56 general-purpose I/O pins (one of these pins is output only). See [Chapter 2, “MC1321x Pins and Connections”](#), for more information about the logic and hardware aspects of these pins. Also, the MC1321x does not pin-out all of these I/O.

Many of these pins are shared with on-chip peripherals such as timer systems, external interrupts, or keyboard interrupts. When these other modules are not controlling the port pins, they revert to general-purpose I/O control. For each I/O pin, a port data bit provides access to input (read) and output (write) data, a data direction bit controls the direction of the pin, and a pullup enable bit enables an internal pullup device (provided the pin is configured as an input), and a slew rate control bit controls the rise and fall times of the pins.

#### NOTE

Not all general-purpose I/O pins are available on the MC1321x. To avoid extra current drain from floating input pins, the user's reset initialization routine in the application program should either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

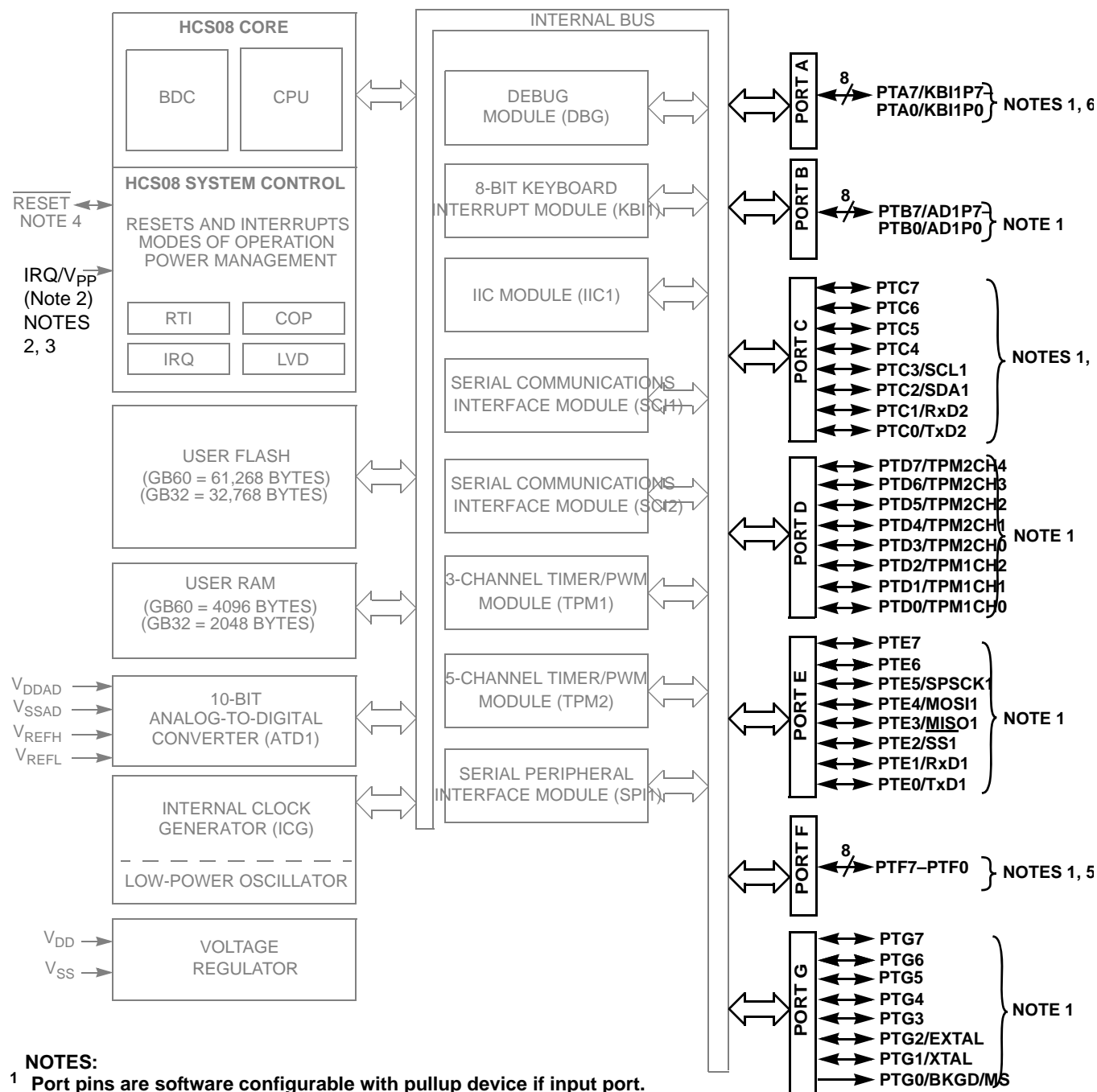


Figure 13-1. Block Diagram Highlighting Parallel Input/Output Pins

## 13.2 Features

Parallel I/O features, depending on package choice, include:

- A total of 56 general-purpose I/O pins in seven ports (PTG0 is output only)
- High-current drivers on port C and port F pins
- Hysteresis input buffers
- Software-controlled pullups on each input pin
- Software-controlled slew rate output buffers
- Eight port A pins shared with KBI1
- Eight port B pins shared with ATD1
- Eight high-current port C pins shared with SCI2 and IIC1
- Eight port D pins shared with TPM1 and TPM2
- Eight port E pins shared with SCI1 and SPI1
- Eight high-current port F pins
- Eight port G pins shared with EXTAL, XTAL, and BKGD/MS

## 13.3 Pin Descriptions

The HCS08 has a total of 56 parallel I/O pins (one is output only) in seven 8-bit ports (PTA–PTG). Not all pins are bonded out in all packages. Consult the pin assignment in [Chapter 2, “MC1321x Pins and Connections”](#), for available parallel I/O pins. All of these pins are available for general-purpose I/O when they are not used by other on-chip peripheral systems.

After reset, BKGD/MS is enabled and therefore is not usable as an output pin until BKGDPE in SOPT is cleared. The rest of the peripheral functions are disabled. After reset, all data direction and pullup enable controls are set to 0s. These pins default to being high-impedance inputs with on-chip pullup devices disabled.

The following paragraphs discuss each port and the software controls that determine each pin’s use.

### 13.3.1 Port A and Keyboard Interrupts

Port A	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTA7/ KBI1P7	PTA6/ KBI1P6	PTA5/ KBI1P5	PTA4/ KBI1P4	PTA3/ KBI1P3	PTA2/ KBI1P2	PTA1/ KBI1P1	PTA0/ KBI1P0/ VPP

**Figure 13-2. Port A Pin Names**

Port A is an 8-bit port shared among the KBI keyboard interrupt inputs and general-purpose I/O. Any pins enabled as KBI inputs will be forced to act as inputs.

Port A pins are available as general-purpose I/O pins controlled by the port A data (PTAD), data direction (PTADD), pullup enable (PTAPE), and slew rate control (PTASE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#), for more information about general-purpose I/O control.

Port A can be configured to be keyboard interrupt input pins. Refer to [Chapter 16, “MCU Keyboard Interrupt \(KBI\)”](#), for more information about using port A pins as keyboard interrupts pins.

### 13.3.2 Port B and Analog to Digital Converter Inputs

Port B	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTB7/ AD1P7	PTB6/ AD1P6	PTB5/ AD1P5	PTB4/ AD1P4	PTB3/ AD1P3	PTB2/ AD1P2	PTB1/ AD1P1	PTB0/ AD1P0

**Figure 13-3. Port B Pin Names**

Port B is an 8-bit port shared among the ATD inputs and general-purpose I/O. Any pin enabled as an ATD input will be forced to act as an input.

Port B pins are available as general-purpose I/O pins controlled by the port B data (PTBD), data direction (PTBDD), pullup enable (PTBPE), and slew rate control (PTBSE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#), for more information about general-purpose I/O control.

When the ATD module is enabled, analog pin enables are used to specify which pins on port B will be used as ATD inputs. Refer to [Chapter 20, “Analog to Digital \(ATD\) Module”](#), for more information about using port B pins as ATD pins.

### 13.3.3 Port C and SCI2, IIC, and High-Current Drivers

Port C	Bit 7	6	5	3	3	2	1	Bit 0
MCU Pin:	PTC7	PTC6	PTC5	PTC4	PTC3/ SCL1	PTC2/ SDA1	PTC1/ RxD2	PTC0/ TxD2

**Figure 13-4. Port C Pin Names**

Port C is an 8-bit port which is shared among the SCI2 and IIC1 modules, and general-purpose I/O. When SCI2 or IIC1 modules are enabled, the pin direction will be controlled by the module or function. Port C has high current output drivers.

Port C pins are available as general-purpose I/O pins controlled by the port C data (PTCD), data direction (PTCDD), pullup enable (PTCPE), and slew rate control (PTCSE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#), for more information about general-purpose I/O control.

When the SCI2 module is enabled, PTC0 serves as the SCI2 module’s transmit pin (TxD2) and PTC1 serves as the receive pin (RxD2). Refer to [Chapter 18, “MCU Serial Communications Interface \(SCI\)”](#), for more information about using PTC0 and PTC1 as SCI pins

When the IIC module is enabled, PTC2 serves as the IIC modules’s serial data input/output pin (SDA1) and PTC3 serves as the clock pin (SCL1). Refer to [Chapter 19, “Inter-Integrated Circuit \(IIC\)”](#), for more information about using PTC2 and PTC3 as IIC pins.

### 13.3.4 Port D, TPM1 and TPM2

Port D	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTD7/ TPM2CH4	PTD6/ TPM2CH3	PTD5/ TPM2CH2	PTD4/ TPM2CH1	PTD3/ TPM2CH0	PTD2/ TPM1CH2	PTD1/ TPM1CH1	PTD0/ TPM1CH0

**Figure 13-5. Port D Pin Names**

Port D is an 8-bit port shared with the two TPM modules, TPM1 and TPM2, and general-purpose I/O. When the TPM1 or TPM2 modules are enabled in output compare or input capture modes of operation, the pin direction will be controlled by the module function.

Port D pins are available as general-purpose I/O pins controlled by the port D data (PTDD), data direction (PTDDD), pullup enable (PTDPE), and slew rate control (PTDSE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#) for more information about general-purpose I/O control.

The TPM2 module can be configured to use PTD7–PTD3 as either input capture, output compare, PWM, or external clock input pins (PTD3 only). Refer to [Chapter 17, “MCU Timer/PWM \(TPM Module\)”](#) for more information about using PTD7–PTD3 as timer pins.

The TPM1 module can be configured to use PTD2–PTD0 as either input capture, output compare, PWM, or external clock input pins (PTD0 only). Refer to [Chapter 17, “MCU Timer/PWM \(TPM Module\)”](#) for more information about using PTD2–PTD0 as timer pins.

### 13.3.5 Port E, SCI1, and SPI

Port E	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTE7	PTE6	PTE5/ SPSCK1	PTE4/ MOSI1	PTE3/ MISO1	PTE2/ SS1	PTE1/ RxD1	PTE0/ TxD1

**Figure 13-6. Port E Pin Names**

Port E is an 8-bit port shared with the SCI1 module, SPI1 module, and general-purpose I/O. When the SCI or SPI modules are enabled, the pin direction will be controlled by the module function.

Port E pins are available as general-purpose I/O pins controlled by the port E data (PTED), data direction (PTEDD), pullup enable (PTEPE), and slew rate control (PTESE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#) for more information about general-purpose I/O control.

When the SCI1 module is enabled, PTE0 serves as the SCI1 module’s transmit pin (TxD1) and PTE1 serves as the receive pin (RxD1). Refer to [Chapter 18, “MCU Serial Communications Interface \(SCI\)”](#) for more information about using PTE0 and PTE1 as SCI pins.

When the SPI module is enabled, PTE2 serves as the SPI module’s slave select pin ( $\overline{SS1}$ ), PTE3 serves as the master-in slave-out pin (MISO1), PTE4 serves as the master-out slave-in pin (MOSI1), and PTE5 serves as the SPI clock pin (SPSCK1). Refer to [Chapter 4, “MC1321x Serial Peripheral Interface \(SPI\)”](#) for more information about using PTE5–PTE2 as SPI pins.

### 13.3.6 Port F and High-Current Drivers

Port F	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0

Figure 13-7. Port F Pin Names

Port F is an 8-bit port general-purpose I/O that is not shared with any peripheral module. Port F has high current output drivers.

Port F pins are available as general-purpose I/O pins controlled by the port F data (PTFD), data direction (PTFDD), pullup enable (PTFPE), and slew rate control (PTFSE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#) for more information about general-purpose I/O control.

### 13.3.7 Port G, BKGD/MS, and Oscillator

Port G	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTG7	PTG6	PTG5	PTG4	PTG3	PTG2/ EXTAL	PTG1/ XTAL	PTG0/ BKGD/MS

Figure 13-8. Port G Pin Names

Port G is an 8-bit port which is shared among the background/mode select function, oscillator, and general-purpose I/O. When the background/mode select function or oscillator is enabled, the pin direction will be controlled by the module function.

Port G pins are available as general-purpose I/O pins controlled by the port G data (PTGD), data direction (PTGDD), pullup enable (PTGPE), and slew rate control (PTGSE) registers. Refer to [Section 13.4, “Parallel I/O Controls”](#) for more information about general-purpose I/O control.

The internal pullup for PTG0 is enabled when the background/mode select function is enabled, regardless of the state of PTGPE0. During reset, the BKGD/MS pin functions as a mode select pin. After the MCU is out of reset, the BKGD/MS pin becomes the background communications input/output pin. The PTG0 can be configured to be a general-purpose output pin. Refer to [Chapter 7, “Modem Modes of Operation”](#), [Chapter 12, “MCU Resets, Interrupts, and System Configuration”](#), and [Chapter 21, “Development Support”](#) for more information about using this pin.

The ICG module can be configured to use PTG2–PTG1 ports as crystal oscillator or external clock pins.

Refer to [Chapter 19, “Inter-Integrated Circuit \(IIC\)”](#) for more information about using these pins as oscillator pins.

## 13.4 Parallel I/O Controls

Provided no on-chip peripheral is controlling a port pin, the pins operate as general-purpose I/O pins that are accessed and controlled by a data register (PTxD), a data direction register (PTxDD), a pullup enable register (PTxPE), and a slew rate control register (PTxSE) where x is A, B, C, D, E, F, or G.

Reads of the data register return the pin value (if PTxDDn = 0) or the contents of the port data register (if PTxDDn = 1). Writes to the port data register are latched into the port register whether the pin is controlled by an on-chip peripheral or the pin is configured as an input. If the corresponding pin is not controlled by a peripheral and is configured as an output, this level will be driven out the port pin.

### 13.4.1 Data Direction Control

The data direction control bits determine whether the pin output driver is enabled, and they control what is read for port data register reads. Each port pin has a data direction control bit. When PTxDDn = 0, the corresponding pin is an input and reads of PTxD return the pin value. When PTxDDn = 1, the corresponding pin is an output and reads of PTxD return the last value written to the port data register. When a peripheral module or system function is in control of a port pin, the data direction control still controls what is returned for reads of the port data register, even though the peripheral system has overriding control of the actual pin direction.

For the HCS08 MCU, reads of PTG0/BKGD/MS will return the value on the output pin.

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

### 13.4.2 Internal Pullup Control

An internal pullup device can be enabled for each port pin that is configured as an input (PTxDDn = 0). The pullup device is available for a peripheral module to use, provided the peripheral is enabled and is an input function as long as the PTxDDn = 0.

For the four configurable KBI module inputs on PTA7–PTA4, when a pin is configured to detect rising edges, the port pullup enable associated with the pin (PTAPEn) selects a pulldown rather than a pullup device.

### 13.4.3 Slew Rate Control

Slew rate control can be enabled for each port pin that is configured as an output (PTxDDn = 1) or if a peripheral module is enabled and its function is an output. Not all peripheral modules' outputs have slew rate control; refer to [Chapter 2, “MC1321x Pins and Connections”](#) for more information about which pins have slew rate control.

## 13.5 Stop Modes

Depending on the Stop Mode, I/O functions differently as the result of executing a STOP instruction. An explanation of I/O behavior for the various stop modes follows:

- When the MCU enters Stop1 Mode, all internal registers including general-purpose I/O control and data registers are powered down. All of the general-purpose I/O pins assume their reset state: output buffers and pullups turned off. Upon exit from Stop1, all I/O must be initialized as if the MCU had been reset.
- When the MCU enters Stop2 Mode, the internal registers are powered down as in Stop1 but the I/O pin states are latched and held. For example, a port pin that is an output driving low continues to function as an output driving low even though its associated data direction and output data registers are powered down internally. Upon exit from Stop2, the pins continue to hold their states until a 1 is written to the PPDACK bit. To avoid discontinuity in the pin state following exit from Stop2, the user must restore the port control and data registers to the values they held before entering Stop2. These values can be stored in RAM before entering Stop2 because the RAM is maintained during Stop2.
- 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.

### NOTE

For low power modes, unused GPIO including unpinned-out signals must be initialized.

## 13.6 Parallel I/O Registers and Control Bits

This section provides information about all registers and control bits associated with the parallel I/O ports.

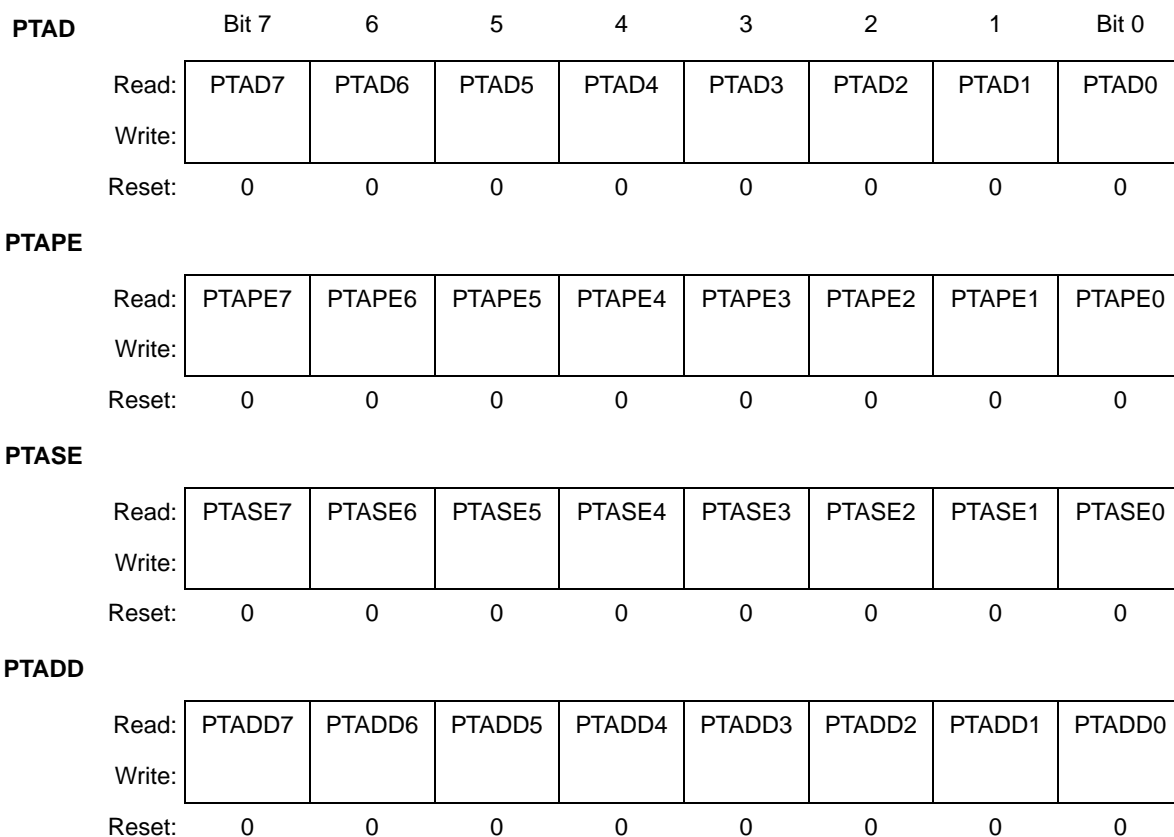
Refer to tables in [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all parallel I/O registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

### 13.6.1 Port A Registers (PTAD, PTAPE, PTASE, and PTADD)

Port A includes eight pins shared between general-purpose I/O and the KBI module. Port A pins used as general-purpose I/O pins are controlled by the port A data (PTAD), data direction (PTADD), pullup enable (PTAPE), and slew rate control (PTASE) registers.

If the KBI takes control of a port A pin, the corresponding PTASE bit is ignored since the pin functions as an input. As long as PTADD is 0, the PTAPE controls the pullup enable for the KBI function. Reads of PTAD will return the logic value of the corresponding pin, provided PTADD is 0.




**Figure 13-9. Port A Registers**

PTAD<sub>n</sub> — Port A Data Register Bit *n* (*n* = 0–7)

For port A pins that are inputs, reads return the logic level on the pin. For port A pins that are configured as outputs, reads return the last value written to this register.

Writes are latched into all bits of this register. For port A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

PTAPEn — Pullup Enable for Port A Bit *n* (*n* = 0–7)

For port A pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled provided the corresponding PTADD<sub>n</sub> is 0. For port A pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled. When any of bits 7 through 4 of port A are enabled as KBI inputs and are configured to detect rising edges/high levels, the pullup enable bits enable pulldown rather than pullup devices.

1 = Internal pullup device enabled.

0 = Internal pullup device disabled.

**PTASE<sub>n</sub>** — Slew Rate Control Enable for Port A Bit *n* (*n* = 0–7)

For port A pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port A pins that are configured as inputs, these bits are ignored.

- 1 = Slew rate control enabled.
- 0 = Slew rate control disabled.

**PTADD<sub>n</sub>** — Data Direction for Port A Bit *n* (*n* = 0–7)

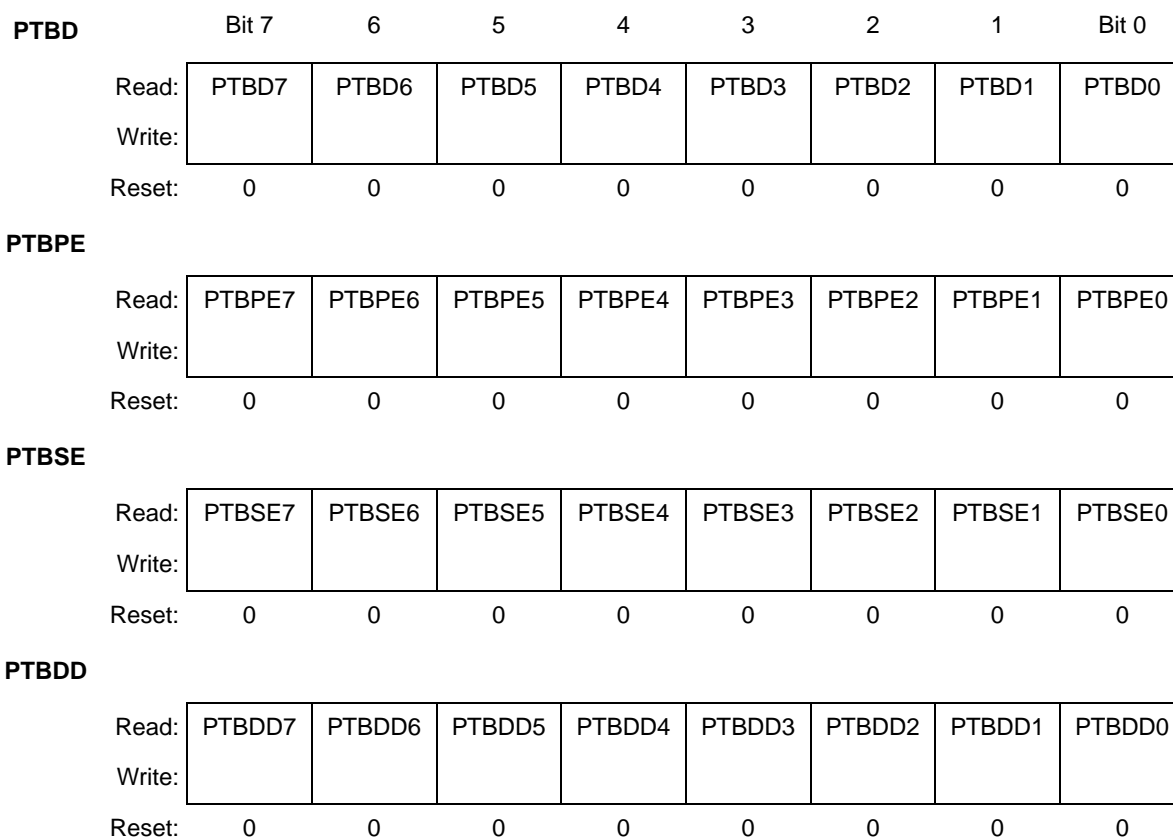
These read/write bits control the direction of port A pins and what is read for PTAD reads.

- 1 = Output driver enabled for port A bit *n* and PTAD reads return the contents of PTAD<sub>*n*</sub>.
- 0 = Input (output driver disabled) and reads return the pin value.

### 13.6.2 Port B Registers (PTBD, PTBPE, PTBSE, and PTBDD)

Port B includes eight general-purpose I/O pins that share with the ATD function. Port B pins used as general-purpose I/O pins are controlled by the port B data (PTBD), data direction (PTBDD), pullup enable (PTBPE), and slew rate control (PTBSE) registers.

If the ATD takes control of a port B pin, the corresponding PTBDD, PTBSE, and PTBPE bits are ignored. When a port B pin is being used as an ATD pin, reads of PTBD will return a 0 of the corresponding pin, provided PTBDD is 0.



**Figure 13-10. Port B Registers**

**PTBDn** — Port B Data Register Bit n (n = 0–7)

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 on the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

**PTBPEn** — Pullup Enable for Port B Bit n (n = 0–7)

For port B pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port B pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.

0 = Internal pullup device disabled.

**PTBSEn** — Slew Rate Control Enable for Port B Bit n (n = 0–7)

For port B pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port B pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.

0 = Slew rate control disabled.

**PTBDDn** — Data Direction for Port B Bit n (n = 0–7)

These read/write bits control the direction of port B pins and what is read for PTBD reads.

1 = Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.

0 = Input (output driver disabled) and reads return the pin value.

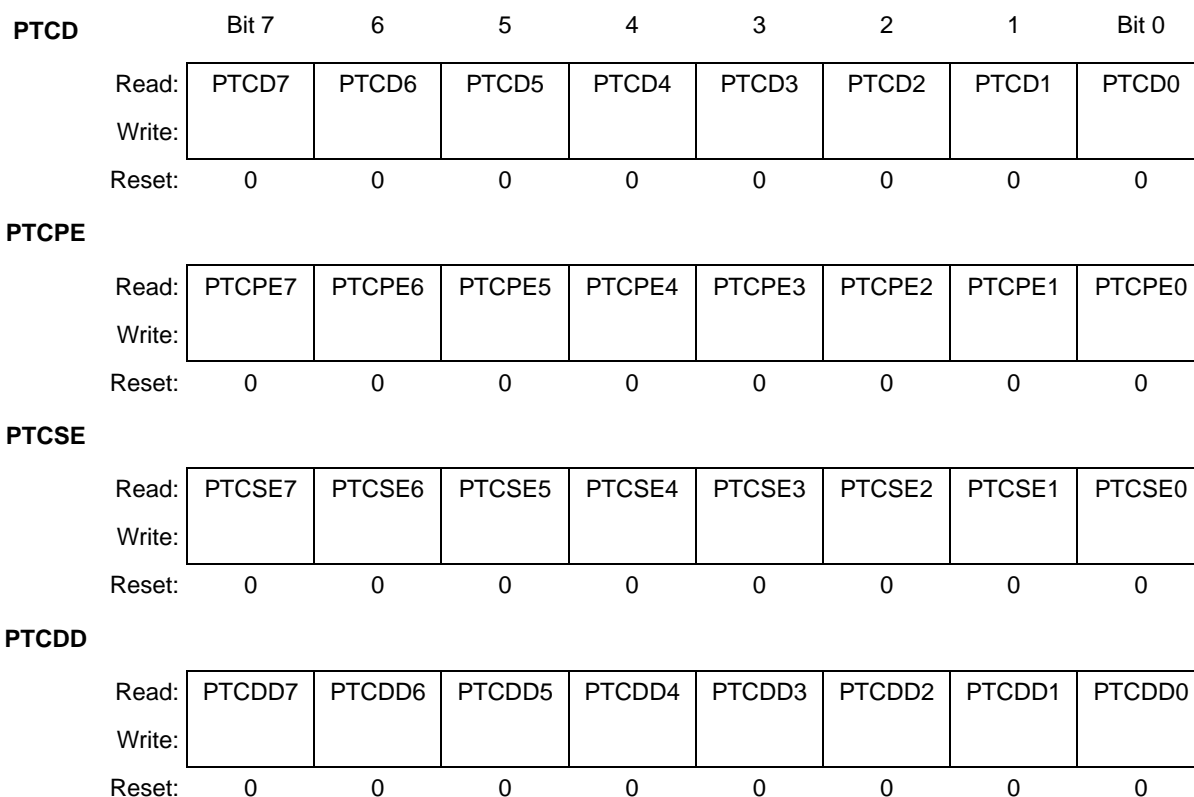
### 13.6.3 Port C Registers (PTCD, PTCPE, PTCSE, and PTCDD)

Port C includes eight general-purpose I/O pins that share with the SCI2 and IIC modules. Port C pins used as general-purpose I/O pins are controlled by the port C data (PTCD), data direction (PTCDD), pullup enable (PTCPE), and slew rate control (PTCSE) registers.

If the SCI2 takes control of a port C pin, the corresponding PTCDD bit is ignored. PTCSE can be used to provide slew rate on the SCI2 transmit pin, TxD2. PTCPE can be used, provided the corresponding PTCDD bit is 0, to provide a pullup device on the SCI2 receive pin, RxD2.

If the IIC takes control of a port C pin, the corresponding PTCDD bit is ignored. PTCSE can be used to provide slew rate on the IIC serial data pin (SDA1), when in output mode and the IIC clock pin (SCL1). PTCPE can be used, provided the corresponding PTCDD bit is 0, to provide a pullup device on the IIC serial data pin, when in receive mode.

Reads of PTCD will return the logic value of the corresponding pin, provided PTCDD is 0.



**Figure 13-11. Port C Registers**

**PTCD<sub>n</sub>** — Port C Data Register Bit *n* (*n* = 0–7)

For port C pins that are inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register.

Writes are latched into all bits of this register. For port C pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

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

**PTCPE<sub>n</sub>** — Pullup Enable for Port C Bit *n* (*n* = 0–7)

For port C pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port C pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.

0 = Internal pullup device disabled.

**PTCSE<sub>n</sub>** — Slew Rate Control Enable for Port C Bit *n* (*n* = 0–7)

For port C pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port B pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.

0 = Slew rate control disabled.

PTCDD<sub>n</sub> — Data Direction for Port C Bit n (n = 0–7)

These read/write bits control the direction of port C pins and what is read for PTCDD reads.

1 = Output driver enabled for port C bit n and PTCDD reads return the contents of PTCDD<sub>n</sub>.

0 = Input (output driver disabled) and reads return the pin value.

### 13.6.4 Port D Registers (PTDD, PTDPE, PTDSE, and PTDDD)

Port D includes eight pins shared between general-purpose I/O, TPM1, and TPM2. Port D pins used as general-purpose I/O pins are controlled by the port D data (PTDD), data direction (PTDDD), pullup enable (PTDPE), and slew rate control (PTDSE) registers.

If a TPM takes control of a port D pin, the corresponding PTDDD bit is ignored. When the TPM is in output compare mode, the corresponding PTDSE can be used to provide slew rate on the pin. When the TPM is in input capture mode, the corresponding PTDPE can be used, provided the corresponding PTDDD bit is 0, to provide a pullup device on the pin.

Reads of PTDD will return the logic value of the corresponding pin, provided PTDDD is 0.

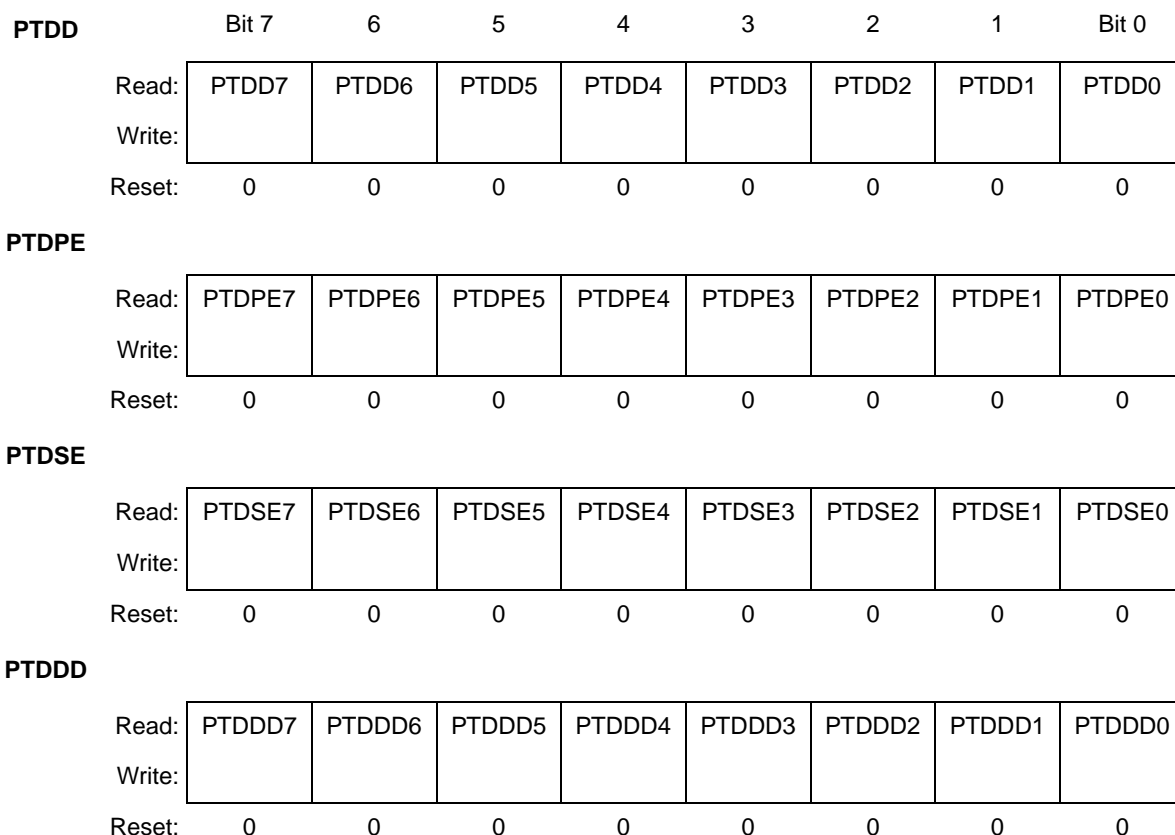


Figure 13-12. Port D Registers

PTDD<sub>n</sub> — Port D Data Register Bit n (n = 0–7)

For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register.

Writes are latched into all bits of this register. For port D pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

**PTDPE<sub>n</sub>** — Pullup Enable for Port D Bit *n* (*n* = 0–7)

For port D pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port D pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.

0 = Internal pullup device disabled.

**PTDSE<sub>n</sub>** — Slew Rate Control Enable for Port D Bit *n* (*n* = 0–7)

For port D pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port D pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.

0 = Slew rate control disabled.

**PTDDD<sub>n</sub>** — Data Direction for Port D Bit *n* (*n* = 0–7)

These read/write bits control the direction of port D pins and what is read for PTDD reads.

1 = Output driver enabled for port D bit *n* and PTDD reads return the contents of PTDD<sub>*n*</sub>.

0 = Input (output driver disabled) and reads return the pin value.

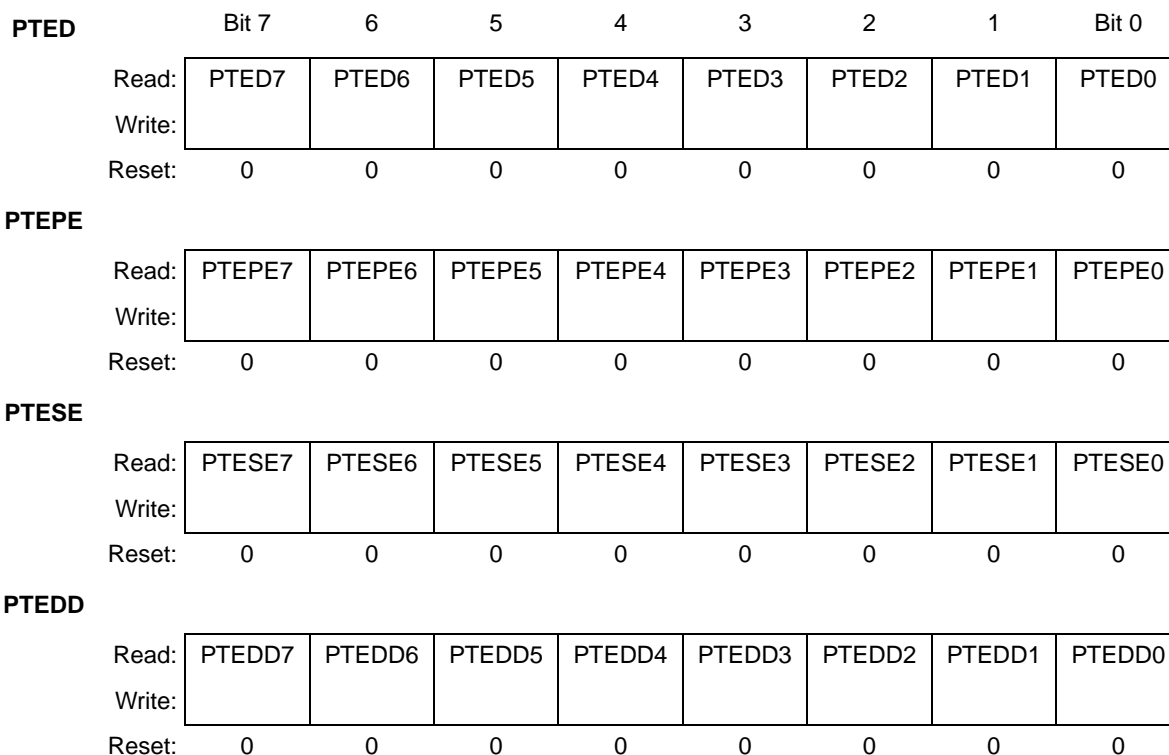
### 13.6.5 Port E Registers (PTED, PTEPE, PTESE, and PTEDD)

Port E includes eight general-purpose I/O pins that share with the SCI1 and SPI modules. Port E pins used as general-purpose I/O pins are controlled by the port E data (PTED), data direction (PTEDD), pullup enable (PTEPE), and slew rate control (PTESE) registers.

If the SCI1 takes control of a port E pin, the corresponding PTEDD bit is ignored. PTESE can be used to provide slew rate on the SCI1 transmit pin, TxD1. PTEPE can be used, provided the corresponding PTEDD bit is 0, to provide a pullup device on the SCI1 receive pin, RxD1.

If the SPI takes control of a port E pin, the corresponding PTEDD bit is ignored. PTESE can be used to provide slew rate on the SPI serial output pin (MOSI1 or MISO1) and serial clock pin (SPSCK1) depending on the SPI operational mode. PTEPE can be used, provided the corresponding PTEDD bit is 0, to provide a pullup device on the SPI serial input pins (MOSI1 or MISO1) and slave select pin ( $\overline{SS1}$ ) depending on the SPI operational mode.

Reads of PTED will return the logic value of the corresponding pin, provided PTEDD is 0.


**Figure 13-13. Port E Registers**
**PTED<sub>n</sub> — Port E Data Register Bit n (n = 0–7)**

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 in this register. For port E pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

Reset forces PTED to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

**PTEPE<sub>n</sub> — Pullup Enable for Port E Bit n (n = 0–7)**

For port E pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port E pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.

0 = Internal pullup device disabled.

**PTESE<sub>n</sub> — Slew Rate Control Enable for Port E Bit n (n = 0–7)**

For port E pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port E pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.

0 = Slew rate control disabled.

PTEDDn — Data Direction for Port E Bit n (n = 0–7)

These read/write bits control the direction of port E pins and what is read for PTED reads.

1 = Output driver enabled for port E bit n and PTED reads return the contents of PTEDn.

0 = Input (output driver disabled) and reads return the pin value.

### 13.6.6 Port F Registers (PTFD, PTFPE, PTFSE, and PTFDD)

Port F includes eight general-purpose I/O pins that are not shared with any peripheral module. Port F pins used as general-purpose I/O pins are controlled by the port F data (PTFD), data direction (PTFDD), pullup enable (PTFPE), and slew rate control (PTFSE) registers.

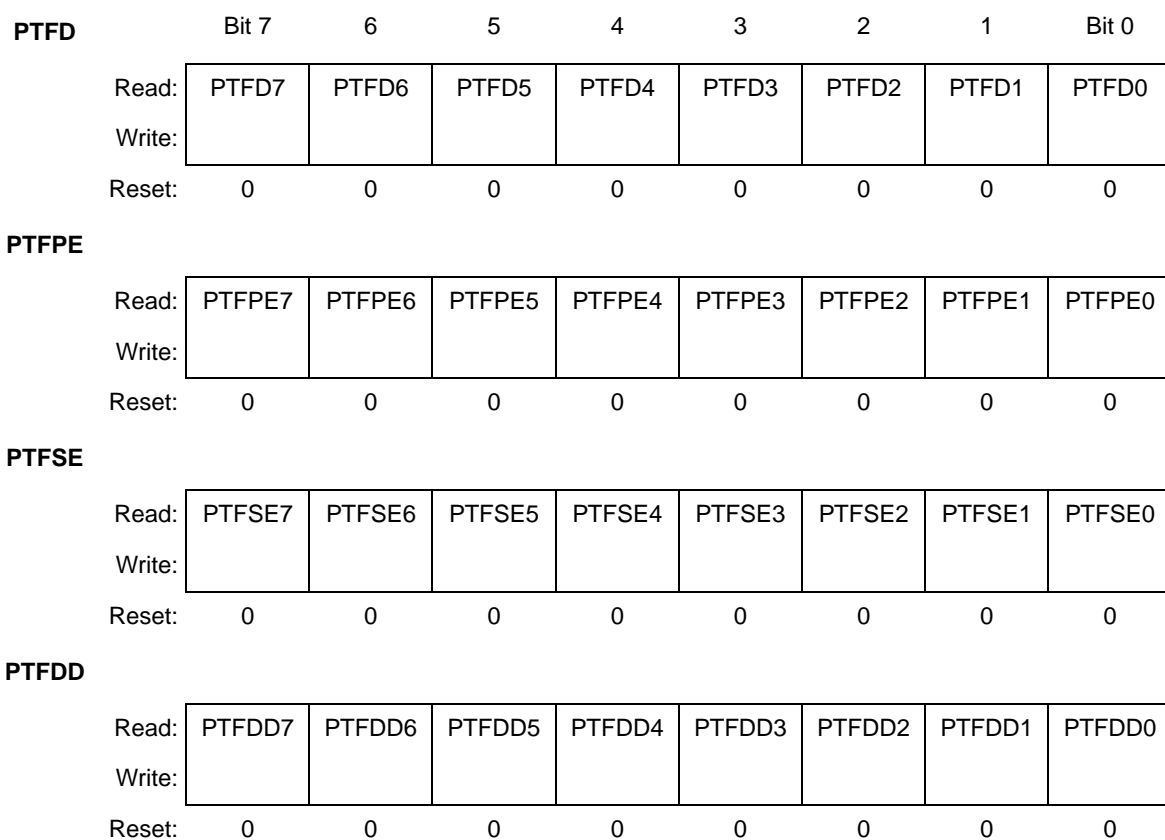


Figure 13-14. Port F Registers

PTFDn — Port PTF Data Register Bit n (n = 0–7)

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 disabled.



**PTFPEn** — Pullup Enable for Port F Bit n (n = 0–7)

For port F pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port F pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.

0 = Internal pullup device disabled.

**PTFSEn** — Slew Rate Control Enable for Port F Bit n (n = 0–7)

For port F pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port F pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.

0 = Slew rate control disabled.

**PTFDDn** — Data Direction for Port F Bit n (n = 0–7)

These read/write bits control the direction of port F pins and what is read for PTFD reads.

1 = Output driver enabled for port F bit n and PTFD reads return the contents of PTFDn.

0 = Input (output driver disabled) and reads return the pin value.

### 13.6.7 Port G Registers (PTGD, PTGPE, PTGSE, and PTGDD)

Port G includes eight general-purpose I/O pins that are shared with BKGD/MS function and the oscillator or external clock pins. Port G pins used as general-purpose I/O pins are controlled by the port G data (PTGD), data direction (PTGDD), pullup enable (PTGPE), and slew rate control (PTGSE) registers.

Port pin PTG0, while in reset, defaults to the BKGD/MS pin. After the MCU is out of reset, PTG0 can be configured to be a general-purpose output pin. When BKGD/MS takes control of PTG0, the corresponding PTGDD, PTGPE, and PTGPSE bits are ignored.

Port pins PTG1 and PTG2 can be configured to be oscillator or external clock pins. When the oscillator takes control of a port G pin, the corresponding PTGD, PTGDD, PTGSE, and PTGPE bits are ignored.

Reads of PTGD will return the logic value of the corresponding pin, provided PTGDD is 0.

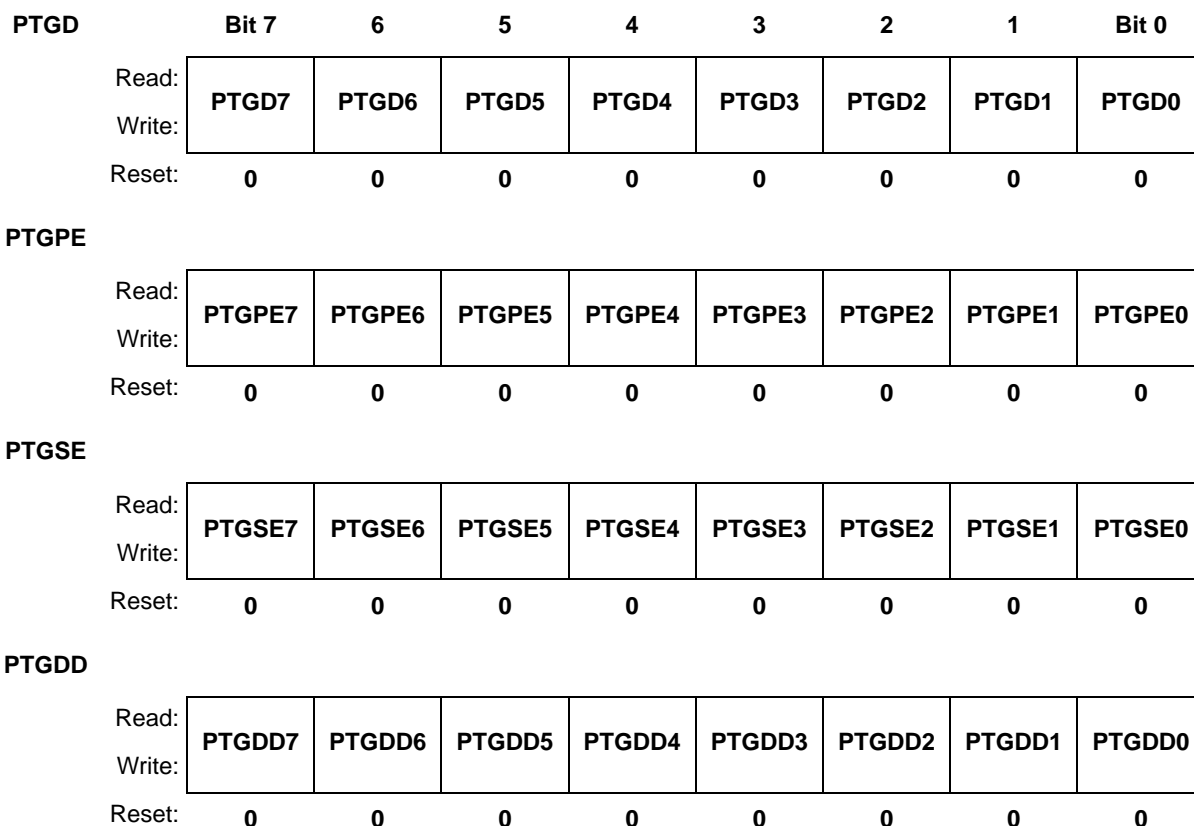


Figure 13-15. Port G Registers

**PTGD<sub>n</sub>** — Port PTG Data Register Bit n (n = 0–7)

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 disabled.

**PTGPE<sub>n</sub>** — Pullup Enable for Port G Bit n (n = 0–7)

For port G pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port G pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

- 1 = Internal pullup device enabled.
- 0 = Internal pullup device disabled.

PTGSE<sub>n</sub> — Slew Rate Control Enable for Port G Bit *n* (*n* = 0–7)

For port G pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port G pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.

0 = Slew rate control disabled.

PTGDD<sub>n</sub> — Data Direction for Port G Bit *n* (*n* = 0–7)

These read/write bits control the direction of port G pins and what is read for PTGD reads.

1 = Output driver enabled for port G bit *n* and PTGD reads return the contents of PTGD<sub>n</sub>.

0 = Input (output driver disabled) and reads return the pin value.

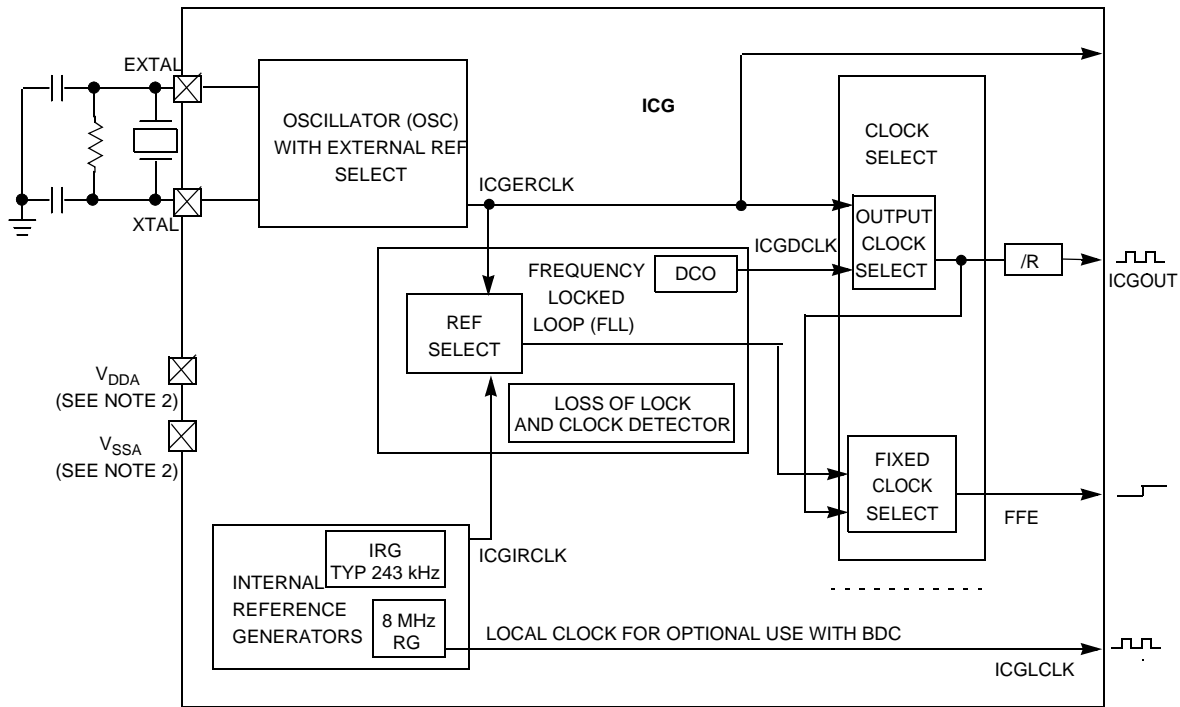


# Chapter 14

## MCU Internal Clock Generator (ICG)

### 14.1 Introduction

Figure 14-1 is a top-level diagram that shows the functional organization of the internal clock generation (ICG) module. This section includes a general description and a feature list.



NOTES:

1. See Table 14-1 for specific use of ICGOUT, FFE, ICGLCLK, ICGERCLK
2. Not all HCS08 microcontrollers have unique supply pins for the ICG. See the device pin assignments in Chapter 2, "MC1321x Pins and Connections" for details.

**Figure 14-1. ICG Block Diagram**

The ICG provides multiple options for clock sources. This offers a user great flexibility when making choices between cost, precision, current draw, and performance. As seen in Figure 14-1, the ICG consists of four functional blocks. Each of these is briefly described here and then in more detail in a later section.

- **Oscillator block** — The oscillator block provides means for connecting an external crystal or resonator. Two frequency ranges are software selectable to allow optimal start-up and stability. Alternatively, the oscillator block can be used to route an external square wave to the system clock. External sources can provide a very precise clock source. The oscillator is capable of being configured for low power mode or high amplitude mode as selected by HGO.
- **Internal reference generator** — The internal reference generator consists of two controlled clock sources. One is designed to be approximately 8 MHz and can be selected as a local clock for the background debug controller. The other internal reference clock source is typically 243 kHz and can be trimmed for finer accuracy via software when a precise timed event is input to the MCU. This provides a highly reliable, low-cost clock source.

- **Frequency-locked loop** — A frequency-locked loop (FLL) stage takes either the internal or external clock source and multiplies it to a higher frequency. Status bits provide information when the circuit has achieved lock and when it falls out of lock. Additionally, this block can monitor the external reference clock and signals whether the clock is valid or not.
- **Clock select block** — The clock select block provides several switch options for connecting different clock sources to the system clock tree. ICGDCLK is the multiplied clock frequency out of the FLL, ICGERCLK is the reference clock frequency from the crystal or external clock source, and FFE (fixed frequency enable) is a control signal used to control the system fixed frequency clock (XCLK). ICGLCLK is the clock source for the background debug controller (BDC).

The module is intended to be very user friendly with many of the features occurring automatically without user intervention. To quickly configure the module, go to [Section 14.4, “Initialization/Application Information”](#) and pick an example that best suits the application needs.

### 14.1.1 Features

Features of the ICG and clock distribution system:

- Several options for the primary clock source allow a wide range of cost, frequency, and precision choices:
  - 32 kHz–100 kHz crystal or resonator
  - 1 MHz–16 MHz crystal or resonator
  - External clock
  - Internal reference generator
- Defaults to Self-clocked Mode to minimize start-up delays
- Frequency-locked loop (FLL) generates 8 MHz to 40 MHz (for bus rates up to 20 MHz)
  - Uses external or internal clock as reference frequency
- Automatic lockout of non-running clock sources
- Reset or interrupt on loss of clock or loss of FLL lock
- Digitally-controlled oscillator (DCO) preserves previous frequency settings, allowing fast frequency lock when recovering from Stop3 Mode
- DCO will maintain operating frequency during a loss or removal of reference clock
- Post-FLL divider selects 1 of 8 bus rate divisors (/1 through /128)
- Separate self-clocked source for real-time interrupt
- Trimmable internal clock source supports SCI communications without additional external components
- Automatic FLL engagement after lock is acquired
- Selectable low-power/high-gain oscillator modes

## 14.1.2 Modes of Operation

This is a high-level description only. Detailed descriptions of operating modes are contained in [Section 14.3, “Functional Description”](#).

- **Mode 1 — Off**  
The output clock, ICGOUT, is static. This mode may be entered when the STOP instruction is executed.
- **Mode 2 — Self-clocked (SCM)**  
Default mode of operation that is entered out of reset. The ICG’s FLL is open loop and the digitally controlled oscillator (DCO) is free running at a frequency set by the filter bits.
- **Mode 3 — FLL Engaged Internal (FEI)**  
In this mode, the ICG’s FLL is used to create frequencies that are programmable multiples of the internal reference clock.
  - FLL Engaged Internal unlocked is a transition state which occurs while the FLL is attempting to lock. The FLL DCO frequency is off target and the FLL is adjusting the DCO to match the target frequency.
  - FLL engaged internal locked is a state which occurs when the FLL detects that the DCO is locked to a multiple of the internal reference.
- **Mode 4 — FLL Bypassed External (FBE)**  
In this mode, the ICG is configured to bypass the FLL and use an external clock as the clock source.
- **Mode 5 — FLL Engaged External (FEE)**  
The ICG’s FLL is used to generate frequencies that are programmable multiples of the external clock reference.
  - FLL Engaged External unlocked is a transition state which occurs while the FLL is attempting to lock. The FLL DCO frequency is off target and the FLL is adjusting the DCO to match the target frequency.
  - FLL Engaged External locked is a state which occurs when the FLL detects that the DCO is locked to a multiple of the internal reference.

## 14.2 Oscillator Pins

The oscillator pins are used to provide an external clock source for the MCU.

### 14.2.1 EXTAL— External Reference Clock / Oscillator Input

If upon the first write to ICGC1, either FEE Mode or FBE Mode is selected, this pin functions as either the external clock input or the input of the oscillator circuit as determined by REFS. If upon the first write to ICGC1, either FEI Mode or SCM Mode is selected, this pin is not used by the ICG.

### 14.2.2 XTAL— Oscillator Output

If upon the first write to ICGC1, either FEE Mode or FBE Mode is selected, this pin functions as the output of the oscillator circuit. If upon the first write to ICGC1, either FEI Mode or SCM Mode is selected, this pin is not used by the ICG. The oscillator is capable of being configured to provide a higher amplitude output for improved noise immunity. This Mode of operation is selected by HGO = 1.

### 14.2.3 External Clock Connections

If an external clock is used, then the pins are connected as shown in [Figure 14-2](#).

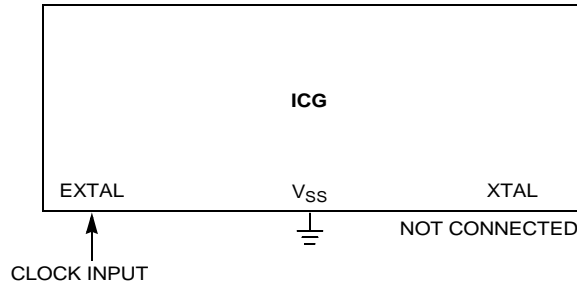


Figure 14-2. External Clock Connections

### 14.2.4 External Crystal/Resonator Connections

If an external crystal/resonator frequency reference is used, then the pins are connected as shown in [Figure 14-3](#).

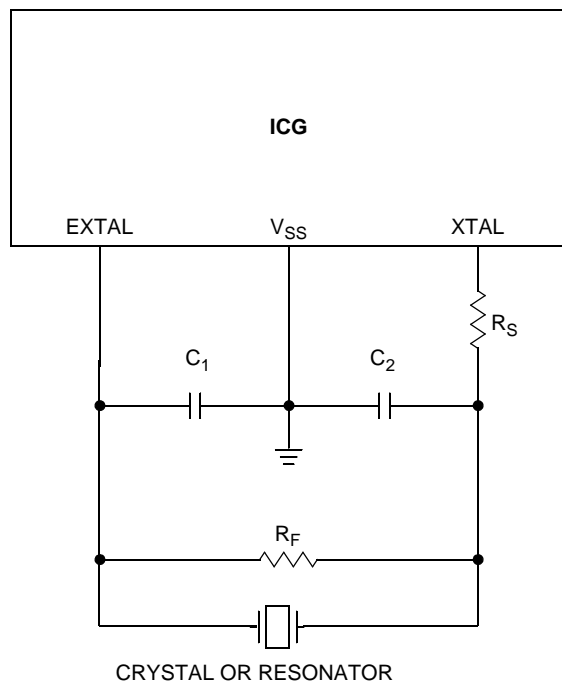


Figure 14-3. External Frequency Reference Connection



## 14.3 Functional Description

This section provides a functional description of each of the five operating modes of the ICG. Also covered are the loss of clock and loss of lock errors and requirements for entry into each mode. The ICG is very flexible, and in some configurations, it is possible to exceed certain clock specifications. When using the FLL, configure the ICG so that the frequency of ICGDCLK does not exceed its maximum value to ensure proper MCU operation.

### 14.3.1 Off Mode (Off)

Normally when the CPU enters Stop Mode, the ICG will cease all clock activity and is in the off state. However there are two cases to consider when clock activity continues while the CPU is in Stop Mode.

#### 14.3.1.1 BDM Active

When the BDM is enabled ( $ENBDM = 1$ ), the ICG continues activity as originally programmed. This allows access to memory and control registers via the BDC.

#### 14.3.1.2 OSCSTEN Bit Set

When the oscillator is enabled in Stop Mode ( $OSCSTEN = 1$ ), the individual clock generators are enabled but the clock feed to the rest of the MCU is turned off. This option is provided to avoid long oscillator start-up times if necessary, or to run the RTI from the oscillator during Stop3.

#### 14.3.1.3 Stop/Off Mode Recovery

Upon the CPU exiting Stop Mode due to an interrupt, the previously set control bits are valid and the system clock feed resumes. If FEE is selected, the ICG will source the internal reference until the external clock is stable. If FBE is selected, the ICG will wait for the external clock to stabilize before enabling ICGOUT.

Upon the CPU exiting Stop Mode due to a reset, the previously set ICG control bits are ignored and the default reset values applied. Therefore the ICG will exit stop in SCM mode configured for an approximately 8 MHz DCO output (4 MHz bus clock) with trim value maintained. If using a crystal, 4096 clocks are detected prior to engaging ICGERCLK. This is incorporated in crystal start-up time.

### 14.3.2 Self-Clocked Mode (SCM)

Self-clocked Mode (SCM) is the default mode of operation and is entered when any of the following conditions occur:

- After any reset.
- Exiting from Off Mode when  $CLKS$  does not equal 10. If  $CLKS = X1$ , the ICG enters this state temporarily until the DCO is stable ( $DCOS = 1$ ).
- $CLKS$  bits are written from  $X1$  to  $00$ .
- $CLKS = 1X$  and ICGERCLK is not detected (both  $ERCS = 0$  and  $LOCS = 1$ ).

In this state, the FLL loop is open. The DCO is on, and the output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / R$ . The ICGDCLK frequency can be varied from 8 MHz to 40 MHz by writing a new value into the filter registers (ICGFLTU and ICGFLTLL). This is the only mode in which the filter registers can be written.

If this mode is entered due to a reset,  $f_{ICGDCLK}$  will default to  $f_{Self\_reset}$  which is nominally 8 MHz. If this mode is entered from FLL engaged internal,  $f_{ICGDCLK}$  will maintain the previous frequency. If this mode is entered from FLL engaged external (either by programming CLKS or due to a loss of external reference clock),  $f_{ICGDCLK}$  will maintain the previous frequency, but ICGOUT will double if the FLL was unlocked. If this mode is entered from Off Mode,  $f_{ICGDCLK}$  will be equal to the frequency of ICGDCLK before entering Off Mode. If CLKS bits are set to 01 or 11 coming out of the Off state, the ICG enters this mode until ICGDCLK is stable as determined by the DCOS bit. Once ICGDCLK is considered stable, the ICG automatically closes the loop by switching to FLL engaged (internal or external) as selected by the CLKS bits.

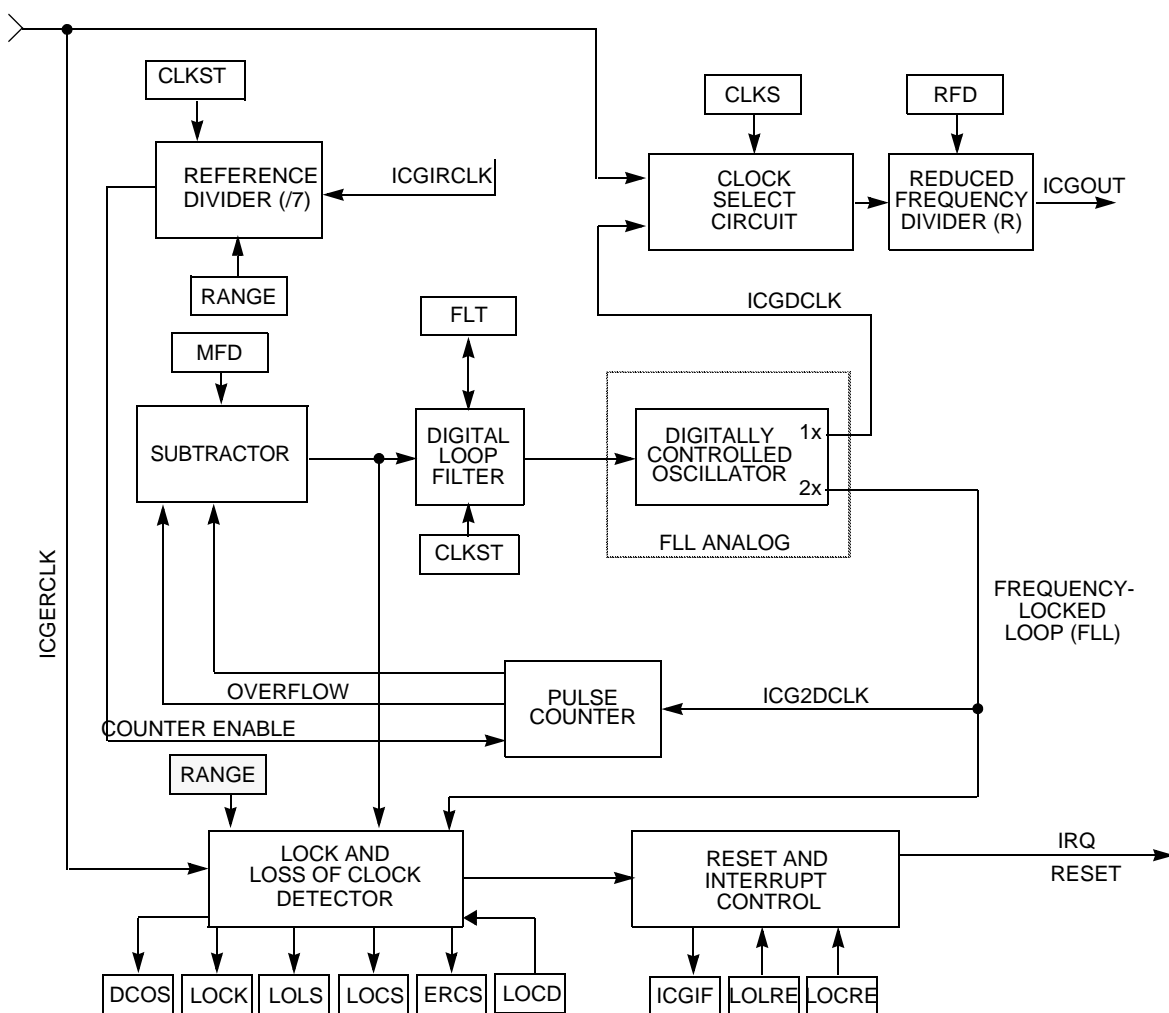


Figure 14-4. Detailed Frequency-Locked Loop Block Diagram

### 14.3.3 FLL Engaged, Internal Clock (FEI) Mode

FLL engaged internal (FEI) is entered when any of the following conditions occur:

- CLKS bits are written to 01.
- The DCO clock stabilizes ( $DCOS = 1$ ) while in SCM upon exiting the off state with  $CLKS = 01$

In FLL Engaged Internal Mode, the reference clock is derived from the internal reference clock ICGIRCLK, and the FLL loop will attempt to lock the ICGDCLK frequency to the desired value, as selected by the MFD bits.

#### 14.3.3.1 FLL Engaged Internal Unlocked

FEI unlocked is a temporary state that is entered when FEI is entered and the count error ( $\Delta n$ ) output from the subtractor is greater than the maximum  $n_{unlock}$  or less than the minimum  $n_{unlock}$ , as required by the lock detector to detect the unlock condition.

The remains in this state while the count error ( $\Delta n$ ) is greater than the maximum  $n_{lock}$  or less than the minimum  $n_{lock}$ , as required by the lock detector to detect the lock condition.

In this state the output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / R$ .

#### 14.3.3.2 FLL Engaged Internal Locked

FLL engaged internal locked is entered from FEI unlocked when the count error ( $\Delta n$ ), which comes from the subtractor, is less than  $n_{lock}$  (max) and greater than  $n_{lock}$  (min) for a given number of samples, as required by the lock detector to detect the lock condition. The output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / R$ . In FEI locked, the filter value is only updated once every four comparison cycles. The update made is an average of the error measurements taken in the four previous comparisons.

### 14.3.4 FLL Bypassed, External Clock (FBE) Mode

FLL bypassed external (FBE) is entered when any of the following conditions occur:

- From SCM when  $CLKS = 10$  and ERCS is high
- When  $CLKS = 10$ , ERCS = 1 upon entering Off Mode, and Off is then exited
- From FLL Engaged External Mode if a loss of DCO clock occurs and the external reference is still valid (both LOCS = 1 and ERCS = 1)

In this state, the DCO and IRG are off and the reference clock is derived from the external reference clock, ICGERCLK. The output clock signal ICGOUT frequency is given by  $f_{ICGERCLK} / R$ . If an external clock source is used ( $REFS = 0$ ), then the input frequency on the EXTAL pin can be anywhere in the range 0 MHz to 40 MHz. If a crystal or resonator is used ( $REFS = 1$ ), then frequency range is either low for RANGE = 0 or high for RANGE = 1.

### 14.3.5 FLL Engaged, External Clock (FEE) Mode

The FLL Engaged External (FEE) Mode is entered when any of the following conditions occur:

- $CLKS = 11$  and ERCS and DCOS are both high.
- The DCO stabilizes ( $DCOS = 1$ ) while in SCM upon exiting the off state with  $CLKS = 11$ .

In FEE Mode, the reference clock is derived from the external reference clock ICGERCLK, and the FLL loop will attempt to lock the ICGDCLK frequency to the desired value, as selected by the MFD bits. To run in FEE Mode, there must be a working 32 kHz–100 kHz or 2 MHz–10 MHz external clock source. The maximum external clock frequency is limited to 10 MHz in FEE Mode to prevent over-clocking the DCO. The minimum multiplier for the FLL, from Table 14-7, is 4. Because  $4 \times 10$  MHz is 40 MHz, which is the operational limit of the DCO, the reference clock cannot be any faster than 10 MHz.

#### 14.3.5.1 FLL Engaged External Unlocked

FEE unlocked is entered when FEE is entered and the count error ( $\Delta n$ ) output from the subtractor is greater than the maximum  $n_{unlock}$  or less than the minimum  $n_{unlock}$ , as required by the lock detector to detect the unlock condition.

The will remain in this state while the count error ( $\Delta n$ ) is greater than the maximum  $n_{lock}$  or less than the minimum  $n_{lock}$ , as required by the lock detector to detect the lock condition.

In this state, the pulse counter, subtractor, digital loop filter, and DCO form a closed loop and attempt to lock it according to their operational descriptions later in this section. Upon entering this state and until the FLL becomes locked, the output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / (2 \times R)$ . This extra divide by two prevents frequency overshoots during the initial locking process from exceeding chip-level maximum frequency specifications. As soon as the FLL has locked, if an unexpected loss of lock causes it to re-enter the unlocked state while the remains in FEE Mode, the output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / R$ .

#### 14.3.5.2 FLL Engaged External Locked

FEE locked is entered from FEE unlocked when the count error ( $\Delta n$ ) is less than  $n_{lock} (max)$  and greater than  $n_{lock} (min)$  for a given number of samples, as required by the lock detector to detect the lock condition. The output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / R$ . In FLL engaged external locked, the filter value is only updated once every four comparison cycles. The update made is an average of the error measurements taken in the four previous comparisons.

### 14.3.6 FLL Lock and Loss-of-Lock Detection

To determine the FLL locked and loss-of-lock conditions, the pulse counter counts the pulses of the DCO for one comparison cycle (see Table 14-2 for explanation of a comparison cycle) and passes this number to the subtractor. The subtractor compares this value to the value in MFD and produces a count error,  $\Delta n$ . To achieve locked status,  $\Delta n$  must be between  $n_{lock} (min)$  and  $n_{lock} (max)$ . As soon as the FLL has locked,  $\Delta n$  must stay between  $n_{unlock} (min)$  and  $n_{unlock} (max)$  to remain locked. If  $\Delta n$  goes outside this range unexpectedly, the LOLS status bit is set and remains set until acknowledged or until the MCU is reset.

LOLS is cleared by reading ICGS1 then writing 1 to ICGIF (LOLRE = 0), or by a loss-of-lock induced reset (LOLRE = 1), or by any MCU reset.

If the ICG enters the off state due to Stop Mode when ENBDM = OSCSTEN = 0, the FLL loses locked status (LOCK is cleared), but LOLS remains unchanged because this is not an unexpected loss-of-lock condition. Though it would be unusual, if ENBDM is cleared to 0 while the MCU is in stop, the ICG enters the off state. Because this is an unexpected stopping of clocks, LOLS will be set when the MCU wakes up from stop.

Expected loss of lock occurs when the MFD or CLKS bits are changed or in FEI Mode only, when the TRIM bits are changed. In these cases, the LOCK bit will be cleared until the FLL regains lock, but the LOLS will not be set.

### 14.3.7 FLL Loss-of-Clock Detection

The reference clock and the DCO clock are monitored under different conditions (see [Table 14-1](#)). Provided the reference frequency is being monitored, ERCS = 1 indicates that the reference clock meets minimum frequency requirements. When the reference and/or DCO clock(s) are being monitored, if either one falls below a certain frequency,  $f_{LOR}$  and  $f_{LOD}$ , respectively, the LOCS status bit will be set to indicate the error. LOCS will remain set until it is cleared by software or until the MCU is reset. LOCS is cleared by reading ICGS1 then writing 1 to ICGIF (LOCRES = 0), or by a loss-of-clock induced reset (LOCRES = 1), or by any MCU reset.

If the ICG is in FEE, a loss of reference clock causes the ICG to enter SCM, and a loss of DCO clock causes the ICG to enter FBE Mode. If the ICG is in FBE Mode, a loss of reference clock will cause the ICG to enter SCM. In each case, the CLKST and CLKS bits will be automatically changed to reflect the new state.

A loss of clock will also cause a loss of lock when in FEE or FEI Modes. Because the method of clearing the LOCS and LOLS bits is the same, this would only be an issue in the unlikely case that LOLRE = 1 and LOCRES = 0. In this case, the interrupt would be overridden by the reset for the loss of lock.

**Table 14-1. Clock Monitoring (When LOCD = 0)**

Mode	CLKS	REFST	ERCS	External Reference Clock Monitored?	DCO Clock Monitored?
Off	0X or 11	X	Forced Low	No	No
	10	0	Forced Low	No	No
	10	1	Real-Time <sup>1</sup>	Yes <sup>(1)</sup>	No
SCM (CLKST = 00)	0X	X	Forced Low	No	Yes <sup>2</sup>
	10	0	Forced High	No	Yes <sup>(2)</sup>
	10	1	Real-Time	Yes	Yes <sup>(2)</sup>
	11	X	Real-Time	Yes	Yes <sup>(2)</sup>
FEI (CLKST = 01)	0X	X	Forced Low	No	Yes
	11	X	Real-Time	Yes	Yes

**Table 14-1. Clock Monitoring (When LOCD = 0) (continued)**

Mode	CLKS	REFST	ERCS	External Reference Clock Monitored?	DCO Clock Monitored?
FBE (CLKST = 10)	10	0	Forced High	No	No
	10	1	Real-Time	Yes	No
FEE (CLKST = 11)	11	X	Real-Time	Yes	Yes

<sup>1</sup> If ENABLE is high (waiting for external crystal start-up after exiting stop).

<sup>2</sup> DCO clock will not be monitored until DCOS = 1 upon entering SCM from off or FLL Bypassed External Mode.

### 14.3.8 Clock Mode Requirements

A clock mode is requested by writing to CLKS1:CLKS0 and the actual clock mode is indicated by CLKST1:CLKST0. Provided minimum conditions are met, the status shown in CLKST1:CLKST0 should be the same as the requested mode in CLKS1:CLKS0. Table 14-2 shows the relationship between CLKS, CLKST, and ICGOUT. It also shows the conditions for CLKS = CLKST or the reason CLKS ≠ CLKST.

#### NOTE

If a crystal will be used before the next reset, then be sure to set REFS = 1 and CLKS = 1x on the first write to the ICGC1 register. Failure to do so will result in “locking” REFS = 0, which will prevent the oscillator amplifier from being enabled until the next reset occurs.

**Table 14-2. ICG State Table**

Actual Mode (CLKST)	Desired Mode (CLKS)	Range	Reference Frequency ( $f_{\text{REFERENCE}}$ )	Comparison Cycle Time	ICGOUT	Conditions <sup>1</sup> for CLKS = CLKST	Reason CLKS1 = CLKST
Off (XX)	Off (XX)	X	0	—	0	—	—
	FBE (10)	X	0	—	0	—	ERCS = 0
SCM (00)	SCM (00)	X	$f_{\text{ICGIRCLK}}/7^2$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	Not switching from FBE to SCM	—
	FEI (01)	0	$f_{\text{ICGIRCLK}}/7^{(1)}$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	DCOS = 0
	FBE (10)	X	$f_{\text{ICGIRCLK}}/7^{(1)}$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	ERCS = 0
	FEE (11)	X	$f_{\text{ICGIRCLK}}/7^{(1)}$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	DCOS = 0 or ERCS = 0
FEI (01)	FEI (01)	0	$f_{\text{ICGIRCLK}}/7$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	DCOS = 1	—
	FEE (11)	X	$f_{\text{ICGIRCLK}}/7$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	ERCS = 0

**Table 14-2. ICG State Table (continued)**

Actual Mode (CLKST)	Desired Mode (CLKS)	Range	Reference Frequency ( $f_{\text{REFERENCE}}$ )	Comparison Cycle Time	ICGOUT	Conditions <sup>1</sup> for CLKS = CLKST	Reason CLKS1 = CLKST
FBE (10)	FBE (10)	X	0	—	ICGERCLK/R	ERCS = 1	—
	FEE (11)	X	0	—	ICGERCLK/R	—	LOCS = 1 & ERCS = 1
FEE (11)	FEE (11)	0	$f_{\text{ICGERCLK}}$	$2/f_{\text{ICGERCLK}}$	ICGDCLK/R <sup>3</sup>	ERCS = 1 and DCOS = 1	—
		1	$f_{\text{ICGERCLK}}$	$128/f_{\text{ICGERCLK}}$	ICGDCLK/R <sup>(2)</sup>	ERCS = 1 and DCOS = 1	—

<sup>1</sup> CLKST will not update immediately after a write to CLKS. Several bus cycles are required before CLKST updates to the new value.

<sup>2</sup> The reference frequency has no effect on ICGOUT in SCM, but the reference frequency is still used in making the comparisons that determine the DCOS bit.

<sup>3</sup> After initial LOCK; will be ICGDCLK/2R during initial locking process and while FLL is re-locking after the MFD bits are changed.

### 14.3.9 Fixed Frequency Clock

The ICG provides a fixed frequency clock output, XCLK, for use by on-chip peripherals. This output is equal to the internal bus clock, BUSCLK, in FBE Mode. In FEE Mode, XCLK is equal to  $\text{ICGERCLK} \div 2$  when the following conditions are met:

- $(P \times N) \div R \geq 4$  where P is determined by RANGE (see [Table 14-4](#)), N and R are determined by MFD and RFD, respectively (see [Table 14-5](#)).
- LOCK = 1.

If the above conditions are not true, then XCLK is equal to BUSCLK.

When the ICG is in either FEI or SCM Mode, XCLK is turned off. Any peripherals which can use XCLK as a clock source must not do so when the ICG is in FEI or SCM Mode.

### 14.3.10 High Gain Oscillator

The oscillator has the option of running in a high gain oscillator (HGO) Mode, which improves the oscillator's resistance to EMC noise when running in FBE or FEE Modes. This option is selected by writing a 1 to the HGO bit in the ICGC1 register. HGO is used with both the high and low range oscillators but is only valid when REFS = 1 in the ICGC1 register. When HGO = 0, the standard low-power oscillator is selected.

If the high gain option is to be switched after the initial write to the ICGC1 register, then the ICG should first be changed to SCM or FEI Mode to stop the external oscillator. Then the HGO bit can be modified and FEE or FBE Mode can be re-selected in the same write to ICGC1. The oscillator will go through the standard start-up delay before the ICG switches to the external oscillator.

## 14.4 Initialization/Application Information

### 14.4.1 Introduction

This section is intended to give some basic direction on which configuration a user would want to select when initializing the ICG. For some applications, the serial communication link may dictate the accuracy of the clock reference. For other applications, lowest power consumption may be the chief clock consideration. Still others may have lowest cost as the primary goal. The ICG allows great flexibility in choosing which is best for any application.

**Table 14-3. ICG Configuration Consideration**

	Clock Reference Source = Internal	Clock Reference Source = External
<b>FLL Engaged</b>	<b>FEI</b> 4 MHz < $f_{Bus}$ < 20 MHz. Medium power (will be less than FEE if oscillator range = high) Medium clock accuracy (After IRG is trimmed) <b>Lowest system cost</b> (no external components required) IRG is on. DCO is on. <sup>1</sup>	<b>FEE</b> 4 MHz < $f_{Bus}$ < 20 MHz Medium power (will be less than FEI if oscillator range = low) Good clock accuracy Medium/High system cost (crystal, resonator or external clock source required) IRG is off. DCO is on.
<b>FLL Bypassed</b>	<b>SCM</b> This mode is mainly provided for quick and reliable system startup. 3 MHz < $f_{Bus}$ < 5 MHz (default). 3 MHz < $f_{Bus}$ < 20 MHz (via filter bits). Medium power Poor accuracy. IRG is off. DCO is on and open loop.	<b>FBE</b> $f_{Bus}$ range <= 8 MHz when crystal or resonator is used. <b>Lowest power</b> Highest clock accuracy Medium/High system cost (Crystal, resonator or external clock source required) IRG is off. DCO is off.

<sup>1</sup> The IRG typically consumes 100  $\mu$ A. The FLL and DCO typically consumes 0.5 to 2.5 mA, depending upon output frequency. For minimum power consumption and minimum jitter, choose N and R to be as small as possible.

The following sections contain initialization examples for various configurations.

#### NOTE

Hexadecimal values designated by a preceding \$, binary values designated by a preceding percent, and decimal values have no preceding character.

Important configuration information is repeated here for reference.

**Table 14-4. ICGOUT Frequency Calculation Options**

Clock Scheme	$f_{ICGOUT}^1$	P	Note
SCM — Self-clocked Mode (FLL bypassed internal)	$f_{ICGDCLK} / R$	NA	Typical $f_{ICGOUT}$ = 8 MHz out of reset
FBE — FLL bypassed external	$f_{ext} / R$	NA	
FEI — FLL engaged internal	$(f_{IRG} / 7) * 64 * N / R$	64	Typical $f_{IRG}$ = 243 kHz
FEE — FLL engaged external	$f_{ext} * P * N / R$	Range = 0 ; P = 64 Range = 1; P = 1	



<sup>1</sup> Ensure that  $f_{ICGDCLK}$ , which is equal to  $f_{ICGOUT} * R$ , does not exceed  $f_{ICGDCLKmax}$ .

**Table 14-5. MFD and RFD Decode Table**

MFD Value	Multiplication Factor (N)	RFD	Division Factor (R)
000	4	000	÷1
001	6	001	÷2
010	8	010	÷4
011	10	011	÷8
100	12	100	÷16
101	14	101	÷32
110	16	110	÷64
111	18	111	÷128

Register	Bit 7	6	5	4	3	2	1	Bit 0
ICGC1	HGO	RANGE	REFS	CLKS		OSCSTEN	LOCD	0
ICGC2	LOLRE	MFD			LOCRE	RFD		
ICGS1	CLKST		REFST	LOLS	LOCK	LOCS	ERCS	ICGIF
ICGS2	0	0	0	0	0	0	0	DCOS
ICGFLTU	0	0	0	0	FLT			
ICGFLTL	FLT							
ICGTRM	TRIM							



= Unimplemented or Reserved

**Figure 14-5. Register Set**

### 14.4.2 Example #1: External Crystal = 32 kHz, Bus Frequency = 4.19 MHz

In this example, the FLL will be used (in FEE Mode) to multiply the external 32 kHz oscillator up to 8.38 MHz to achieve 4.19 MHz bus frequency.

After the MCU is released from reset, the ICG is in Self-clocked Mode (SCM) and supplies approximately 8 MHz on ICGOUT, which corresponds to a 4 MHz bus frequency ( $f_{Bus}$ ).

The clock scheme will be FLL engaged, external (FEE). So

$$f_{ICGOUT} = f_{ext} * P * N / R ; P = 64, f_{ext} = 32 \text{ kHz} \quad \text{Eqn. 14-1}$$

Solving for N / R gives:

$$N / R = 8.38 \text{ MHz} / (32 \text{ kHz} * 64) = 4 ; \text{ we can choose } N = 4 \text{ and } R = 1 \quad \text{Eqn. 14-2}$$

The values needed in each register to set up the desired operation are:

**ICGC1 = \$38 (%00111000)**

Bit 7 HGO 0 Configures oscillator for low-power operation

### MCU Internal Clock Generator (ICG)

Bit 6	RANGE	0	Configures oscillator for low-frequency range; FLL prescale factor is 64
Bit 5	REFS	1	Oscillator using crystal or resonator is requested
Bits 4:3	CLKS	11	FLL engaged, external reference clock mode
Bit 2	OSCSTEN	0	Oscillator disabled in stop modes
Bit 1	LOCD	0	Loss-of-clock detection enabled
Bit 0		0	Unimplemented or reserved, always reads zero

### ICGC2 = \$00 (%00000000)

Bit 7	LOLRE	0	Generates an interrupt request on loss of lock
Bits 6:4	MFD	000	Sets the MFD multiplication factor to 4
Bit 3	LOCRES	0	Generates an interrupt request on loss of clock
Bits 2:0	RFD	000	Sets the RFD division factor to ÷1

### ICGS1 = \$xx

This is read only except for clearing interrupt flag

### ICGS2 = \$xx

This is read only; should read DCOS = 1 before performing any time critical tasks

### ICGFLTLU/L = \$xx

Only needed in Self-clocked Mode; FLT will be adjusted by loop to give 8.38 MHz DCO clock

Bits 15:12	unused	0000
Bits 11:0	FLT	No need for user initialization

### ICGTRM = \$xx

Bits 7:0 TRIM Only need to write when trimming internal oscillator; not used when external crystal is clock source

Figure 14-6 shows flow charts for three conditions requiring ICG initialization.

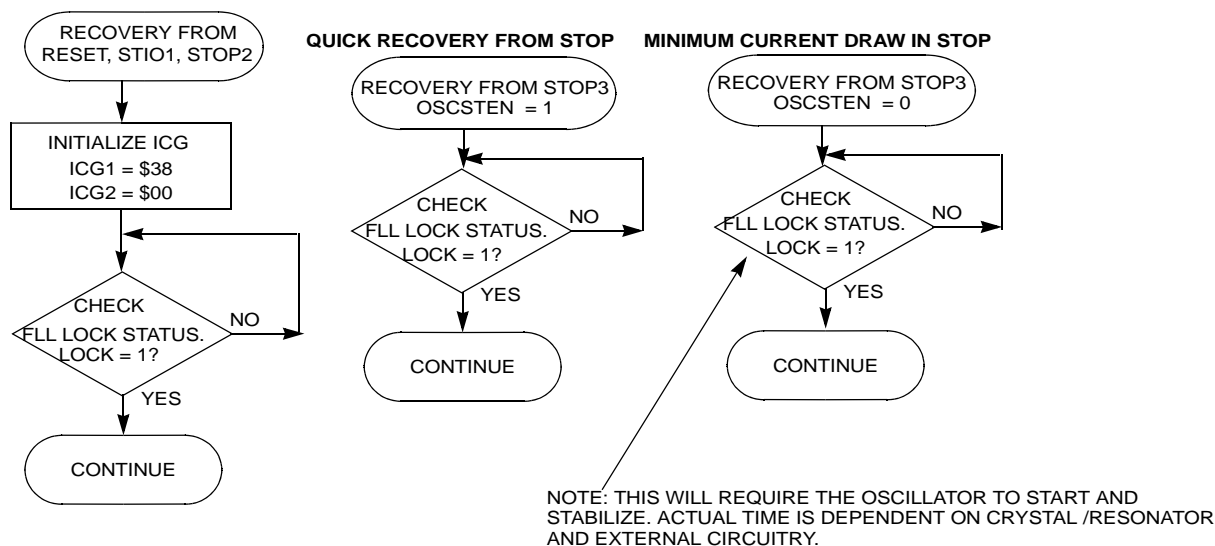


Figure 14-6. ICG Initialization for FEE in Example #1

### 14.4.3 Example #2: External Crystal = 4 MHz, Bus Frequency = 20 MHz

In this example, the FLL will be used (in FEE Mode) to multiply the external 4 MHz oscillator up to 40-MHz to achieve 20 MHz bus frequency.

After the MCU is released from reset, the ICG is in Self-clocked Mode (SCM) and supplies approximately 8 MHz on ICGOUT which corresponds to a 4 MHz bus frequency ( $f_{\text{Bus}}$ ).

During reset initialization software, the clock scheme will be set to FLL engaged, external (FEE). So

$$f_{\text{ICGOUT}} = f_{\text{ext}} * P * N / R ; P = 1, f_{\text{ext}} = 4.00 \text{ MHz} \quad \text{Eqn. 14-3}$$

Solving for N / R gives:

$$N / R = 40 \text{ MHz} / (4 \text{ MHz} * 1) = 10 ; \text{ We can choose } N = 10 \text{ and } R = 1 \quad \text{Eqn. 14-4}$$

The values needed in each register to set up the desired operation are:

**ICGC1 = \$78 (%01111000)**

Bit 7	HGO	0	Configures oscillator for low-power operation
Bit 6	RANGE	1	Configures oscillator for high-frequency range; FLL prescale factor is 1
Bit 5	REFS	1	Requests an oscillator
Bits 4:3	CLKS	11	FLL engaged, external reference clock mode
Bit 2	OSCSTEN	0	Disables the oscillator in stop modes
Bit 1	LOCD	0	Loss-of-clock detection enabled
Bit 0		0	Unimplemented or reserved, always reads zero

**ICGC2 = \$30 (%00110000)**

Bit 7	LOLRE	0	Generates an interrupt request on loss of lock
Bit 6:4	MFD	011	Sets the MFD multiplication factor to 10
Bit 3	LOCRE	0	Generates an interrupt request on loss of clock
Bit 2:0	RFD	000	Sets the RFD division factor to ÷1

**ICGS1 = \$xx**

This is read only except for clearing interrupt flag

**ICGS2 = \$xx**

This is read only. Should read DCOS before performing any time critical tasks

**ICGFLTLU/L = \$xx**

Not used in this example

**ICGTRM**

Not used in this example

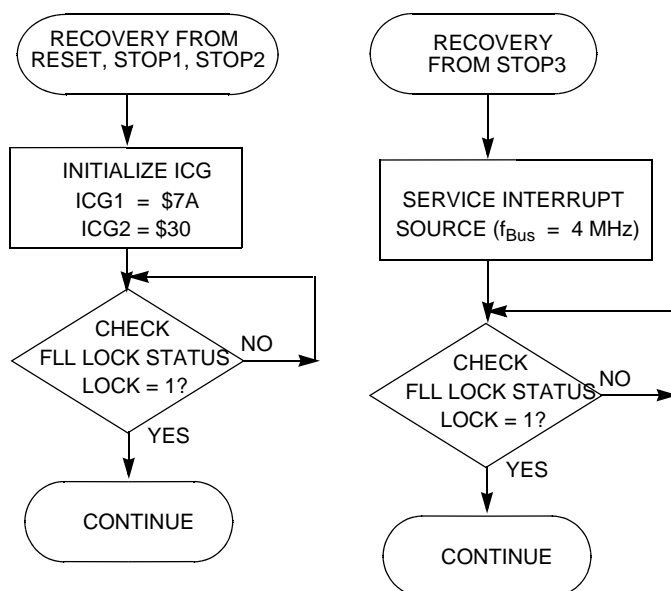


Figure 14-7. ICG Initialization and Stop Recovery for Example #2

### 14.4.4 Example #3: No External Crystal Connection, 5.4 MHz Bus Frequency

In this example, the FLL will be used (in FEI Mode) to multiply the internal 243 kHz (approximate) reference clock up to 10.8 MHz to achieve 5.4 MHz bus frequency. This system will also use the trim function to fine tune the frequency based on an external reference signal.

After the MCU is released from reset, the ICG is in Self-clocked Mode (SCM) and supplies approximately 8 MHz on ICGOUT which corresponds to a 4 MHz bus frequency ( $f_{Bus}$ ).

The clock scheme will be FLL engaged, internal (FEI). So

$$f_{ICGOUT} = (f_{IRG} / 7) * P * N / R ; P = 64, f_{IRG} = 243 \text{ kHz} \quad \text{Eqn. 14-5}$$

Solving for N / R gives:

$$N / R = 10.8 \text{ MHz} / (243/7 \text{ kHz} * 64) = 4.86 ; \text{ We can choose } N = 10 \text{ and } R = 2. \quad \text{Eqn. 14-6}$$

A trim procedure will be required to hone the frequency to exactly 5.4 MHz. An example of the trim procedure is shown in example #4.

The values needed in each register to set up the desired operation are:

**ICGC1 = \$28 (%00101000)**

Bit 7	HGO	0	Configures oscillator for low-power operation
Bit 6	RANGE	0	Configures oscillator for low-frequency range; FLL prescale factor is 64
Bit 5	REFS	1	Oscillator using crystal or resonator requested (bit is really a don't care)
Bits 4:3	CLKS	01	FLL engaged, internal reference clock mode
Bit 2	OSCSTEN	0	Disables the oscillator in stop modes
Bit 1	LOCD	0	Loss-of-clock detection enabled

Bit 0                    0      Unimplemented or reserved, always reads zero

**ICGC2 = \$31 (%00110001)**

Bit 7    LOLRE    0      Generates an interrupt request on loss of lock  
 Bit 6:4   MFD      011    Sets the MFD multiplication factor to 10  
 Bit 3    LOCRE    0      Generates an interrupt request on loss of clock  
 Bit 2:0   RFD      001    Sets the RFD division factor to ÷2

**ICGS1 = \$xx**

This is read only except for clearing interrupt flag

**ICGS2 = \$xx**

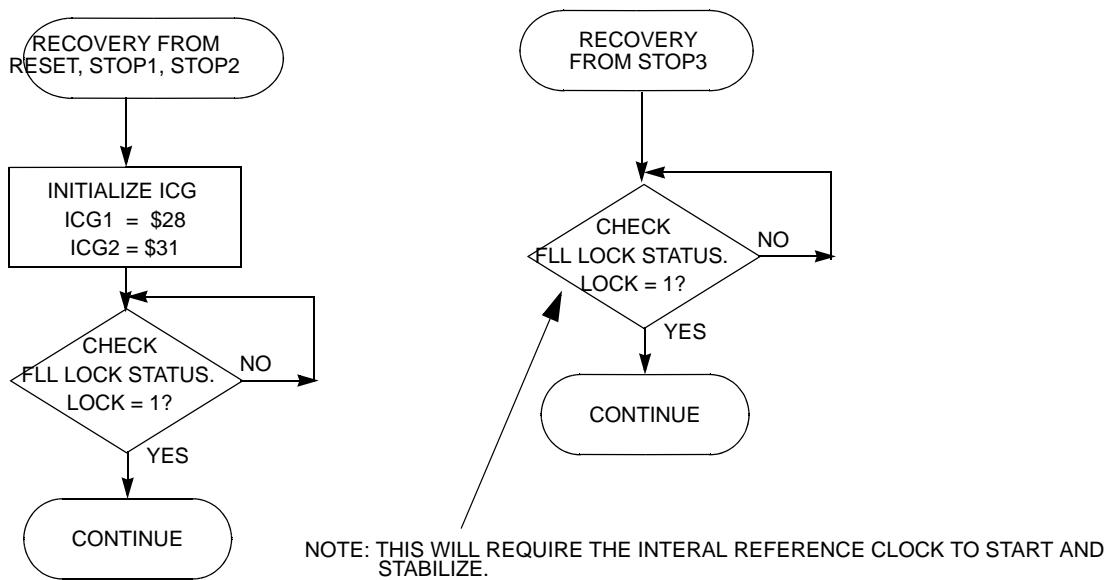
This is read only; good idea to read this before performing time critical operations

**ICGFLTLU/L = \$xx**

Not used in this example

**ICGTRM = \$xx**

Bit 7:0   TRIM                    Only need to write when trimming internal oscillator; done in separate operation (see example #4)



**Figure 14-8. ICG Initialization and Stop Recovery for Example #3**

### 14.4.5 Example #4: Internal Clock Generator Trim

The internally generated clock source is guaranteed to have a period  $\pm 25\%$  of the nominal value. In some case this may be sufficient accuracy. For other applications that require a tight frequency tolerance, a trimming procedure is provided that will allow a very accurate source. This section outlines one example of trimming the internal oscillator. Many other possible trimming procedures are valid and can be used.

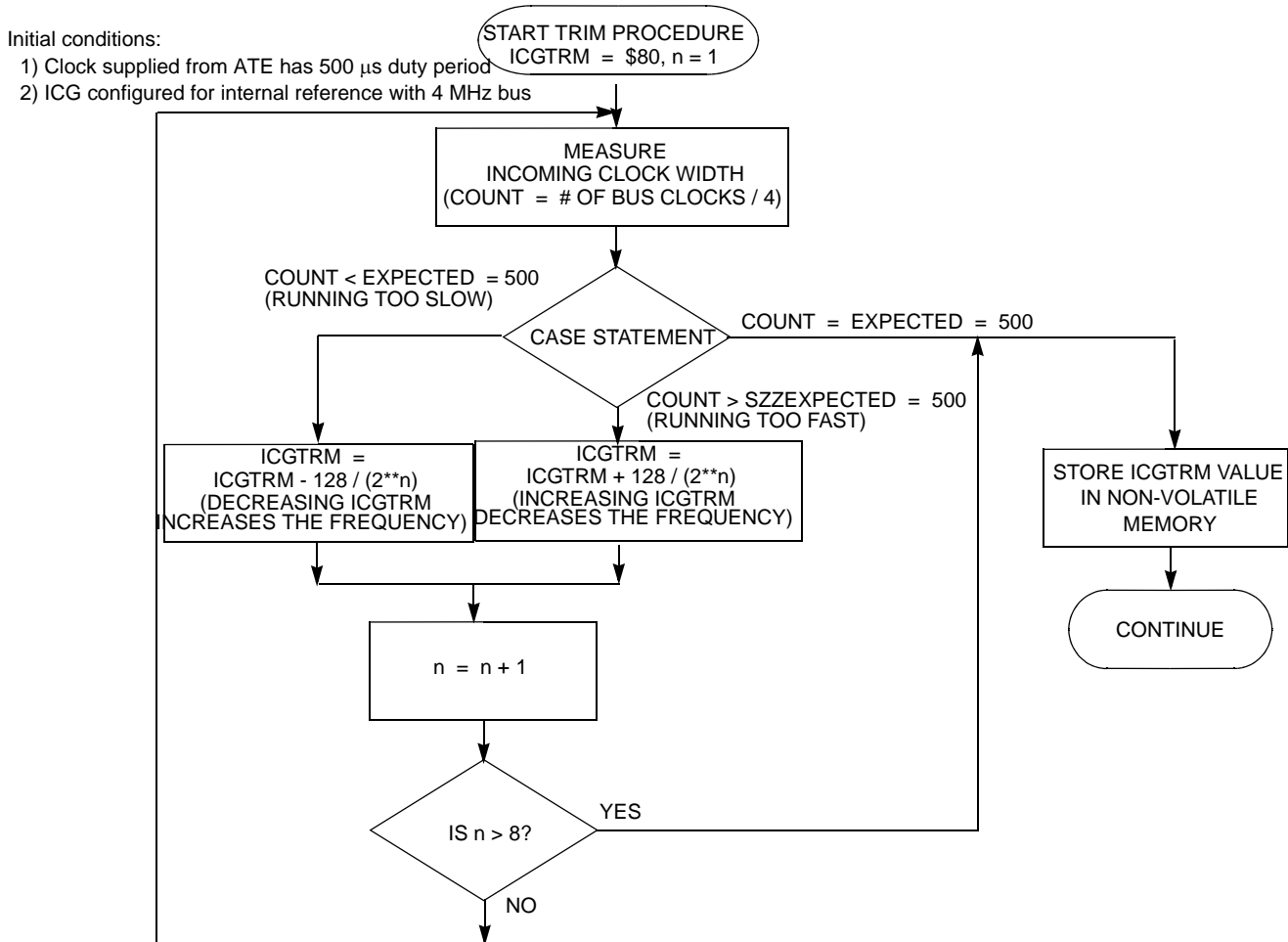


Figure 14-9. Trim Procedure

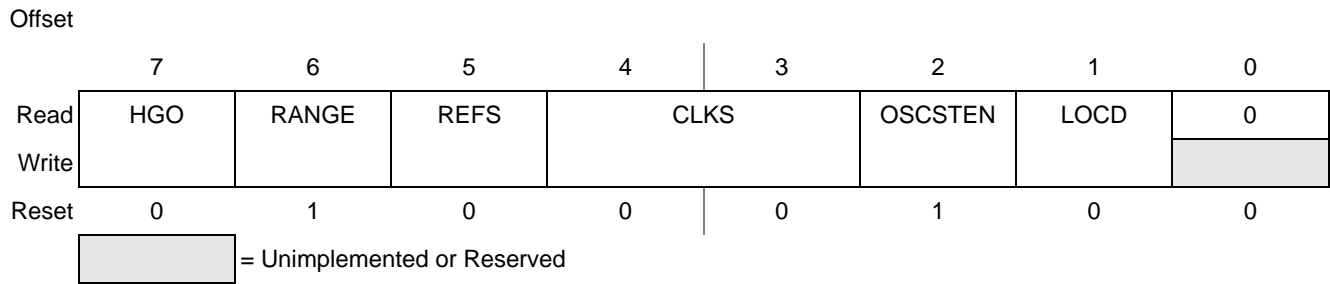
In this particular case, the MCU has been attached to a PCB and the entire assembly is undergoing final test with automated test equipment. A separate signal or message is provided to the MCU operating under user provided software control. The MCU initiates a trim procedure as outlined in Figure 14-9 while the tester supplies a precision reference signal.

If the intended bus frequency is near the maximum allowed for the device, it is recommended to trim using a reduction divisor (R) twice the final value. Once the trim procedure is complete, the reduction divisor can be restored. This will prevent accidental overshoot of the maximum clock frequency.

## 14.5 ICG Registers and Control Bits

Refer to the direct-page register summary in [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all ICG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 14.5.1 Control Register 1 (C1)



**Figure 14-10. Control Register 1 (C1)**

**Table 14-6. C1 Field Descriptions**

Field	Description
7 HGO	<b>High Gain Oscillator Select</b> — The HGO bit is used to select between low-power operation and high-amplitude operation. 0 Oscillator configured for low power operation. 1 Oscillator configured for high amplitude operation.
6 RANGE	<b>Frequency Range Select</b> — The RANGE bit controls the oscillator, reference divider, and FLL loop prescaler multiplication factor (P). It selects one of two reference frequency ranges for the . The RANGE bit is write-once after a reset. The RANGE bit only has an effect in FLL engaged external and FLL bypassed external modes. 0 Oscillator configured for low frequency range. FLL loop prescale factor P is 64. 1 Oscillator configured for high frequency range. FLL loop prescale factor P is 1.
5 REFS	<b>External Reference Select</b> — The REFS bit controls the external reference clock source for ICGERCLK. The REFS bit is write-once after a reset. 0 External clock requested. 1 Oscillator using crystal or resonator requested.
4:3 CLKS	<b>Clock Mode Select</b> — The CLKS bits control the clock mode. If FLL bypassed external is requested, it will not be selected until ERCS = 1. If the enters Off Mode, the CLKS bits will remain unchanged. Writes to the CLKS bits will not take effect if a previous write is not complete. The CLKS bits are writable at any time, unless the first write after a reset was CLKS = 0X, the CLKS bits cannot be written to 1X until after the next reset (because the EXTAL pin was not reserved). 00 Self-clocked 01 FLL engaged, internal reference 10 FLL bypassed, external reference 11 FLL engaged, external reference

**Table 14-6. C1 Field Descriptions (continued)**

Field	Description
2 OSCSTEN	<b>Enable Oscillator in Off Mode</b> — The OSCSTEN bit controls whether or not the oscillator circuit remains enabled when the ICG enters Off Mode. 0 Oscillator disabled when is in Off Mode unless ENABLE is high, CLKS = 10, and REFST = 1. 1 Oscillator enabled when is in Off Mode, CLKS = 1X and REFST = 1.
1 LOCD	<b>Loss of Clock Disable</b> 0 Loss of clock detection enabled. 1 Loss of clock detection disabled.

**IRGSTEN — Enable Internal Reference Generator in Off Mode**

The IRGSTEN bit controls whether or not the internal reference generator (IRG) circuit remains enabled when the enters off mode. If STOP is low, the IRG will be disabled in off mode regardless of IRGSTEN.

1 = IRG enabled when is in off mode and CLKST is not equal to 10.

0 = IRG disabled when is in off mode.

## 14.5.2 Control Register 2 (C2)

Offset

	7	6	5	4	3	2	1	0
Read	LOLRE		MFD		LOCRE		RFD	
Write								
Reset	0	0	0	0	0	0	0	0

**Figure 14-11. Control Register 2 (C2)**

**Table 14-7. C2 Field Descriptions**

Field	Description
7 LOLRE	<b>Loss of Lock Reset Enable</b> — The LOLRE bit determines what type of request is made by the following a loss of lock indication. The LOLRE bit only has an effect when LOLS is set. 0 Generate an interrupt request on loss of lock. 1 Generate a reset request on loss of lock.
6:4 MFD	<b>Multiplication Factor</b> — The MFD bits control the programmable multiplication factor in the FLL loop. The value specified by the MFD bits establishes the multiplication factor (N) applied to the reference frequency. Writes to the MFD bits will not take effect if a previous write is not complete. Select a low enough value for N such that $f_{ICGCLK}$ does not exceed its maximum specified rating. 000 Multiplication Factor (N) = 4 001 Multiplication Factor (N) = 6 010 Multiplication Factor (N) = 8 011 Multiplication Factor (N) = 10 100 Multiplication Factor (N) = 12 101 Multiplication Factor (N) = 14 110 Multiplication Factor (N) = 16 111 Multiplication Factor (N) = 18



**Table 14-7. C2 Field Descriptions (continued)**

Field	Description
3 LOCRE	<b>Loss of Clock Reset Enable</b> — The LOCRE bit determines how the system handles a loss of clock condition. 0 Generate an interrupt request on loss of clock. 1 Generate a reset request on loss of clock.
2:0 RFD	<b>Reduced Frequency Divider</b> — The RFD bits control the value of the divider following the clock select circuitry. The value specified by the RFD bits establishes the division factor (R) applied to the selected output clock source. Writes to the RFD bits will not take effect if a previous write is not complete. 000 Division Factor (R) = 1 001 Division Factor (R) = 2 010 Division Factor (R) = 4 011 Division Factor (R) = 8 100 Division Factor (R) = 16 101 Division Factor (R) = 32 110 Division Factor (R) = 64 111 Division Factor (R) = 128

### 14.5.3 Status Register 1 (S1)

Offset

	7	6	5	4	3	2	1	0
Read	CLKST		REFST	LOLS	LOCK	LOCS	ERCS	IF
Write								1
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

**Figure 14-12. Status Register 1 (S1)**
**Table 14-8. S1 Field Descriptions**

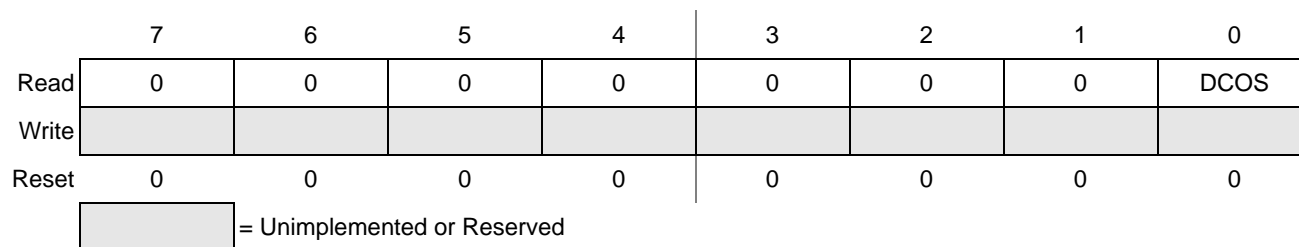
Field	Description
7:6 CLKST	<b>Clock Mode Status</b> — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Self-clocked 01 FLL engaged, internal reference 10 FLL bypassed, external reference 11 FLL engaged, external reference
5 REFST	<b>Reference Clock Status</b> — The REFST bit indicates which clock reference is currently selected by the Reference Select circuit. 0 External Clock selected. 1 Crystal/Resonator selected.
4 LOLS	<b>FLL Loss of Lock Status</b> — The LOLS bit is an indication of FLL-lock status. If LOLS is set, it remains set until cleared by clearing the ICGIF flag or an MCU reset. 0 FLL has not unexpectedly lost lock since LOLS was last cleared. 1 FLL has unexpectedly lost lock since LOLS was last cleared, LOLRE determines action taken.

**Table 14-8. S1 Field Descriptions (continued)**

Field	Description
3 LOCK	<b>FLL Lock Status</b> — The LOCK bit indicates whether the FLL has acquired lock. The LOCK bit is cleared in Off, Self-clocked, and FLL bypassed modes. 0 FLL is currently unlocked. 1 FLL is currently locked.
2 LOCS	<b>Loss Of Clock Status</b> — The LOCS bit is an indication of ICG loss-of-clock status. If LOCS is set, it remains set until cleared by clearing the ICGIF flag or an MCU reset. 0 has not lost clock since LOCS was last cleared. 1 has lost clock since LOCS was last cleared, LOCRE determines action taken.
1 ERCS	<b>External Reference Clock Status</b> — The ERCS bit is an indication of whether or not the external reference clock (ICGERCLK) meets the minimum frequency requirement. 0 External reference clock is not stable, frequency requirement is not met. 1 External reference clock is stable, frequency requirement is met.
0 IF	<b>Interrupt Flag</b> — The ICGIF read/write flag is set when an interrupt request is pending. It is cleared by a reset or by reading the ICG status register when ICGIF is set and then writing a 1 to ICGIF. If another ICG interrupt occurs before the clearing sequence is complete, the sequence is reset so ICGIF would remain set after the clear sequence was completed for the earlier interrupt. Writing a 0 to ICGIF has no effect. 0 No interrupt request is pending. 1 An interrupt request is pending.

### 14.5.4 Status Register 2 (S2)

Offset



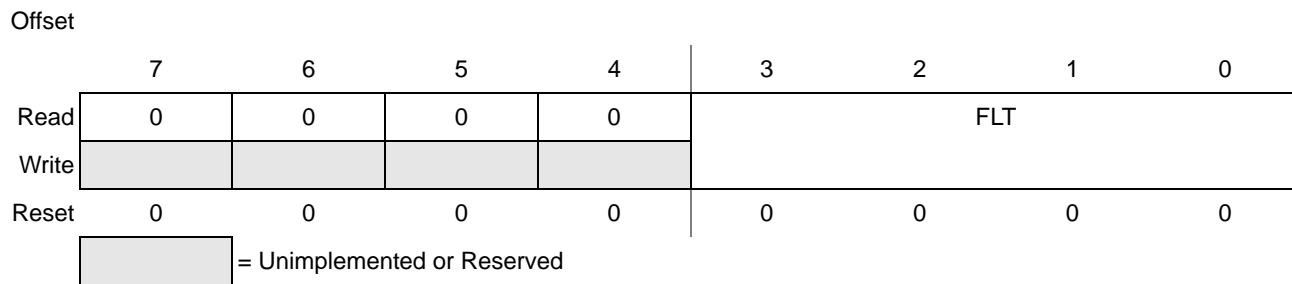
**Figure 14-13. Status Register 2 (S2)**

**Table 14-9. S2 Field Descriptions**

Field	Description
0 DCOS	<b>DCO Clock Stable</b> — The DCOS bit is set when the DCO clock (ICG2DCLK) is stable, meaning the count error has not changed by more than $n_{unlock}$ for two consecutive samples and the DCO clock is not static. This bit is used when exiting off state if $CLKS = X1$ to determine when to switch to the requested clock mode. It is also used in Self-clocked Mode to determine when to start monitoring the DCO clock. This bit is cleared upon entering the off state. 0 DCO clock is unstable. 1 DCO clock is stable.

## 14.5.5 Filter Registers (FLTU, ICGFLTL)

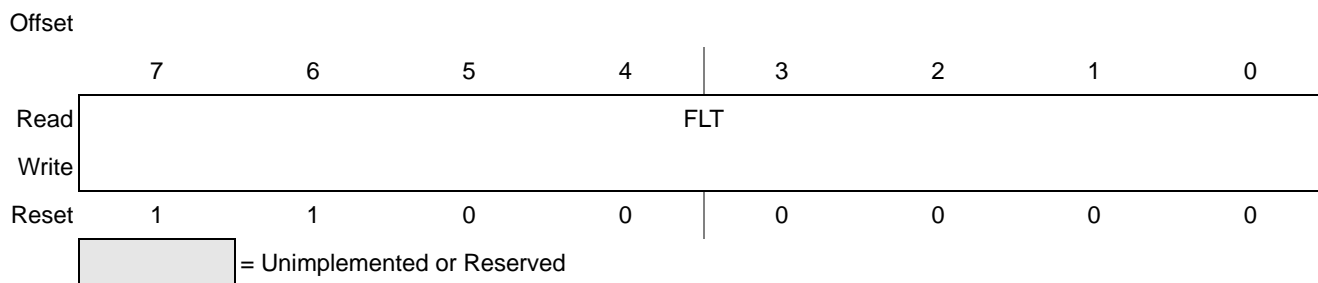
The filter registers show the filter value (FLT).



**Figure 14-14. Upper Filter Register (FLTU)**

**Table 14-10. FLTU Field Descriptions**

Field	Description
3:0 FLT	<b>Filter Value</b> — The FLT bits indicate the current filter value, which controls the DCO frequency. The FLT bits are read only except when the CLKS bits are programmed to Self-clocked Mode (CLKS = 00). In Self-clocked Mode, any write to ICGFLTU updates the current 12-bit filter value. Writes to the ICGFLTU register will not affect FLT if a previous latch sequence is not complete.



**Figure 14-15. Upper Filter Register (FLTL)**

**Table 14-11. FLTL Field Descriptions**

Field	Description
7:0 FLT	<b>Filter Value</b> — The FLT bits indicate the current filter value, which controls the DCO frequency. The FLT bits are read only except when the CLKS bits are programmed to Self-clocked Mode (CLKS = 00). In Self-clocked Mode, any write to ICGFLTU updates the current 12-bit filter value. Writes to the ICGFLTU register will not affect FLT if a previous latch sequence is not complete.

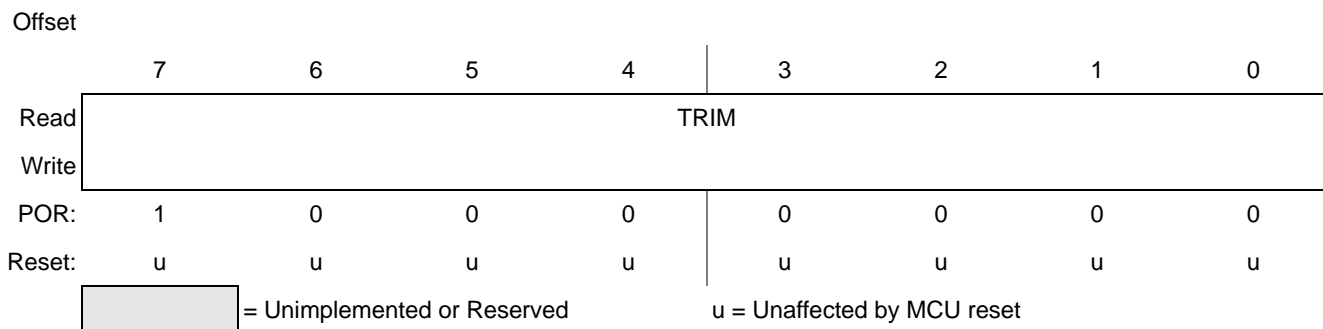
### CNT — Counter Value

The CNT bits indicate a previous counter value, which is the output of the pulse counter. The CNT bits are read only.

### ERR — Error Value

The ERR bits indicate a previous error value, which is the output of the subtractor. The ERR bits are read only.

## 14.5.6 Trim Register (TRM)



**Figure 14-16. ICG Trim Register (TRM)**

**Table 14-12. TRM Field Descriptions**

Field	Description
7:0 TRIM	<b>ICG Trim Setting</b> — The TRIM bits control the internal reference generator frequency. They allow a $\pm 25\%$ adjustment of the nominal (POR) period. The bit's effect on period is binary weighted (i.e., bit 1 will adjust twice as much as changing bit 0). Increasing the binary value in TRIM will increase the period and decreasing the value will decrease the period.

# Chapter 15

## MCU Central Processor Unit (CPU)

### 15.1 Introduction

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

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

### 15.2 Features

Features of the HCS08 CPU include:

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

## 15.3 Programmer's Model and CPU Registers

Figure 15-1 shows the five CPU registers. CPU registers are not part of the memory map.

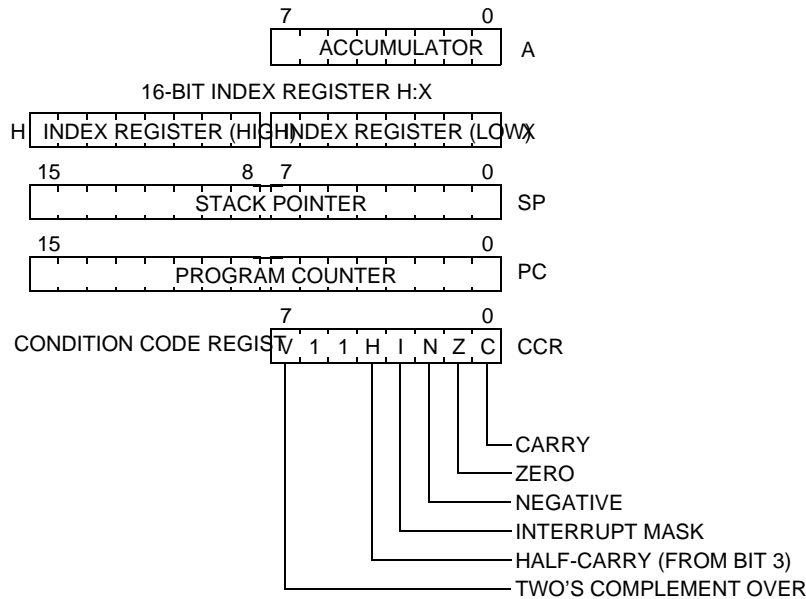


Figure 15-1. CPU Registers

### 15.3.1 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.

### 15.3.2 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.

### 15.3.3 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.

### 15.3.4 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 \$FFFE and \$FFFF. The vector stored there is the address of the first instruction that will be executed after exiting the reset state.

### 15.3.5 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/D.

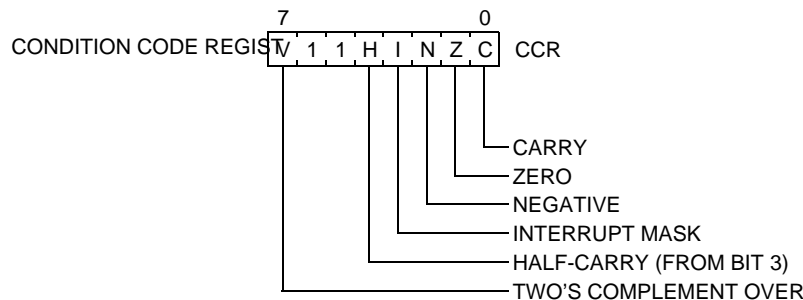


Figure 15-2. Condition Code Register

**Table 15-1. 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

## 15.4 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, all memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte linear address space so a 16-bit binary address can uniquely identify any memory location. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location



of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

### 15.4.1 Inherent Addressing Mode (INH)

In Inherent 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.

### 15.4.2 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.

### 15.4.3 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.

### 15.4.4 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.

### 15.4.5 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).

### 15.4.6 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.

### 15.4.6.1 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.

### 15.4.6.2 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.

### 15.4.6.3 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.

### 15.4.6.4 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.

### 15.4.6.5 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.

### 15.4.6.6 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.

### 15.4.6.7 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.

## 15.5 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.

## 15.5.1 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 [Chapter 12, “MCU Resets, Interrupts, and System Configuration”](#).

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.

## 15.5.2 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 users 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.

### 15.5.3 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.

### 15.5.4 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 [Chapter 10, “MCU Modes of Operation”](#) for more details.

### 15.5.5 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.

## 15.6 HCS08 Instruction Set Summary

The instruction set summary nomenclature listed here is used in the instruction descriptions in [Table 15-2](#).

### Operators

- ( ) = Contents of register or memory location shown inside parentheses
- ← = Is loaded with (read: “gets”)
- & = Boolean AND
- | = Boolean OR
- ⊕ = Boolean exclusive-OR
- × = Multiply
- ÷ = Divide
- :
- + = Add
- = Negate (two’s complement)

### CPU registers

- A = Accumulator
- CCR = Condition code register
- H = Index register, higher order (most significant) 8 bits
- X = Index register, lower order (least significant) 8 bits
- PC = Program counter
- PCH = Program counter, higher order (most significant) 8 bits
- PCL = Program counter, lower order (least significant) 8 bits
- SP = Stack pointer

### Memory and addressing

- M = A memory location or absolute data, depending on addressing mode
- M:M + 0x0001 = A 16-bit value in two consecutive memory locations. The higher-order (most significant) 8 bits are located at the address of M, and the lower-order (least significant) 8 bits are located at the next higher sequential address.

### Condition code register (CCR) bits

- V = Two’s complement overflow indicator, bit 7
- H = Half carry, bit 4
- I = Interrupt mask, bit 3
- N = Negative indicator, bit 2
- Z = Zero indicator, bit 1
- C = Carry/borrow, bit 0 (carry out of bit 7)

### CCR activity notation

- = Bit not affected
- 0 = Bit forced to 0

- 1 = Bit forced to 1
- P = Bit set or cleared according to results of operation
- U = Undefined after the operation

### Machine coding notation

- dd = Low-order 8 bits of a direct address 0x0000–0x00FF (high byte assumed to be 0x00)
- ee = Upper 8 bits of 16-bit offset
- ff = Lower 8 bits of 16-bit offset or 8-bit offset
- ii = One byte of immediate data
- jj = High-order byte of a 16-bit immediate data value
- kk = Low-order byte of a 16-bit immediate data value
- hh = High-order byte of 16-bit extended address
- ll = Low-order byte of 16-bit extended address
- rr = Relative offset

### Source form

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

- n* — Any label or expression that evaluates to a single integer in the range 0–7
- 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 value. The instruction treats this 8-bit value as the low order 8 bits of an address in the direct page of the 64-Kbyte address space (0x00xx).
- opr16a* — Any label or expression that evaluates to a 16-bit value. The instruction treats this value as an address in the 64-Kbyte address space.
- 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. Because the HCS08 has a 16-bit address bus, this can be either a signed or an unsigned value.
- rel* — Any label or expression that refers to an address that is within –128 to +127 locations from the next address after the last byte of object code for the current instruction. The assembler will calculate the 8-bit signed offset and include it in the object code for this instruction.

### Address modes

- INH = Inherent (no operands)
- IMM = 8-bit or 16-bit immediate
- DIR = 8-bit direct
- EXT = 16-bit extended
- IX = 16-bit indexed no offset

- IX+ = 16-bit indexed no offset, post increment (CBEQ and MOV only)  
 IX1 = 16-bit indexed with 8-bit offset from H:X  
 IX1+ = 16-bit indexed with 8-bit offset, post increment (CBEQ only)  
 IX2 = 16-bit indexed with 16-bit offset from H:X  
 REL = 8-bit relative offset  
 SP1 = Stack pointer with 8-bit offset  
 SP2 = Stack pointer with 16-bit offset

Table 15-2. HCS08 Instruction Set Summary (Sheet 1 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	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	$A \leftarrow (A) + (M) + (C)$	p	p		p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 ee ff E9 ff F9 9ED9 ee ff 9EE9 ff	2 3 4 4 3 3 5 4	
ADD #opr8i ADD opr8a ADD opr16a ADD oprx16,X ADD oprx8,X ADD ,X ADD oprx16,SP ADD oprx8,SP	Add without Carry	$A \leftarrow (A) + (M)$	p	p	-	p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB ii BB dd CB hh ll DB ee ff EB ff FB 9EDB ee ff 9EEB ff	2 3 4 4 4 3 5 4	
AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer	$SP \leftarrow (SP) + (M)$ M is sign extended to a 16-bit value	-	-	-	-	-	-	IMM	A7 ii	2	
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X)	$H:X \leftarrow (H:X) + (M)$ M is sign extended to a 16-bit value	-	-	-	-	-	-	IMM	AF ii	2	
AND #opr8i AND opr8a AND opr16a AND oprx16,X AND oprx8,X AND ,X AND oprx16,SP AND oprx8,SP	Logical AND	$A \leftarrow (A) \& (M)$	0	-	-	p	p	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 ii B4 dd C4 hh ll D4 ee ff E4 ff F4 9ED4 ee ff 9EE4 ff	2 3 4 4 3 5 4	
ASL opr8a ASLA ASLX ASL oprx8,X ASL ,X ASL oprx8,SP	Arithmetic Shift Left (Same as LSL)		p	-	-	p	p	p	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E68 ff	5 1 1 5 4 6	
ASR opr8a ASRA ASRX ASR oprx8,X ASR ,X ASR oprx8,SP	Arithmetic Shift Right		p	-	-	p	p	p	DIR INH INH IX1 IX SP1	37 dd 47 57 67 ff 77 9E67 ff	5 1 1 5 4 6	
BCC rel	Branch if Carry Bit Clear	Branch if (C) = 0	-	-	-	-	-	-	REL	24 rr	3	

Table 15-2. HCS08 Instruction Set Summary (Sheet 2 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
BCLR <i>n,opr8a</i>	Clear Bit n in Memory	$M_n \leftarrow 0$	-	-	-	-	-	-	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS <i>rel</i>	Branch if Carry Bit Set (Same as BLO)	Branch if (C) = 1	-	-	-	-	-	-	REL	25	rr	3
BEQ <i>rel</i>	Branch if Equal	Branch if (Z) = 1	-	-	-	-	-	-	REL	27	rr	3
BGE <i>rel</i>	Branch if Greater Than or Equal To (Signed Operands)	Branch if $(N \oplus V) = 0$	-	-	-	-	-	-	REL	90	rr	3
BGND	Enter Active Background if ENBDM = 1	Waits For and Processes BDM Commands Until GO, TRACE1, or TAGGO	-	-	-	-	-	-	INH	82		5+
BGT <i>rel</i>	Branch if Greater Than (Signed Operands)	Branch if (Z)   $(N \oplus V) = 0$	-	-	-	-	-	-	REL	92	rr	3
BHCC <i>rel</i>	Branch if Half Carry Bit Clear	Branch if (H) = 0	-	-	-	-	-	-	REL	28	rr	3
BHCS <i>rel</i>	Branch if Half Carry Bit Set	Branch if (H) = 1	-	-	-	-	-	-	REL	29	rr	3
BHI <i>rel</i>	Branch if Higher	Branch if (C)   (Z) = 0	-	-	-	-	-	-	REL	22	rr	3
BHS <i>rel</i>	Branch if Higher or Same (Same as BCC)	Branch if (C) = 0	-	-	-	-	-	-	REL	24	rr	3
BIH <i>rel</i>	Branch if IRQ Pin High	Branch if IRQ pin = 1	-	-	-	-	-	-	REL	2F	rr	3
BIL <i>rel</i>	Branch if IRQ Pin Low	Branch if IRQ pin = 0	-	-	-	-	-	-	REL	2E	rr	3
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)	0	-	-	b	b	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
BLE <i>rel</i>	Branch if Less Than or Equal To (Signed Operands)	Branch if (Z)   $(N \oplus V) = 1$	-	-	-	-	-	-	REL	93	rr	3
BLO <i>rel</i>	Branch if Lower (Same as BCS)	Branch if (C) = 1	-	-	-	-	-	-	REL	25	rr	3
BLS <i>rel</i>	Branch if Lower or Same	Branch if (C)   (Z) = 1	-	-	-	-	-	-	REL	23	rr	3
BLT <i>rel</i>	Branch if Less Than (Signed Operands)	Branch if $(N \oplus V) = 1$	-	-	-	-	-	-	REL	91	rr	3
BMC <i>rel</i>	Branch if Interrupt Mask Clear	Branch if (I) = 0	-	-	-	-	-	-	REL	2C	rr	3
BMI <i>rel</i>	Branch if Minus	Branch if (N) = 1	-	-	-	-	-	-	REL	2B	rr	3
BMS <i>rel</i>	Branch if Interrupt Mask Set	Branch if (I) = 1	-	-	-	-	-	-	REL	2D	rr	3
BNE <i>rel</i>	Branch if Not Equal	Branch if (Z) = 0	-	-	-	-	-	-	REL	26	rr	3
BPL <i>rel</i>	Branch if Plus	Branch if (N) = 0	-	-	-	-	-	-	REL	2A	rr	3
BRA <i>rel</i>	Branch Always	No Test	-	-	-	-	-	-	REL	20	rr	3



Table 15-2. HCS08 Instruction Set Summary (Sheet 3 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear	Branch if (Mn) = 0	-	-	-	-	-	p	DIR (b0)	01	dd rr	5
			DIR (b1)	03	dd rr	5						
			DIR (b2)	05	dd rr	5						
			DIR (b3)	07	dd rr	5						
			DIR (b4)	09	dd rr	5						
			DIR (b5)	0B	dd rr	5						
			DIR (b6)	0D	dd rr	5						
DIR (b7)	0F	dd rr	5									
BRN <i>rel</i>	Branch Never	Uses 3 Bus Cycles	-	-	-	-	-	REL	21	rr	3	
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set	Branch if (Mn) = 1	-	-	-	-	-	p	DIR (b0)	00	dd rr	5
			DIR (b1)	02	dd rr	5						
			DIR (b2)	04	dd rr	5						
			DIR (b3)	06	dd rr	5						
			DIR (b4)	08	dd rr	5						
			DIR (b5)	0A	dd rr	5						
			DIR (b6)	0C	dd rr	5						
DIR (b7)	0E	dd rr	5									
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory	Mn ← 1	-	-	-	-	-	DIR (b0)	10	dd	5	
								DIR (b1)	12	dd	5	
								DIR (b2)	14	dd	5	
								DIR (b3)	16	dd	5	
								DIR (b4)	18	dd	5	
								DIR (b5)	1A	dd	5	
								DIR (b6)	1C	dd	5	
								DIR (b7)	1E	dd	5	
BSR <i>rel</i>	Branch to Subroutine	PC ← (PC) + 0x0002 push (PCL); SP ← (SP) - 0x0001 push (PCH); SP ← (SP) - 0x0001 PC ← (PC) + <i>rel</i>	-	-	-	-	-	REL	AD	rr	5	
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 Equal	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	31	dd rr	5	
			IMM	41	ii rr	4						
			IMM	51	ii rr	4						
			IX1+	61	ff rr	5						
			IX+	71	rr	5						
			SP1	9E61	ff rr	6						
CLC	Clear Carry Bit	C ← 0	-	-	-	-	0	INH	98		1	
CLI	Clear Interrupt Mask Bit	I ← 0	-	-	0	-	-	INH	9A		1	
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 ← 0x00 A ← 0x00 X ← 0x00 H ← 0x00 M ← 0x00 M ← 0x00 M ← 0x00	0	-	-	0	1	-	DIR	3F	dd	5
			INH	4F		1						
			INH	5F		1						
			INH	8C		1						
			IX1	6F	ff	5						
			IX	7F		4						
			SP1	9E6F	ff	6						
CMP <i>#opr8i</i> CMP <i>opr8a</i> CMP <i>opr16a</i> CMP <i>opr16,X</i> CMP <i>opr8,X</i> CMP <i>,X</i> CMP <i>opr16,SP</i> CMP <i>opr8,SP</i>	Compare Accumulator with Memory	(A) - (M) (CCR Updated But Operands Not Changed)	p	-	-	p	p	p	IMM	A1	ii	2
			DIR	B1	dd	3						
			EXT	C1	hh ll	4						
			IX2	D1	ee ff	4						
			IX1	E1	ff	3						
			IX	F1		3						
			SP2	9ED1	ee ff	5						
SP1	9EE1	ff	4									
COM <i>opr8a</i> COMA COMX COM <i>opr8,X</i> COM <i>,X</i> COM <i>opr8,SP</i>	Complement (One's Complement)	M ← (M) = 0xFF - (M) A ← (A) = 0xFF - (A) X ← (X) = 0xFF - (X) M ← (M) = 0xFF - (M) M ← (M) = 0xFF - (M) M ← (M) = 0xFF - (M)	0	-	-	p	p	1	DIR	33	dd	5
			INH	43		1						
			INH	53		1						
			IX1	63	ff	5						
			IX	73		4						
			SP1	9E63	ff	6						
CPHX <i>opr16a</i> CPHX <i>#opr16i</i> CPHX <i>opr8a</i> CPHX <i>opr8,SP</i>	Compare Index Register (H:X) with Memory	(H:X) - (M:M + 0x0001) (CCR Updated But Operands Not Changed)	p	-	-	p	p	p	EXT	3E	hh ll	6
			IMM	65	jj kk	3						
			DIR	75	dd	5						
			SP1	9EF3	ff	6						

Table 15-2. HCS08 Instruction Set Summary (Sheet 4 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
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)	p	–	–	p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 B3 C3 D3 E3 F3 9ED3 9EE3	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	(A) <sub>10</sub>	U	–	–	p	p	p	INH	72		1
DBNZ opr8a,rel DBNZA rel DBNZX rel DBNZ oprx8,X,rel DBNZ ,X,rel DBNZ oprx8,SP,rel	Decrement and Branch if Not Zero	Decrement A, X, or M Branch if (result) ≠ 0 DBNZX Affects X Not H	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr rr rr rr rr rr rr ff rr	7 4 4 7 6 8
DEC opr8a DECA DECX DEC oprx8,X DEC ,X DEC oprx8,SP	Decrement	M ← (M) – 0x01 A ← (A) – 0x01 X ← (X) – 0x01 M ← (M) – 0x01 M ← (M) – 0x01 M ← (M) – 0x01	p	–	–	p	p	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff ff ff	5 1 1 5 4 6
DIV	Divide	A ← (H:A) ÷ (X) H ← Remainder	–	–	–	–	p	p	INH	52		6
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 ← (A ⊕ M)	0	–	–	p	p	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 B8 C8 D8 E8 F8 9ED8 9EE8	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
INC opr8a INCA INCX INC oprx8,X INC ,X INC oprx8,SP	Increment	M ← (M) + 0x01 A ← (A) + 0x01 X ← (X) + 0x01 M ← (M) + 0x01 M ← (M) + 0x01 M ← (M) + 0x01	p	–	–	p	p	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	5 1 1 5 4 6
JMP opr8a JMP opr16a JMP oprx16,X JMP oprx8,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	3 4 4 3 3
JSR opr8a JSR opr16a JSR oprx16,X JSR oprx8,X JSR ,X	Jump to Subroutine	PC ← (PC) + n (n = 1, 2, or 3) Push (PCL); SP ← (SP) – 0x0001 Push (PCH); SP ← (SP) – 0x0001 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	5 6 6 5 5
LDA #opr8i LDA opr8a LDA opr16a LDA oprx16,X LDA oprx8,X LDA ,X LDA oprx16,SP LDA oprx8,SP	Load Accumulator from Memory	A ← (M)	0	–	–	p	p	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 B6 C6 D6 E6 F6 9ED6 9EE6	ii dd hh ll ee ff ff ff	2 3 4 4 3 3 5 4
LDHX #opr16i LDHX opr8a LDHX opr16a LDHX ,X LDHX oprx16,X LDHX oprx8,X LDHX oprx8,SP	Load Index Register (H:X) from Memory	H:X ← (M:M + 0x0001)	0	–	–	p	p	–	IMM DIR EXT IX IX2 IX1 SP1	45 55 32 9EAE 9EBE 9ECE 9EFE	jj kk dd ll hh ll ff ff ff	3 4 5 5 6 5 5

Table 15-2. HCS08 Instruction Set Summary (Sheet 5 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
LDX #opr8i LDX opr8a LDX opr16a LDX oprx16,X LDX oprx8,X LDX ,X LDX oprx16,SP LDX oprx8,SP	Load X (Index Register Low) from Memory	$X \leftarrow (M)$	0	-	-	b	b	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEF	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
LSL opr8a LSLA LSLX LSL oprx8,X LSL ,X LSL oprx8,SP	Logical Shift Left (Same as ASL)		b	-	-	b	b	b	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff ff	5 1 1 5 4 6
LSR opr8a LSRA LSRX LSR oprx8,X LSR ,X LSR oprx8,SP	Logical Shift Right		b	-	-	0	b	b	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	5 1 1 5 4 6
MOV opr8a,opr8a MOV opr8a,X+ MOV #opr8i,opr8a MOV ,X+,opr8a	Move	$(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ $H:X \leftarrow (H:X) + 0x0001$ in IX+/DIR and DIR/IX+ Modes	0	-	-	b	b	-	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd ii ii dd dd	5 5 4 5
MUL	Unsigned multiply	$X:A \leftarrow (X) \times (A)$	-	0	-	-	-	0	INH	42		5
NEG opr8a NEGA NEGX NEG oprx8,X NEG ,X NEG oprx8,SP	Negate (Two's Complement)	$M \leftarrow -(M) = 0x00 - (M)$ $A \leftarrow -(A) = 0x00 - (A)$ $X \leftarrow -(X) = 0x00 - (X)$ $M \leftarrow -(M) = 0x00 - (M)$ $M \leftarrow -(M) = 0x00 - (M)$ $M \leftarrow -(M) = 0x00 - (M)$	b	-	-	b	b	b	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	5 1 1 5 4 6
NOP	No Operation	Uses 1 Bus Cycle	-	-	-	-	-	-	INH	9D		1
NSA	Nibble Swap Accumulator	$A \leftarrow (A[3:0]:A[7:4])$	-	-	-	-	-	-	INH	62		1
ORA #opr8i ORA opr8a ORA opr16a ORA oprx16,X ORA oprx8,X ORA ,X ORA oprx16,SP ORA oprx8,SP	Inclusive OR Accumulator and Memory	$A \leftarrow (A)   (M)$	0	-	-	b	b	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
PSHA	Push Accumulator onto Stack	Push (A); $SP \leftarrow (SP) - 0x0001$	-	-	-	-	-	-	INH	87		2
PSHH	Push H (Index Register High) onto Stack	Push (H); $SP \leftarrow (SP) - 0x0001$	-	-	-	-	-	-	INH	8B		2
PSHX	Push X (Index Register Low) onto Stack	Push (X); $SP \leftarrow (SP) - 0x0001$	-	-	-	-	-	-	INH	89		2
PULA	Pull Accumulator from Stack	$SP \leftarrow (SP + 0x0001)$ ; Pull (A)	-	-	-	-	-	-	INH	86		3
PULH	Pull H (Index Register High) from Stack	$SP \leftarrow (SP + 0x0001)$ ; Pull (H)	-	-	-	-	-	-	INH	8A		3
PULX	Pull X (Index Register Low) from Stack	$SP \leftarrow (SP + 0x0001)$ ; Pull (X)	-	-	-	-	-	-	INH	88		3
ROL opr8a ROLA ROLX ROL oprx8,X ROL ,X ROL oprx8,SP	Rotate Left through Carry		b	-	-	b	b	b	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	5 1 1 5 4 6

Table 15-2. HCS08 Instruction Set Summary (Sheet 6 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry		p	-	-	p	p	p	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	5 1 1 5 4 6
RSP	Reset Stack Pointer	SP ← 0xFF (High Byte Not Affected)	-	-	-	-	-	-	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 0x0001; Pull (CCR) SP ← (SP) + 0x0001; Pull (A) SP ← (SP) + 0x0001; Pull (X) SP ← (SP) + 0x0001; Pull (PCH) SP ← (SP) + 0x0001; Pull (PCL)	p	p	p	p	p	p	INH	80		9
RTS	Return from Subroutine	SP ← SP + 0x0001; Pull (PCH) SP ← SP + 0x0001; Pull (PCL)	-	-	-	-	-	-	INH	81		6
SBC <i>#opr8i</i> SBC <i>opr8a</i> SBC <i>opr16a</i> SBC <i>opr16,X</i> SBC <i>opr8,X</i> SBC <i>,X</i> SBC <i>opr16,SP</i> SBC <i>opr8,SP</i>	Subtract with Carry	A ← (A) - (M) - (C)	p	-	-	p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 B2 C2 D2 E2 F2 9ED2 9EE2	ii dd hh ll ee ff ff ff ff ff	2 3 4 4 3 3 5 4
SEC	Set Carry Bit	C ← 1	-	-	-	-	-	1	INH	99		1
SEI	Set Interrupt Mask Bit	I ← 1	-	-	1	-	-	-	INH	9B		1
STA <i>opr8a</i> STA <i>opr16a</i> STA <i>opr16,X</i> STA <i>opr8,X</i> STA <i>,X</i> STA <i>opr16,SP</i> STA <i>opr8,SP</i>	Store Accumulator in Memory	M ← (A)	0	-	-	p	p	-	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff ff ff ff	3 4 4 3 2 5 4
STHX <i>opr8a</i> STHX <i>opr16a</i> STHX <i>opr8,SP</i>	Store H:X (Index Reg.)	(M:M + 0x0001) ← (H:X)	0	-	-	p	p	-	DIR EXT SP1	35 96 9EFF	dd hh ll ff	4 5 5
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	I bit ← 0; Stop Processing	-	-	0	-	-	-	INH	8E		2+
STX <i>opr8a</i> STX <i>opr16a</i> STX <i>opr16,X</i> STX <i>opr8,X</i> STX <i>,X</i> STX <i>opr16,SP</i> STX <i>opr8,SP</i>	Store X (Low 8 Bits of Index Register) in Memory	M ← (X)	0	-	-	p	p	-	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ff ff ff	3 4 4 3 2 5 4
SUB <i>#opr8i</i> SUB <i>opr8a</i> SUB <i>opr16a</i> SUB <i>opr16,X</i> SUB <i>opr8,X</i> SUB <i>,X</i> SUB <i>opr16,SP</i> SUB <i>opr8,SP</i>	Subtract	A ← (A) - (M)	p	-	-	p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ff ff ff	2 3 4 4 3 3 5 4
SWI	Software Interrupt	PC ← (PC) + 0x0001 Push (PCL); SP ← (SP) - 0x0001 Push (PCH); SP ← (SP) - 0x0001 Push (X); SP ← (SP) - 0x0001 Push (A); SP ← (SP) - 0x0001 Push (CCR); SP ← (SP) - 0x0001 I ← 1; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	-	-	1	-	-	-	INH	83		11

Table 15-2. HCS08 Instruction Set Summary (Sheet 7 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
TAP	Transfer Accumulator to CCR	CCR ← (A)	p	p	p	p	p	INH	84		1	
TAX	Transfer Accumulator to X (Index Register Low)	X ← (A)	-	-	-	-	-	INH	97		1	
TPA	Transfer CCR to Accumulator	A ← (CCR)	-	-	-	-	-	INH	85		1	
TST <i>oprba</i> TSTA TSTX TST <i>opr8,X</i> TST <i>,X</i> TST <i>opr8,SP</i>	Test for Negative or Zero	(M) – 0x00 (A) – 0x00 (X) – 0x00 (M) – 0x00 (M) – 0x00 (M) – 0x00	0	-	-	p	p	-	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	H:X ← (SP) + 0x0001	-	-	-	-	-	INH	95		2	
TXA	Transfer X (Index Reg. Low) to Accumulator	A ← (X)	-	-	-	-	-	INH	9F		1	
TXS	Transfer Index Reg. to SP	SP ← (H:X) – 0x0001	-	-	-	-	-	INH	94		2	
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Halt CPU	-	-	0	-	-	INH	8F		2+	

<sup>1</sup> Bus clock frequency is one-half of the CPU clock frequency.

Table 15-3. Opcode Map (Sheet 1 of 3)

Bit-Manipulation		Branch		Read-Modify-Write				Control				Register/Memory							
00 BRSET0 3 DIR	5 BSET0 2 DIR	20 BRA 2 REL	3 3A 2 REL	30 NEG 2 DIR	5 5A 2 DIR	40 NEGA 1 INH	50 NEG1 1 INH	60 NEG 2 IX1	5 6A 2 IX1	70 NEG 1 IX	80 RTI 1 INH	90 BGE 2 REL	A0 SUB 2 IMM	3 B0 2 DIR	4 C0 3 EXT	5 D0 3 IX2	6 E0 2 IX1	7 F0 1 IX	
01 BRCLR0 3 DIR	5 BCLR0 2 DIR	21 BRN 2 REL	3 31 2 REL	31 CBEQ 3 DIR	5 51 2 IMM	41 CBEQA 4 IMM	51 CBEQX 3 IMM	61 CBEQ 3 IX1+	5 71 2 IX+	81 RTS 1 INH	91 BLT 2 REL	A1 CMP 2 IMM	3 B1 2 DIR	4 C1 3 EXT	5 D1 3 IX2	6 E1 2 IX1	7 F1 1 IX		
02 BRSET1 3 DIR	5 BSET1 2 DIR	22 BHI 2 REL	3 32 2 REL	32 LDHX 3 EXT	5 52 2 IMM	42 MUL 1 INH	52 DIV 1 INH	62 NSA 1 INH	72 DAA 1 INH	82 BGND 5+ INH	92 BGT 2 REL	A2 SBC 2 IMM	3 B2 2 DIR	4 C2 3 EXT	5 D2 3 IX2	6 E2 2 IX1	7 F2 1 IX		
03 BRCLR1 3 DIR	5 BCLR1 2 DIR	23 BLS 2 REL	3 33 2 REL	33 COM 2 DIR	5 53 2 IMM	43 COMA 1 INH	53 COMX 1 INH	63 COM 2 IX1	73 COM 1 IX	83 SWI 1 INH	93 BLE 2 REL	A3 CPX 2 IMM	3 B3 2 DIR	4 C3 3 EXT	5 D3 3 IX2	6 E3 2 IX1	7 F3 1 IX		
04 BRSET2 3 DIR	5 BSET2 2 DIR	24 BCC 2 REL	3 34 2 REL	34 LSR 2 DIR	5 54 2 IMM	44 LSRA 1 INH	54 LSRX 1 INH	64 LSR 2 IX1	74 LSR 1 IX	84 TAP 1 INH	94 TXS 2 REL	A4 AND 2 IMM	3 B4 2 DIR	4 C4 3 EXT	5 D4 3 IX2	6 E4 2 IX1	7 F4 1 IX		
05 BRCLR2 3 DIR	5 BCLR2 2 DIR	25 BCS 2 REL	3 35 2 REL	35 STHX 4 DIR	5 55 2 IMM	45 LDHX 3 IMM	55 LDHX 2 DIR	65 CPHX 3 IMM	75 CPHX 2 DIR	85 TPA 1 INH	95 TSX 1 INH	A5 BIT 2 IMM	3 B5 2 DIR	4 C5 3 EXT	5 D5 3 IX2	6 E5 2 IX1	7 F5 1 IX		
06 BRSET3 3 DIR	5 BSET3 2 DIR	26 BNE 2 REL	3 36 2 REL	36 ROR 2 DIR	5 56 2 IMM	46 RORA 1 INH	56 RORX 1 INH	66 ROR 2 IX1	76 ROR 1 IX	86 PULA 1 INH	96 STHX 3 EXT	A6 LDA 2 IMM	3 B6 2 DIR	4 C6 3 EXT	5 D6 3 IX2	6 E6 2 IX1	7 F6 1 IX		
07 BRCLR3 3 DIR	5 BCLR3 2 DIR	27 BEQ 2 REL	3 37 2 REL	37 ASR 2 DIR	5 57 2 IMM	47 ASRA 1 INH	57 ASRX 1 INH	67 ASR 2 IX1	77 ASR 1 IX	87 PSHA 1 INH	97 TAX 1 INH	A7 AIS 2 IMM	3 B7 2 DIR	4 C7 3 EXT	5 D7 3 IX2	6 E7 2 IX1	7 F7 1 IX		
08 BRSET4 3 DIR	5 BSET4 2 DIR	28 BHCC 2 REL	3 38 2 REL	38 LSL 2 DIR	5 58 2 IMM	48 LSLA 1 INH	58 LSLX 1 INH	68 LSL 2 IX1	78 LSL 1 IX	88 PULX 3 INH	98 CLC 1 INH	A8 EOR 2 IMM	3 B8 2 DIR	4 C8 3 EXT	5 D8 3 IX2	6 E8 2 IX1	7 F8 1 IX		
09 BRCLR4 3 DIR	5 BCLR4 2 DIR	29 BHCS 2 REL	3 39 2 REL	39 ROL 2 DIR	5 59 2 IMM	49 ROLA 1 INH	59 ROLX 1 INH	69 ROL 2 IX1	79 ROL 1 IX	89 PSHX 2 INH	99 SEC 1 INH	A9 ADC 2 IMM	3 B9 2 DIR	4 C9 3 EXT	5 D9 3 IX2	6 E9 2 IX1	7 F9 1 IX		
0A BRSET5 3 DIR	5 BSET5 2 DIR	2A BPL 2 REL	3 3A 2 REL	3A DEC 2 DIR	5 5A 2 IMM	4A DECA 1 INH	5A DECX 1 INH	6A DEC 2 IX1	7A DEC 1 IX	8A PULH 3 INH	9A CLI 1 INH	AA ORA 2 IMM	3 BA 2 DIR	4 CA 3 EXT	5 DA 3 IX2	6 EA 2 IX1	7 FA 1 IX		
0B BRCLR5 3 DIR	5 BCLR5 2 DIR	2B BMI 2 REL	3 3B 2 REL	3B DBNZ 3 DIR	5 5B 2 IMM	4B DBNZA 2 INH	5B DBNZX 4 INH	6B DBNZ 3 IX1	7B DBNZ 2 IX	8B PSHH 2 INH	9B SEI 1 INH	AB ADD 2 IMM	3 BB 2 DIR	4 CB 3 EXT	5 DB 3 IX2	6 EB 2 IX1	7 FB 1 IX		
0C BRSET6 3 DIR	5 BSET6 2 DIR	2C BMC 2 REL	3 3C 2 REL	3C INC 2 DIR	5 5C 2 IMM	4C INCA 1 INH	5C INCX 1 INH	6C INC 2 IX1	7C INC 1 IX	8C CLR1 1 INH	9C RSP 1 INH		3 BC 2 DIR	4 CC 3 EXT	5 DC 3 IX2	6 EC 2 IX1	7 FC 1 IX		
0D BRCLR6 3 DIR	5 BCLR6 2 DIR	2D BMS 2 REL	3 3D 2 REL	3D TST 2 DIR	5 5D 2 IMM	4D TSTA 1 INH	5D TSTX 1 INH	6D TST 2 IX1	7D TST 1 IX		9D NOP 1 INH	AD BSR 2 REL	3 BD 2 DIR	4 CD 3 EXT	5 DD 3 IX2	6 ED 2 IX1	7 FD 1 IX		
0E BRSET7 3 DIR	5 BSET7 2 DIR	2E BIL 2 REL	3 3E 2 REL	3E CPHX 3 EXT	5 5E 2 IMM	4E MOV 3 DD	5E MOV 2 DIX+	6E MOV 3 IMD	7E MOV 2 IX+D	8E STOP 2+ INH	9E Page 2	AE LDX 2 IMM	3 BE 2 DIR	4 CE 3 EXT	5 DE 3 IX2	6 EE 2 IX1	7 FE 1 IX		

**Table 15-3. Opcode Map (Sheet 2 of 3)**

0F BRCLR7 3 DIR	5 2	1F BCLR7 2 DIR	5 2	2F BIH 2 REL	3 2	3F CLR 2 DIR	5 1	4F CLRA 1 INH	1 1	5F CLR 1 INH	1 2	6F CLR 2 IX1	5 1	7F CLR 1 IX	4 1	8F WAIT 2+ INH	1 1	9F TXA 1 INH	1 2	AF AIX 2 IMM	2 2	BF STX 3 DIR	3 3	CF STX 3 EXT	4 3	DF STX 3 IX2	4 2	EF STX 3 IX1	3 1	FF STX 2 IX	2 2
-----------------------	--------	----------------------	--------	--------------------	--------	--------------------	--------	---------------------	--------	--------------------	--------	--------------------	--------	-------------------	--------	----------------------	--------	--------------------	--------	--------------------	--------	--------------------	--------	--------------------	--------	--------------------	--------	--------------------	--------	-------------------	--------

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 Post Increment
EXT	Extended	IX2	Indexed, 16-Bit Offset		Post Increment
DD	DIR to DIR	IMD	IMM to DIR	IX1+	Indexed, 1-Byte Offset with Post Increment
IX+D	IX+ to DIR	DIX+	DIR to IX+		

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







# Chapter 16

## MCU Keyboard Interrupt (KBI)

### 16.1 Introduction

The KBI module allows up to eight pins to act as additional interrupt sources. Four of these pins allow falling-edge sensing while the other four can be configured for either rising-edge sensing or falling-edge sensing. The sensing mode for all eight pins can also be modified to detect edges and levels instead of only edges.

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 up the MCU from stop or wait low-power modes.

#### 16.1.1 KBI Block Diagram

Figure 16-1 shows the block diagram for a KBI module.

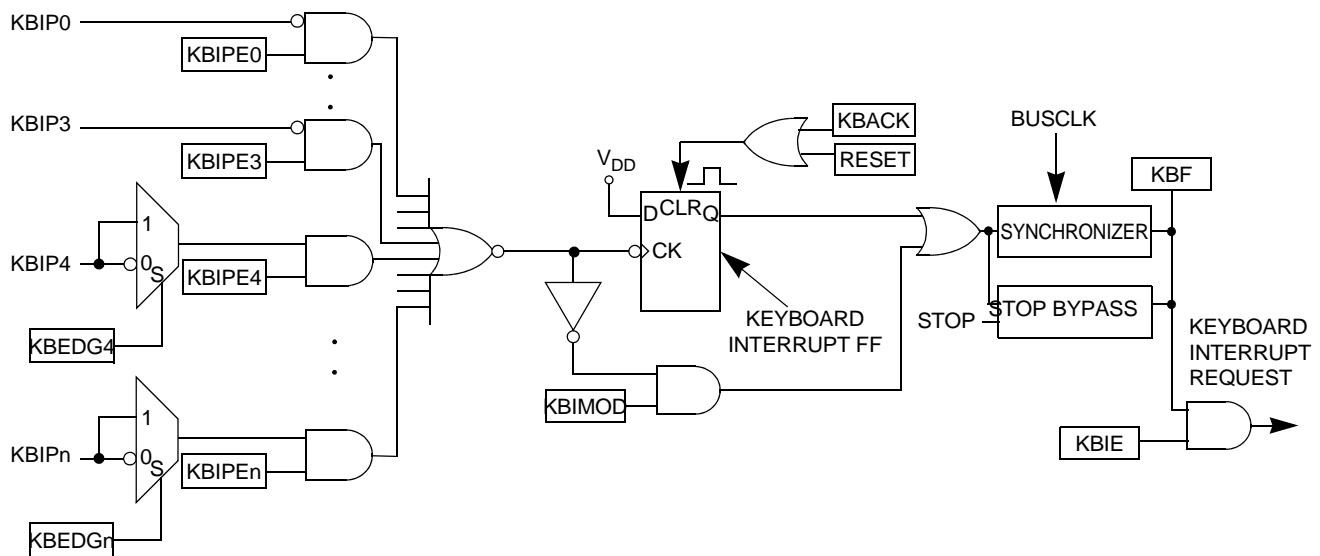


Figure 16-1. KBI Block Diagram

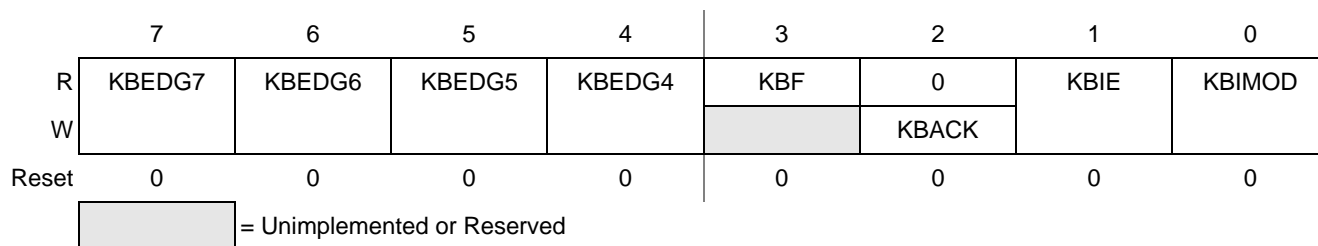
### 16.2 Register Definitions

This section provides information about all registers and control bits associated with the KBI module.

Refer to the direct-page register summary in [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all KBI 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.

## 16.2.1 KBI Status and Control Register (KBISC)

Offset



**Figure 16-2. KBI Status and Control Register (KBISC)**

**Table 16-1. KBISC Register Field Descriptions**

Field	Description
7:4 KBEDG[7:4]	<p><b>Keyboard Edge Select for KBI Port Bits</b> — Each of these read/write bits selects the polarity of the edges and/or levels that are recognized as trigger events on the corresponding KBI port pin when it is configured as a keyboard interrupt input (KBIPEn = 1). Also see the KBIMOD control bit, which determines whether the pin is sensitive to edges-only or edges and levels.</p> <p>0 Falling edges/low levels 1 Rising edges/high levels</p>
3 KBF	<p><b>Keyboard Interrupt Flag</b> — This read-only status flag is set whenever the selected edge event has been detected on any of the enabled KBI port pins. This flag is cleared by writing a 1 to the KBACK control bit. The flag will remain set if KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level.</p> <p>KBF can be used as a software pollable flag (KBIE = 0) or it can generate a hardware interrupt request to the CPU (KBIE = 1).</p> <p>0 No KBI interrupt pending 1 KBI interrupt pending</p>
2 KBACK	<p><b>Keyboard Interrupt Acknowledge</b> — This write-only bit (reads always return 0) is used to clear the KBF status flag by writing a 1 to KBACK. When KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level, KBF is being continuously set so writing 1 to KBACK does not clear the KBF flag.</p>
1 KBIE	<p><b>Keyboard Interrupt Enable</b> — This read/write control bit determines whether hardware interrupts are generated when the KBF status flag equals 1. When KBIE = 0, no hardware interrupts are generated, but KBF can still be used for software polling.</p> <p>0 KBF does not generate hardware interrupts (use polling) 1 KBI hardware interrupt requested when KBF = 1</p>
KBIMOD	<p><b>Keyboard Detection Mode</b> — This read/write control bit selects either edge-only detection or edge-and-level detection. KBI port bits 3 through 0 can detect falling edges-only or falling edges and low levels. KBI port bits 7 through 4 can be configured to detect either:</p> <ul style="list-style-type: none"> <li>• Rising edges-only or rising edges and high levels (KBEDGn = 1)</li> <li>• Falling edges-only or falling edges and low levels (KBEDGn = 0)</li> </ul> <p>0 Edge-only detection 1 Edge-and-level detection</p>

## 16.2.2 KBI Pin Enable Register (KBIPE)

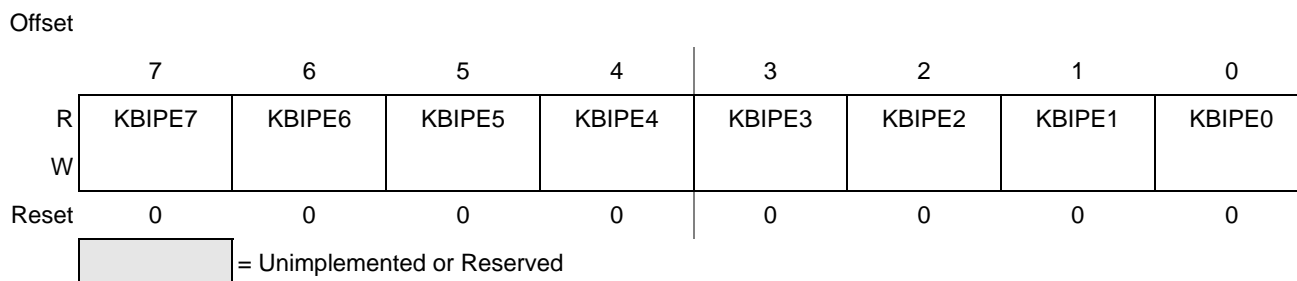


Figure 16-3. KBI Pin Enable Register (KBIPE)

Table 16-2. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPE[7:0]	<p><b>Keyboard Pin Enable for KBI Port Bits</b> — Each of these read/write bits selects whether the associated KBI port pin is enabled as a keyboard interrupt input or functions as a general-purpose I/O pin.</p> <p>0 Bit n of KBI port is a general-purpose I/O pin not associated with the KBI</p> <p>1 Bit n of KBI port enabled as a keyboard interrupt input</p>

## 16.3 Functional Description

### 16.3.1 Pin Enables

The KBIPE<sub>n</sub> control bits in the KBIPE register allow a user to enable (KBIPE<sub>n</sub> = 1) any combination of KBI-related port pins to be connected to the KBI module. Pins corresponding to 0s in KBIPE are general-purpose I/O pins that are not associated with the KBI module.

### 16.3.2 Edge and Level Sensitivity

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled keyboard inputs in a KBI module must be at the de asserted logic level.

A falling edge is detected when an enabled keyboard input signal is seen as a logic 1 (the de asserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle.

A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

The KBIMOD control bit can be set to reconfigure the detection logic so that it detects edges and levels. In KBIMOD = 1 mode, the KBF status flag becomes set when an edge is detected (when one or more enabled pins change from the de asserted to the asserted level while all other enabled pins remain at their de asserted levels), but the flag is continuously set (and cannot be cleared) as long as any enabled keyboard input pin remains at the asserted level. When the MCU enters Stop Mode, the synchronous edge-detection logic is bypassed (because clocks are stopped). In Stop Mode, KBI inputs act as asynchronous level-sensitive inputs so they can wake the MCU from Stop Mode.

### 16.3.3 KBI Interrupt Controls

The KBF status flag becomes set (1) when an edge event has been detected on any KBI input pin. If  $KBIE = 1$  in the KBISC register, a hardware interrupt will be requested whenever  $KBF = 1$ . The KBF flag is cleared by writing a 1 to the keyboard acknowledge (KBACK) bit.

When  $KBIMOD = 0$  (selecting edge-only operation), KBF is always cleared by writing 1 to KBACK. When  $KBIMOD = 1$  (selecting edge-and-level operation), KBF cannot be cleared as long as any keyboard input is at its asserted level.

# Chapter 17

## MCU Timer/PWM (TPM Module)

### 17.1 Introduction

The MC1321x includes two independent timer/PWM (TPM) modules which support traditional input capture, output compare, or buffered edge-aligned pulse-width modulation (PWM) on each channel. A control bit in each TPM configures all channels in that timer to operate as center-aligned PWM functions. In each of these two TPMs, timing functions are based on a separate 16-bit counter with prescaler and modulo features to control frequency and range (period between overflows) of the time reference. This timing system is ideally suited for a wide range of control applications, and the center-aligned PWM capability on the 3-channel TPM extends the field of applications to motor control in small appliances.

The timer system in the MC1321x includes internal logic for a 3-channel TPM1 and a separate 5-channel TPM2. However, in the package, TPM1 only has Channel 2 pinned-out and TPM2 only has Channels 1, 2, 3, and 4 pinned-out.

The use of the fixed system clock, XCLK, as the clock source for either of the TPM modules allows the TPM prescaler to run using the oscillator rate divided by two (ICGERCLK/2). This clock source must be selected only if the ICG is configured in either FBE or FEE mode. In FBE mode, this selection is redundant because the BUSCLK frequency is the same as XCLK. In FEE mode, the proper conditions must be met for XCLK to equal ICGERCLK/2 (see [Section 14.3.9, “Fixed Frequency Clock”](#)). Selecting XCLK as the clock source with the ICG in either FEI or SCM mode will result in the TPM being non-functional.

### 17.2 TPM Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPM1CH<sub>n</sub> where x is the TPM number (for example, 1 or 2) and n is the channel number (for example, 0–4). The TPM shares its I/O pins with general-purpose I/O port pins (see [Chapter 2, “MC1321x Pins and Connections”](#) for more information). [Figure 17-1](#) shows the structure of a TPM.

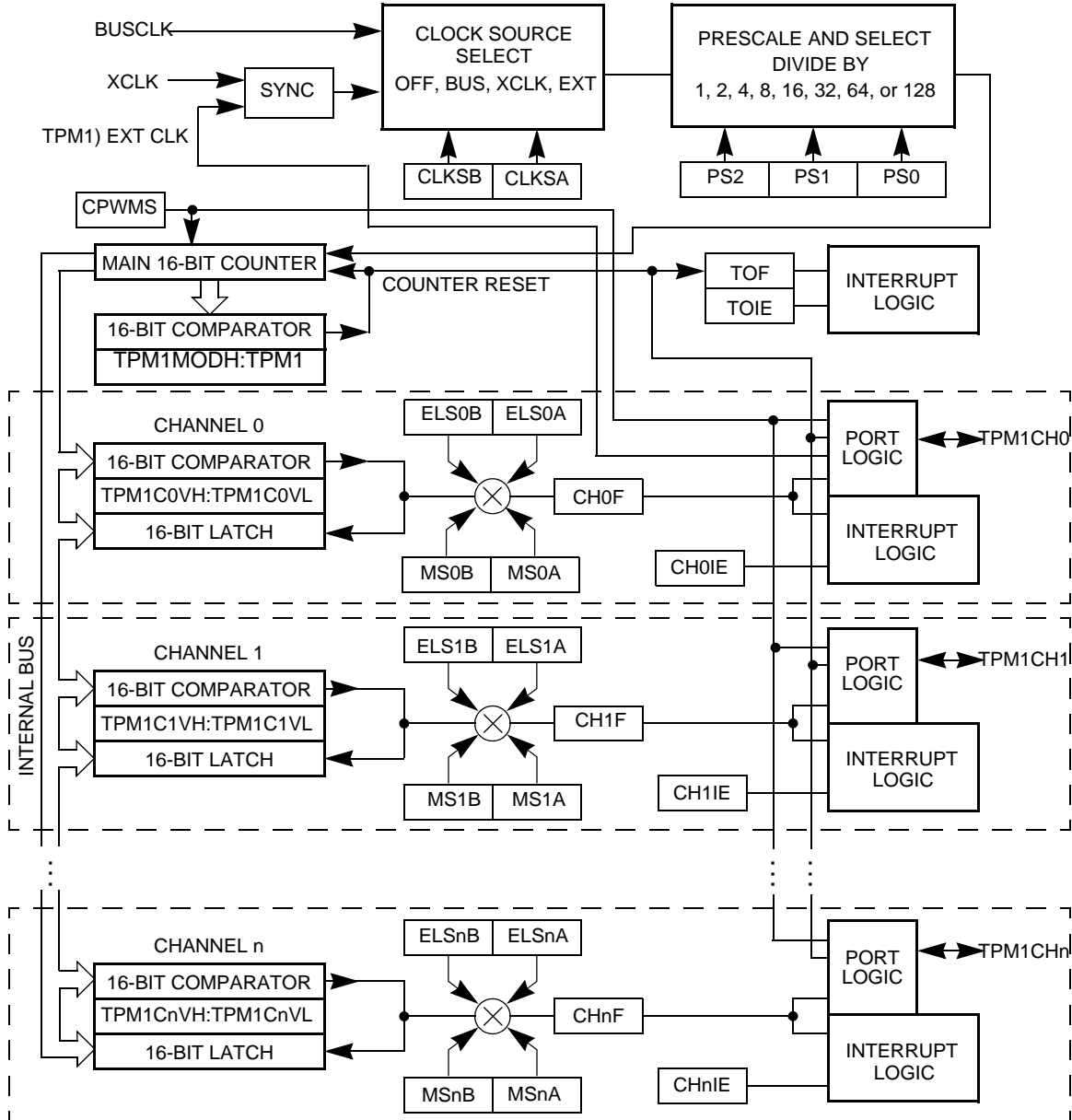


Figure 17-1. TPM Block Diagram

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM. The TPM counter (when operating in normal Up-counting Mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, **TPM1MODH:TPM1MODL**, control the modulo value of the counter. (The values \$0000 or \$FFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the **TPM1CNT** counter resets the counter regardless of the data value written.

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

## 17.3 Pin Descriptions

Table 17-2 shows the MCU pins related to the TPM modules. When TPM1CH0 is used as an external clock input, the associated TPM channel 0 can not use the pin. (Channel 0 can still be used in Output Compare Mode as a software timer.) When any of the pins associated with the timer is configured as a timer input, a passive pullup can be enabled. After reset, the TPM modules are disabled and all pins default to general-purpose inputs with the passive pullups disabled.

### 17.3.1 External TPM Clock Sources

When control bits CLKSB:CLKSA in the timer status and control register are set to 1:1, the prescaler and consequently the 16-bit counter for TPM1 are driven by an external clock source connected to the TPM1CH0 pin. A synchronizer is needed between the external clock and the rest of the TPM. This synchronizer is clocked by the bus clock so the frequency of the external source must be less than one-half the frequency of the bus rate clock. The upper frequency limit for this external clock source is specified to be one-fourth the bus frequency to conservatively accommodate duty cycle and phase-locked loop (PLL) or frequency-locked loop (FLL) frequency jitter effects.

When the TPM is using the channel 0 pin for an external clock, the corresponding ELS0B:ELS0A control bits should be set to 0:0 so channel 0 is not trying to use the same pin.

### 17.3.2 TPM1CHn — TPM1 Channel n I/O Pins

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the configuration of the channel. In some cases, no pin function is needed so the pin reverts to being controlled by general-purpose I/O controls. When a timer has control of a port pin, the port data and data direction registers do not affect the related pin(s). See the [Chapter 2, “MC1321x Pins and Connections”](#) for additional information about shared pin functions.

## 17.4 Functional Description

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

The TPM has center-aligned PWM capabilities controlled by the CPWMS control bit in TPM1SC. When CPWMS is set to 1, timer counter TPM1CNT changes to an up-/down-counter and all channels in the associated TPM act as center-aligned PWM channels. When CPWMS = 0, each channel can independently be configured to operate in input capture, output compare, or Buffered Edge-aligned PWM Mode.

The following sections describe the main 16-bit counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation

and interrupt activity depend on the operating mode, these topics are covered in the associated mode sections.

## 17.4.1 Counter

All timer functions are based on the main 16-bit counter (TPM1CNTH:TPM1CNTL). This section discusses selection of the clock source, up-counting vs. up-/down-counting, end-of-count overflow, and manual counter reset.

After any MCU reset, CLKS<sub>B</sub>:CLKS<sub>A</sub> = 0:0 so no clock source is selected and the TPM is inactive. Normally, CLKS<sub>B</sub>:CLKS<sub>A</sub> would be set to 0:1 so the bus clock drives the timer counter. The clock source for each of the TPM can be independently selected to be off, the bus clock (BUSCLK), the fixed system clock (XCLK), or an external input through the TPM1CH0 pin. The maximum frequency allowed for the external clock option is one-fourth the bus rate. Refer to [Section 17.6.1, “Timer x Status and Control Register \(TPM1SC\)”](#), and [Table 17-2](#) for more information about clock source selection.

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

The main 16-bit counter has two counting modes. When center-aligned PWM is selected (CPWMS = 1), the counter operates in Up-/down-counting Mode. Otherwise, the counter operates as a simple up-counter. As an up-counter, the main 16-bit counter counts from \$0000 through its terminal count and then continues with \$0000. The terminal count is \$FFFF or a modulus value in TPM1MODH:TPM1MODL.

When center-aligned PWM operation is specified, the counter counts upward from \$0000 through its terminal count and then counts downward to \$0000 where it returns to up-counting. Both \$0000 and the terminal count value (value in TPM1MODH:TPM1MODL) are normal length counts (one timer clock period long).

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

The conditions that cause TOF to become set depend on the Counting Mode (up or up/down). In Up-counting Mode, the main 16-bit counter counts from \$0000 through \$FFFF and overflows to \$0000 on the next counting clock. TOF becomes set at the transition from \$FFFF to \$0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to \$0000. When the main 16-bit counter is operating in Up-/down-counting Mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The \$0000 count value corresponds to the center of a period.)

Because the HCS08 MCU is an 8-bit architecture, a coherency mechanism is built into the timer counter for read operations. Whenever either byte of the counter is read (TPM1CNTH or TPM1CNTL), both bytes are captured into a buffer so when the other byte is read, the value will represent the other byte of the count at the time the first byte was read. The counter continues to count normally, but no new value can be read from either byte until both bytes of the old count have been read.



The main timer counter can be reset manually at any time by writing any value to either byte of the timer count TPM1CNTH or TPM1CNTL. Resetting the counter in this manner also resets the coherency mechanism in case only one byte of the counter was read before resetting the count.

## 17.4.2 Channel Mode Selection

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

### 17.4.2.1 Input Capture Mode

With the input capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TPM latches the contents of the TPM counter into the channel value registers (TPM1CnVH:TPM1CnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either byte of the 16-bit capture register is read, both bytes are latched into a buffer to support coherent 16-bit accesses regardless of order. The coherency sequence can be manually reset by writing to the channel status/control register (TPM1CnSC).

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

### 17.4.2.2 Output Compare Mode

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

In Output Compare Mode, values are transferred to the corresponding timer channel value registers only after both 8-bit bytes of a 16-bit register have been written. This coherency sequence can be manually reset by writing to the channel status/control register (TPM1CnSC).

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

### 17.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal Up-counting Mode of the timer counter (CPWMS = 0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the setting in the modulus register (TPM1MODH:TPM1MODL). The duty cycle is determined by the setting in the timer channel value register (TPM1CnVH:TPM1CnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As [Figure 17-2](#) shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. The time between the modulus overflow and the output compare is the pulse width. If ELSnA = 0, the counter overflow forces the PWM signal high and the output compare

forces the PWM signal low. If ELSnA = 1, the counter overflow forces the PWM signal low and the output compare forces the PWM signal high.

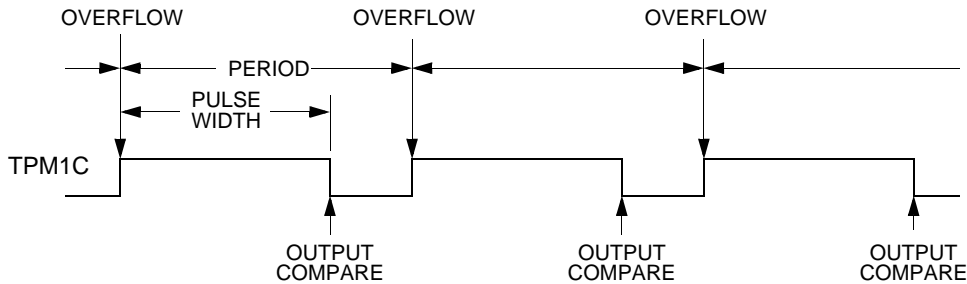


Figure 17-2. PWM Period and Pulse Width (ELSnA = 0)

When the channel value register is set to \$0000, the duty cycle is 0 percent. By setting the timer channel value register (TPM1CnVH:TPM1CnVL) to a value greater than the modulus setting, 100 percent duty cycle can be achieved. This implies that the modulus setting must be less than \$FFFF to get 100 percent duty cycle.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to either register, TPM1CnVH or TPM1CnVL, write to buffer registers. In Edge-PWM Mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the value in the TPM1CNTH:TPM1CNTL counter is \$0000. (The new duty cycle does not take effect until the next full period.)

### 17.4.3 Center-Aligned PWM Mode

This type of PWM output uses the Up-/down-counting Mode of the timer counter (CPWMS = 1). The output compare value in TPM1CnVH:TPM1CnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPM1MODH:TPM1MODL. TPM1MODH:TPM1MODL should be kept in the range of \$0001 to \$7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPM1CnVH:TPM1CnVL}) \quad \text{Eqn. 17-1}$$

$$\begin{aligned} \text{period} &= 2 \times (\text{TPM1MODH:TPM1MODL}); \\ &\text{for TPM1MODH:TPM1MODL} = \$0001\text{--}\$7FFF \end{aligned} \quad \text{Eqn. 17-2}$$

If the channel value register TPM1CnVH:TPM1CnVL is zero or negative (bit 15 set), the duty cycle will be 0 percent. If TPM1CnVH:TPM1CnVL is a positive value (bit 15 clear) and is greater than the (non zero) modulus setting, the duty cycle will be 100 percent because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is \$0001 through \$7FFE (\$7FFF if generation of 100 percent duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

TPM1MODH:TPM1MODL = \$0000 is a special case that should not be used with Center-aligned PWM Mode. When CPWMS = 0, this case corresponds to the counter running free from \$0000 through \$FFFF,

but when  $CPWMS = 1$  the counter needs a valid match to the modulus register somewhere other than at  $0000$  in order to change directions from up-counting to down-counting.

Figure 17-3 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If  $ELSnA = 0$ , the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in  $TPM1MODH:TPM1MODL$ , then counts down until it reaches zero. This sets the period equal to two times  $TPM1MODH:TPM1MODL$ .

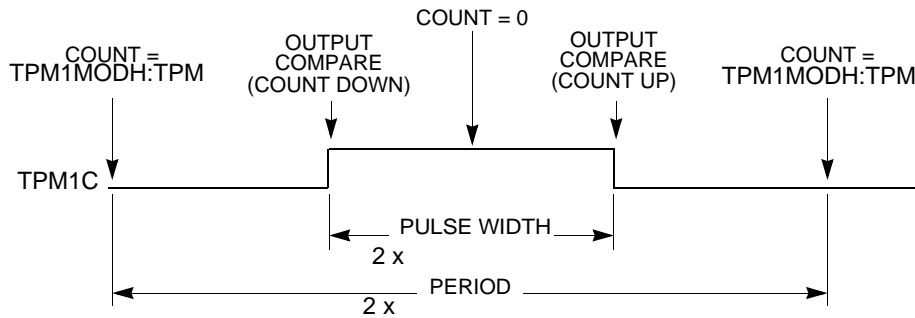


Figure 17-3. CPWM Period and Pulse Width ( $ELSnA = 0$ )

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

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers,  $TPM1MODH$ ,  $TPM1MODL$ ,  $TPM1CnVH$ , and  $TPM1CnVL$ , actually write to buffer registers. Values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the timer counter overflows (reverses direction from up-counting to down-counting at the end of the terminal count in the modulus register). This  $TPM1CNT$  overflow requirement only applies to PWM channels, not output compares.

Optionally, when  $TPM1CNTH:TPM1CNTL = TPM1MODH:TPM1MODL$ , the TPM can generate a TOF interrupt at the end of this count. The user can choose to reload any number of the PWM buffers, and they will all update simultaneously at the start of a new period.

Writing to  $TPM1SC$  cancels any values written to  $TPM1MODH$  and/or  $TPM1MODL$  and resets the coherency mechanism for the modulo registers. Writing to  $TPM1CnSC$  cancels any values written to the channel value registers and resets the coherency mechanism for  $TPM1CnVH:TPM1CnVL$ .

## 17.5 TPM Interrupts

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on the mode of operation for each channel. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for Output Compare or PWM Modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register. See the [Chapter 12](#),

“[MCU Resets, Interrupts, and System Configuration](#)” for absolute interrupt vector addresses, priority, and local interrupt mask control bits.

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

### 17.5.1 Clearing Timer Interrupt Flags

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

### 17.5.2 Timer Overflow Interrupt Description

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In Up-counting Mode, the 16-bit timer counter counts from \$0000 through \$FFFF and overflows to \$0000 on the next counting clock. TOF becomes set at the transition from \$FFFF to \$0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to \$0000. When the counter is operating in Up-/down-counting Mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The \$0000 count value corresponds to the center of a period.)

### 17.5.3 Channel Event Interrupt Description

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

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select rising edges, falling edges, any edge, or no edge (off) as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the 2-step sequence described in [Section 17.5.1, “Clearing Timer Interrupt Flags”](#).

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the 2-step sequence described in [Section 17.5.1, “Clearing Timer Interrupt Flags”](#).

### 17.5.4 PWM End-of-Duty-Cycle Events

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

- When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period.

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

The flag is cleared by the 2-step sequence described in [Section 17.5.1, “Clearing Timer Interrupt Flags”](#).

## 17.6 TPM Registers and Control Bits

The TPM includes:

- An 8-bit status and control register (TPM1SC)
- A 16-bit counter (TPM1CNTH:TPM1CNTL)
- A 16-bit modulo register (TPM1MODH:TPM1MODL)

Each timer channel has:

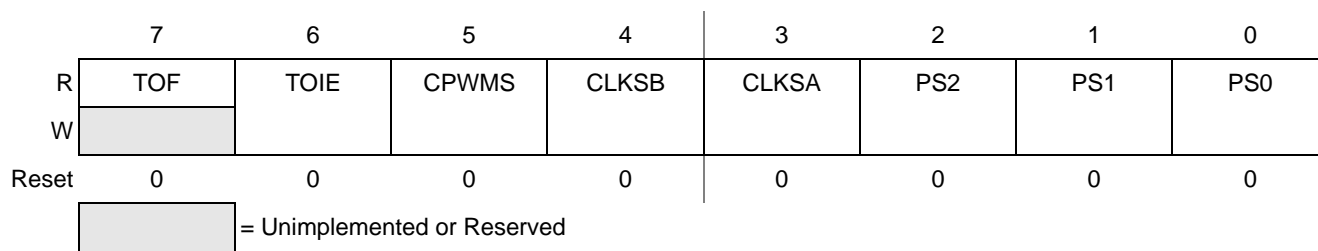
- An 8-bit status and control register (TPM1CnSC)
- A 16-bit channel value register (TPM1CnVH:TPM1CnVL)

Refer to the direct-page register summary in [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all TPM registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

The MC1321x has two TPMs, so register names include placeholder characters to identify which TPM and which channel is being referenced. For example, TPMxCnSC refers to timer (TPM) x, channel n and TPM1C2SC is the status and control register for timer 1, channel 2.

### 17.6.1 Timer x Status and Control Register (TPM1SC)

TPM1SC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.



**Figure 17-4. Timer x Status and Control Register (TPM1SC)**

**Table 17-1. TPM1SC Register Field Descriptions**

Field	Description
7 TOF	<b>Timer Overflow Flag</b> — This flag is set when the TPM counter changes to \$0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value. Clear TOF by reading the TPM status and control register when TOF is set and then writing a 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. Reset clears TOF. Writing a 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	<b>Timer Overflow Interrupt Enable</b> — This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals 1. Reset clears TOIE. 0 TOF interrupts inhibited (use software polling) 1 TOF interrupts enabled
5 CPWMS	<b>Center-Aligned PWM Select</b> — This read/write bit selects CPWM operating mode. Reset clears this bit so the TPM operates in Up-counting Mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in Up-/down-counting Mode for CPWM functions. Reset clears CPWMS. 0 All TPM1 channels operate as input capture, output compare, or Edge-aligned PWM Mode as selected by the MSnB:MSnA control bits in each channel's status and control register 1 All TPM1 channels operate in Center-aligned PWM Mode
4:3 CLKS[B:A]	<b>Clock Source Select</b> — As shown in <a href="#">Table 17-2</a> , this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The external source and the XCLK are synchronized to the bus clock by an on-chip synchronization circuit.
2:0 PS[2:0]	<b>Prescale Divisor Select</b> — This 3-bit field selects one of eight divisors for the TPM clock input as shown in <a href="#">Table 17-3</a> . This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system.

**Table 17-2. TPM Clock Source Selection**

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
0:0	No clock selected (TPM disabled)
0:1	Bus rate clock (BUSCLK)
1:0	Fixed system clock (XCLK)
1:1	External source (TPM1 Ext Clk) <sup>1,2</sup>

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

<sup>2</sup> When the TPM1CH0 pin is selected as the TPM clock source, the corresponding ELS0B:ELS0A control bits should be set to 0:0 so channel 0 does not try to use the same pin for a conflicting function.

**Table 17-3. Prescale Divisor Selection**

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

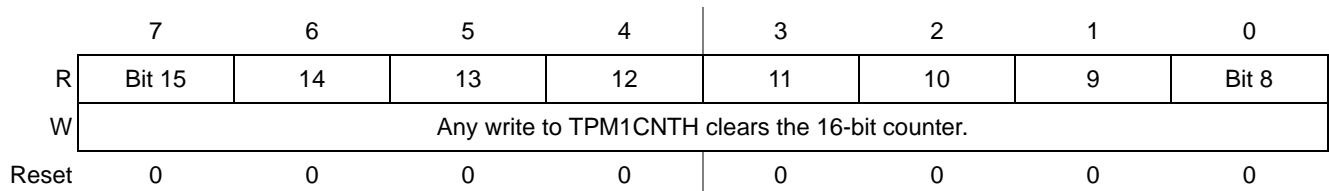
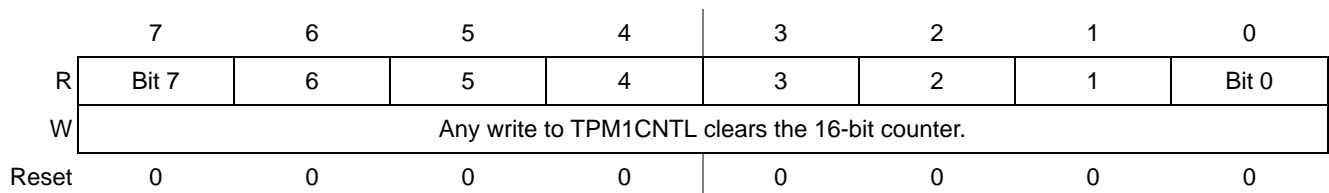
**Table 17-3. Prescale Divisor Selection**

PS2:PS1:PS0	TPM Clock Source Divided-By
1:0:1	32
1:1:0	64
1:1:1	128

### 17.6.2 Timer x Counter Registers (TPM1CNTH:TPM1CNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPM1CNTH or TPM1CNTL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This allows coherent 16-bit reads in either order. The coherency mechanism is automatically restarted by an MCU reset, a write of any value to TPM1CNTH or TPM1CNTL, or any write to the timer status/control register (TPM1SC).

Reset clears the TPM counter registers.


**Figure 17-5. Timer x Counter Register High (TPM1CNTH)**

**Figure 17-6. Timer x Counter Register Low (TPM1CNTL)**

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

### 17.6.3 Timer x Counter Modulo Registers (TPM1MODH:TPM1MODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from \$0000 at the next clock (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPM1MODH or TPM1MODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to \$0000, which results in a free-running timer counter (modulo disabled).

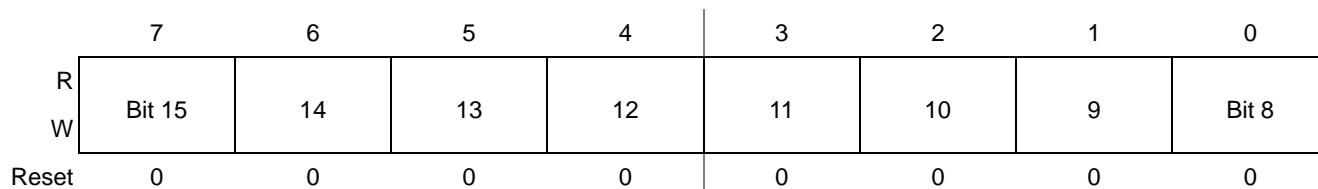


Figure 17-7. Timer x Counter Modulo Register High (TPM1MODH)

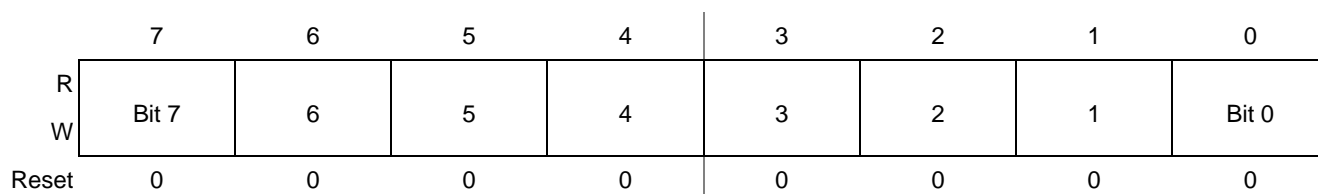


Figure 17-8. Timer x Counter Modulo Register Low (TPM1MODL)

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

### 17.6.4 Timer x Channel n Status and Control Register (TPM1CnSC)

TPM1CnSC contains the channel interrupt status flag and control bits that are used to configure the interrupt enable, channel configuration, and pin function.

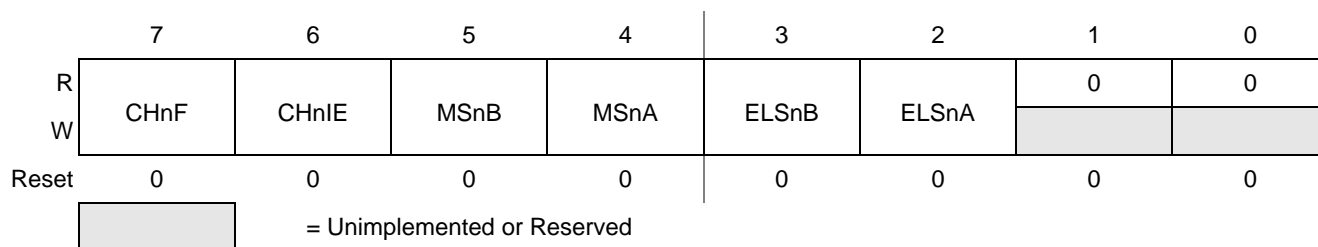


Figure 17-9. Timer x Channel n Status and Control Register (TPM1CnSC)



**Table 17-4. TPM1CnSC Register Field Descriptions**

Field	Description
7 CHnF	<p><b>Channel n Flag</b> — When channel n is configured for input capture, this flag bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. This flag is seldom used with center-aligned PWMs because it is set every time the counter matches the channel value register, which correspond to both edges of the active duty cycle period.</p> <p>A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPM1CnSC while CHnF is set and then writing a 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF would remain set after the clear sequence was completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost by clearing a previous CHnF. Reset clears CHnF. Writing a 1 to CHnF has no effect.</p> <p>0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event occurred on channel n</p>
6 CHnIE	<p><b>Channel n Interrupt Enable</b> — This read/write bit enables interrupts from channel n. Reset clears CHnIE.</p> <p>0 Channel n interrupt requests disabled (use software polling) 1 Channel n interrupt requests enabled</p>
5 MSnB	<p><b>Mode Select B for TPM Channel n</b> — When CPWMS = 0, MSnB = 1 configures TPM channel n for edge-aligned PWM Mode. For a summary of Channel Mode and setup controls, refer to <a href="#">Table 17-5</a>.</p>
4 MSnA	<p><b>Mode Select A for TPM Channel n</b> — When CPWMS = 0 and MSnB = 0, MSnA configures TPM channel n for Input Capture Mode or Output Compare Mode. Refer to <a href="#">Table 17-5</a> for a summary of Channel Mode and setup controls.</p>
3:2 ELSn[B:A]	<p><b>Edge/Level Select Bits</b> — Depending on the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in <a href="#">Table 17-5</a>, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output.</p> <p>Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general-purpose I/O pin unrelated to any timer channel functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general-purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin. This is also the setting required for channel 0 when the TPM1CH0 pin is used as an external clock input.</p>

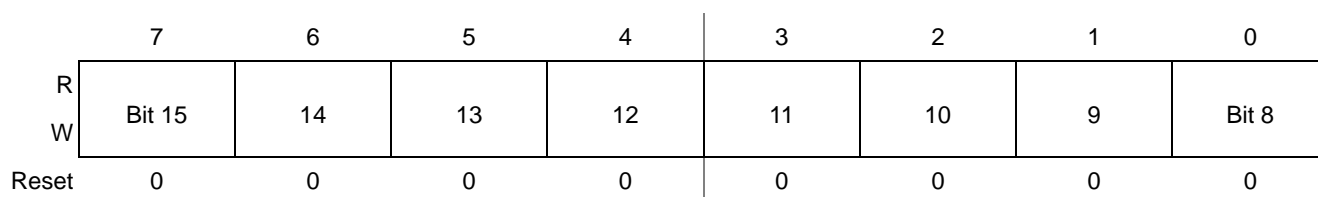
**Table 17-5. Mode, Edge, and Level Selection**

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

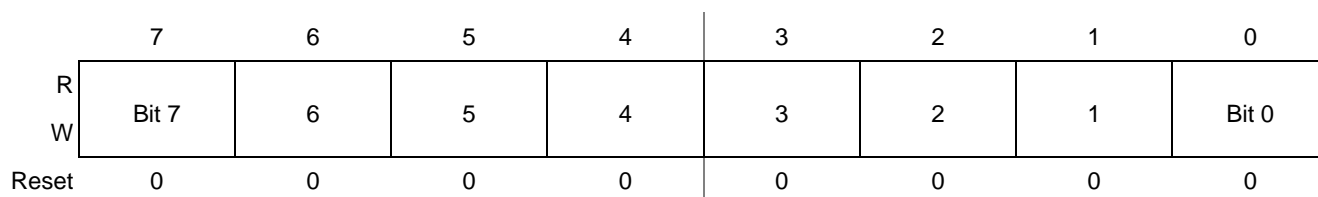
If the associated port pin is not stable for at least two bus clock cycles before changing to Input Capture Mode, it is possible to get an unexpected indication of an edge trigger. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts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

### 17.6.5 Timer x Channel Value Registers (TPM1CnVH:TPM1CnVL)

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



**Figure 17-10. Timer x Channel Value Register High (TPM1CnVH)**



**Figure 17-11. Timer Channel Value Register Low (TPM1CnVL)**

In Input Capture Mode, reading either byte (TPM1CnVH or TPM1CnVL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPM1CnSC register is written.

In output compare or PWM Modes, writing to either byte (TPM1CnVH or TPM1CnVL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the timer channel value registers. This latching mechanism may be manually reset by writing to the TPM1CnSC register.

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



# Chapter 18

## MCU Serial Communications Interface (SCI)

### 18.1 Features

Features of SCI module include:

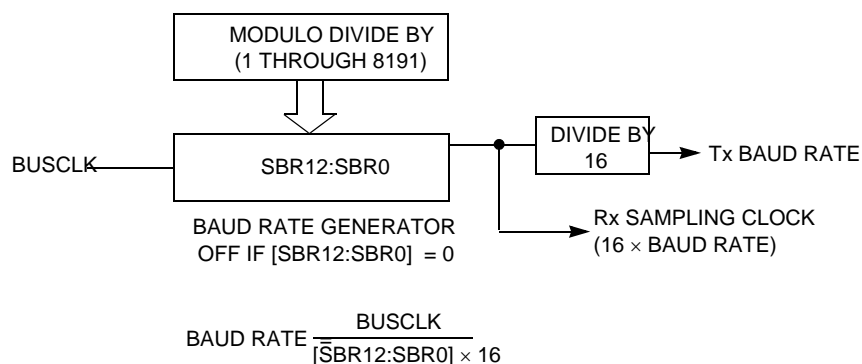
- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark

### 18.2 SCI System 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.

### 18.3 Baud Rate Generation

As shown in [Figure 18-1](#), the clock source for the SCI baud rate generator is the bus-rate clock.



**Figure 18-1. 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 HCS08 re-synchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about  $\pm 4.5$  percent for 8-bit data format and about  $\pm 4$  percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

## 18.4 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.

### 18.4.1 Transmitter Block Diagram

Figure 18-2 shows the transmitter portion of the SCI.

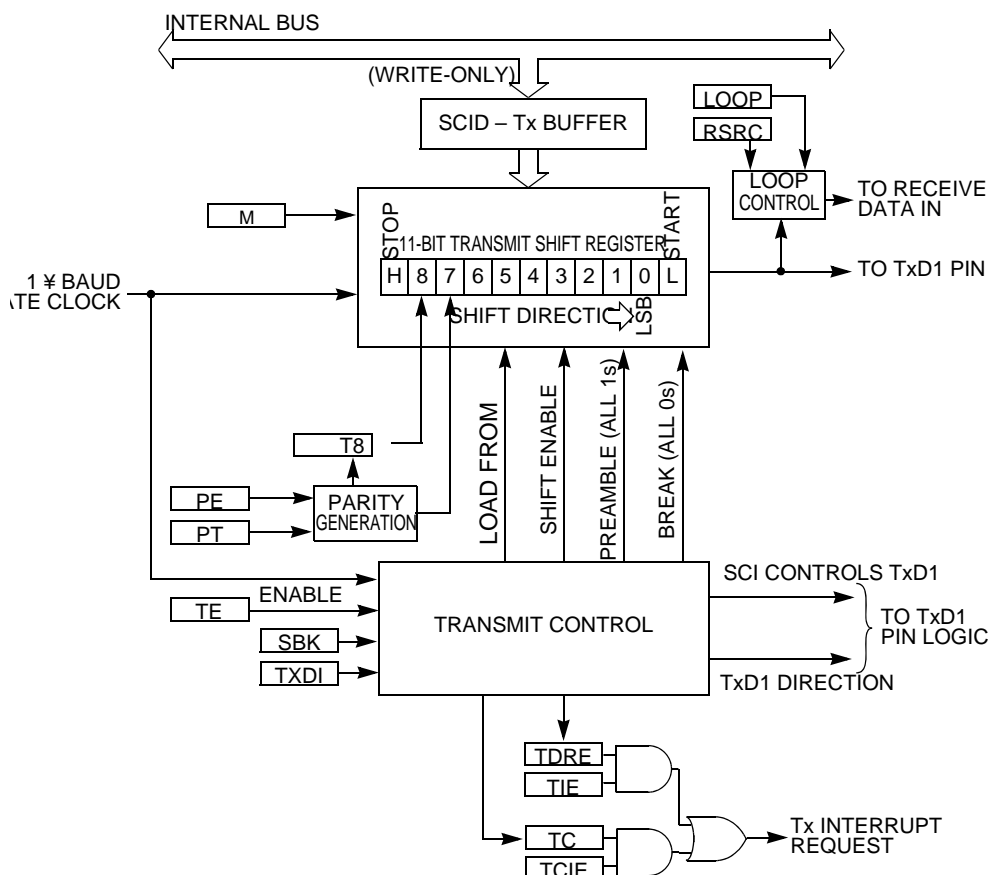


Figure 18-2. SCI Transmitter Block Diagram

The transmitter is enabled by setting the TE bit in SCIxC2. This queues a preamble character that is one full character frame of logic high. The transmitter then remains idle (TxD1 pin remains high) until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume  $M = 0$ , selecting the normal 8-bit data Mode. In 8-bit data Mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD1 pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD1 high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

## 18.4.2 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters that were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (including a 0 where the stop bit would be normally). Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight (or nine) data bits and a framing error ( $FE = 1$ ).

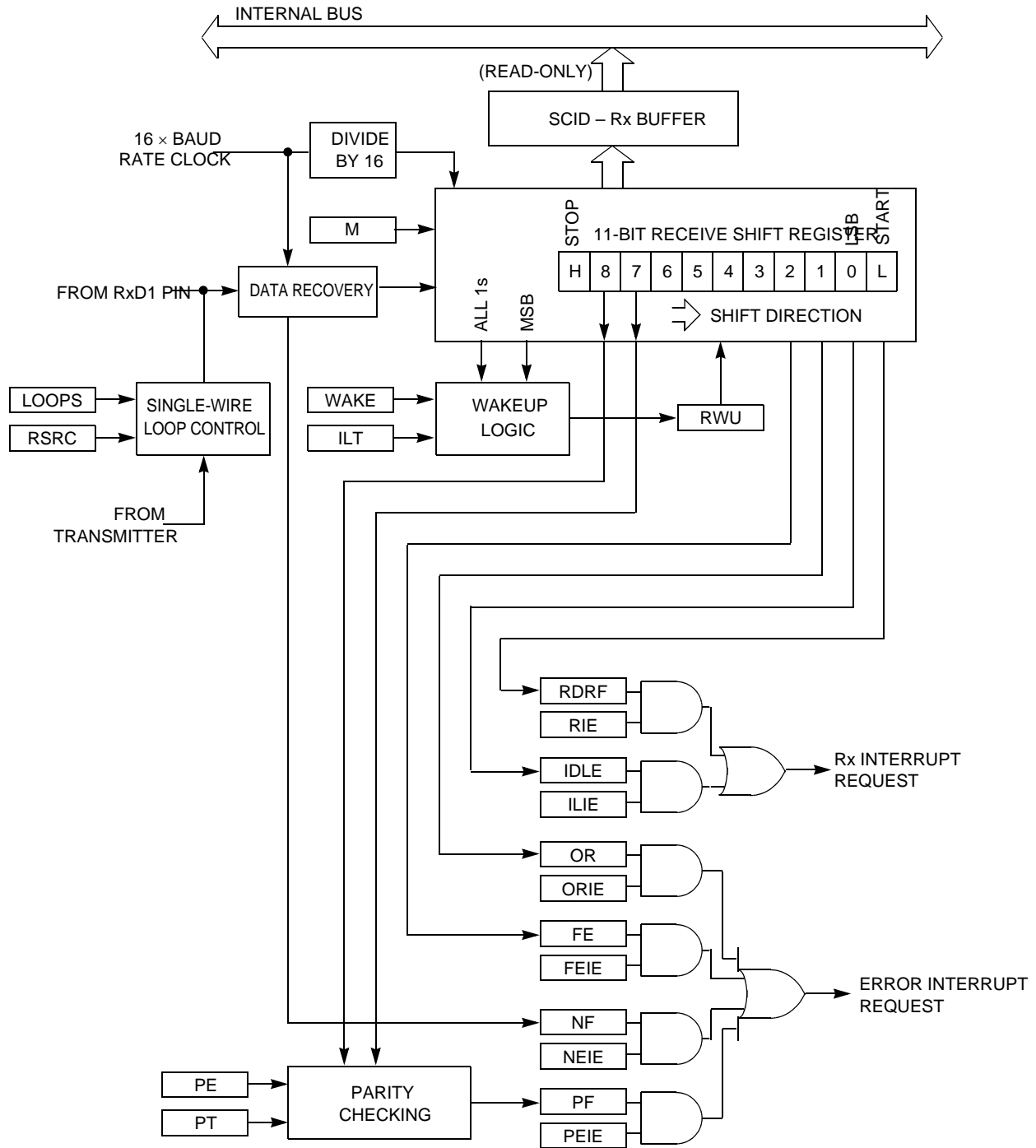
When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while  $TE = 0$ , the SCI transmitter never actually releases control of the TxD1 pin. If there is a possibility of the shifter finishing while  $TE = 0$ , set the general-purpose I/O controls so the pin that is shared with TxD1 is an output driving a logic 1. This ensures that the TxD1 line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

## 18.5 Receiver Functional Description

In this section, the receiver block diagram ([Figure 18-3](#)) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

### 18.5.1 Receiver Block Diagram

Figure 18-3 shows the receiver portion of the SCI.



**Figure 18-3. SCI Receiver Block Diagram**

The receiver is enabled by setting the RE bit in SCIXC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data Mode, refer



to [Section 18.7.1, “8-Bit and 9-Bit Data Modes”](#). For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data Mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence (which is normally satisfied in the course of the user’s program that handles receive data). Refer to [Section 18.6, “Interrupts and Status Flags”](#), for more details about flag clearing.

## 18.5.2 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 RxD1 serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The  $16\times$  baud rate clock is used to divide the bit time into 16 segments labelled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

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

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

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

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

### 18.5.3 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU = 1, it inhibits setting of the status flags associated with the receiver, 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.

#### 18.5.3.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data Mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits). The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter doesn't 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.

#### 18.5.3.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

## 18.6 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 and IDLE events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these eight interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD1 high. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1. Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading  $SCIxD$ . The  $RDRF$  flag is cleared by reading  $SCIxS1$  while  $RDRF = 1$  and then reading  $SCIxD$ . If the SCI is configured to operate in 9-bit Mode, an additional read to the  $SCIxC3$  register is required to clear  $RDRF$ .

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  $RxD1$  line remains idle for an extended period of time. IDLE is cleared by reading  $SCIxS1$  while  $IDLE = 1$  and then reading  $SCIxD$ . After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set  $RDRF$ .

If the associated error was detected in the received character that caused  $RDRF$  to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as  $RDRF$ . These flags are not set in overrun cases.

If  $RDRF$  was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead and the data and any associated NF, FE, or PF condition is lost.

## 18.7 Additional SCI Functions

The following sections describe additional SCI functions.

### 18.7.1 8-Bit and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit 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  $SCIxC3$ .

When transmitting 9-bit data, write to the T8 bit before writing to  $SCIxD$  for coherent writes to the transmit data buffer. 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.

When receiving 9-bit data, clear the  $RDRF$  bit by reading both R8 and  $SCIxD$ . R8 and  $SCIxD$  can be read in either order.

9-bit Mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

## 18.8 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.

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in Stop3 Mode). Software should ensure Stop Mode is not entered while there is a character being transmitted out of or received into the SCI module.

### 18.8.1 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between Loop Mode (RSRC = 0) or Single-wire Mode (RSRC = 1). Loop Mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD1 pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

### 18.8.2 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between Loop Mode (RSRC = 0) or Single-wire Mode (RSRC = 1). Single-wire Mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD1 pin. The RxD1 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 TxD1 pin. When TXDIR = 0, the TxD1 pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD1 pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD1 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.

## 18.9 SCI Registers and Control Bits

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 [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

## 18.9.1 SCI1 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).

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 18-4. SCI1 Baud Rate Register (SCIxBDH)**

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Write:								
Reset:	0	0	0	0	0	1	0	0

 = Unimplemented or Reserved

**Figure 18-5. SCI1 Baud Rate Register (SCIxBDL)**

### SBR12:SBR0 — Baud Rate Modulo Divisor

These 13 bits are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate =  $BUSCLK/(16 \times BR)$ .

## 18.9.2 SCI1 Control Register 1 (SCIxC1)

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Write:								
Reset:	0	0	0	0	0	0	0	0

**Figure 18-6. SCI1 Control Register 1 (SCIxC1)**

### LOOPS — Loop Mode Select

Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input.

- 1 = Loop Mode or Single-wire Mode where transmitter outputs are internally connected to receiver input. (See <st-blue>RSRC bit.) RxD1 pin is not used by SCI.
- 0 = Normal operation — RxD1 and TxD1 use separate pins.

### SCISWAI — SCI Stops in Wait Mode

- 1 = SCI clocks freeze while CPU is 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.

### RSRC — Receiver Source Select

This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD1 pin and RSRC determines whether this connection is also connected to the transmitter output.

- 1 = Single-wire SCI Mode where the TxD1 pin is connected to the transmitter output and receiver input.
- 0 = Provided LOOPS = 1, RSRC = 0 selects internal Loop Back Mode and the SCI does not use the RxD1 or TxD1 pins.

### M — 9-Bit or 8-Bit Mode Select

- 1 = Receiver and transmitter use 9-bit data characters  
start + 8 data bits (LSB first) + 9th data bit + stop.
- 0 = Normal — start + 8 data bits (LSB first) + stop.

### WAKE — Receiver Wakeup Method Select

Refer to [Section 18.5.3, “Receiver Wakeup Operation”](#), for more information.

- 1 = Address-mark wakeup.
- 0 = Idle-line wakeup.

### 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 the logic high level by the idle line detection logic. Refer to [Section 18.5.3.1, “Idle-Line Wakeup”](#), for more information.

- 1 = Idle character bit count starts after stop bit.
- 0 = Idle character bit count starts after start bit.

#### 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.

- 1 = Parity enabled.
- 0 = No hardware parity generation or checking.

#### 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.

- 1 = Odd parity.
- 0 = Even parity.

### 18.9.3 SCI1 Control Register 2 (SCIxC2)

This register can be read or written at any time.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0

**Figure 18-7. SCI1 Control Register 2 (SCIxC2)**

#### TIE — Transmit Interrupt Enable (for TDRE)

- 1 = Hardware interrupt requested when TDRE flag is 1.
- 0 = Hardware interrupts from TDRE disabled (use polling).

#### TCIE — Transmission Complete Interrupt Enable (for TC)

- 1 = Hardware interrupt requested when TC flag is 1.
- 0 = Hardware interrupts from TC disabled (use polling).

#### RIE — Receiver Interrupt Enable (for RDRF)

- 1 = Hardware interrupt requested when RDRF flag is 1.
- 0 = Hardware interrupts from RDRF disabled (use polling).

#### ILIE — Idle Line Interrupt Enable (for IDLE)

- 1 = Hardware interrupt requested when IDLE flag is 1.
- 0 = Hardware interrupts from IDLE disabled (use polling).

**TE — Transmitter Enable**

- 1 = Transmitter on.
- 0 = Transmitter off.

TE must be 1 in order to use the SCI transmitter. Normally, when TE = 1, the SCI forces the TxD1 pin to act as an output for the SCI system. If LOOPS = 1 and RSRC = 0, the TxD1 pin reverts to being a port B general-purpose I/O pin even if TE = 1.

When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD1 pin).

TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to [Section 18.4.2, “Send Break and Queued Idle”](#), for more details.

When TE is written to 0, the transmitter keeps control of the port TxD1 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.

**RE — Receiver Enable**

When the SCI receiver is off, the RxD1 pin reverts to being a general-purpose port I/O pin.

- 1 = Receiver on.
- 0 = Receiver off.

**RWU — Receiver Wakeup Control**

This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to [Section 18.5.3, “Receiver Wakeup Operation”](#), for more details.

- 1 = SCI receiver in standby waiting for wakeup condition.
- 0 = Normal SCI receiver operation.

**SBK — Send Break**

Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to [Section 18.4.2, “Send Break and Queued Idle”](#), for more details.

- 1 = Queue break character(s) to be sent.
- 0 = Normal transmitter operation.



## 18.9.4 SCI1 Status Register 1 (SCIxS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (that do not involve writing to this register) are used to clear these status flags.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write:								
Reset:	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

**Figure 18-8. SCI1 Status Register 1 (SCIxS1)**

### 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 SCIxS1 with TDRE = 1 and then write to the SCI data register (SCIxD).

- 1 = Transmit data register (buffer) empty.
- 0 = Transmit data register (buffer) full.

### TC — Transmission Complete Flag

TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.

- 1 = Transmitter idle (transmission activity complete).
- 0 = Transmitter active (sending data, a preamble, or a break).

TC is cleared automatically by reading SCIxS1 with TC = 1 and then doing one of the following:

- Write to the SCI data register (SCIxD) to transmit new data
- Queue a preamble by changing TE from 0 to 1
- Queue a break character by writing 1 to SBK in SCIxC2

### RDRF — Receive Data Register Full Flag

RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). In 8-bit Mode, to clear RDRF, read SCIxS1 with RDRF = 1 and then read the SCI data register (SCIxD). In 9-bit Mode, to clear RDRF, read SCIxS1 with RDRF = 1 and then read SCIxD and the SCI control 3 register (SCIxC3). SCIxD and SCIxC3 can be read in any order, but the flag is cleared only after both data registers are read.

- 1 = Receive data register full.
- 0 = Receive data register empty.

### IDLE — Idle Line Flag

IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or

11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When  $ILT = 1$ , the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.

To clear IDLE, read SCIxS1 with  $IDLE = 1$  and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will be set only once even if the receive line remains idle for an extended period.

1 = Idle line was detected.

0 = No idle line detected.

#### 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 SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with  $OR = 1$  and then read the SCI data register (SCIxD).

1 = Receive overrun (new SCI data lost).

0 = No overrun.

#### NF — Noise Flag

The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD).

1 = Noise detected in the received character in SCIxD.

0 = No noise detected.

#### 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 SCIxS1 with  $FE = 1$  and then read the SCI data register (SCIxD).

1 = Framing error.

0 = No framing error detected. This does not guarantee the framing is correct.

#### 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 SCIxS1 and then read the SCI data register (SCIxD).


1 = Parity error.

0 = No parity error.

### 18.9.5 SCI1 Status Register 2 (SCIxS2)

This register has one read-only status flag. Writes have no effect.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	RAF
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 18-9. SCI1 Status Register 2 (SCIxS2)**

#### 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.

1 = SCI receiver active (RxD1 input not idle).

0 = SCI receiver idle waiting for a start bit.

### 18.9.6 SCI1 Control Register 3 (SCIxC3)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	TXDIR	0	ORIE	NEIE	FEIE	PEIE
Write:								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 18-10. SCI1 Control Register 3 (SCIxC3)**

#### 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 SCIxD register. When reading 9-bit data, both R8 and SCIxD must be read to complete the automatic RDRF clearing sequence.

#### 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 SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written.

**TXDIR** — TxD1 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 TxD1 pin.

- 1 = TxD1 pin is an output in Single-wire Mode.
- 0 = TxD1 pin is an input in Single-wire Mode.

**ORIE** — Overrun Interrupt Enable

This bit enables the overrun flag (OR) to generate hardware interrupt requests.

- 1 = Hardware interrupt requested when OR = 1.
- 0 = OR interrupts disabled (use polling).

**NEIE** — Noise Error Interrupt Enable

This bit enables the noise flag (NF) to generate hardware interrupt requests.

- 1 = Hardware interrupt requested when NF = 1.
- 0 = NF interrupts disabled (use polling).

**FEIE** — Framing Error Interrupt Enable

This bit enables the framing error flag (FE) to generate hardware interrupt requests.

- 1 = Hardware interrupt requested when FE = 1.
- 0 = FE interrupts disabled (use polling).

**PEIE** — Parity Error Interrupt Enable

This bit enables the parity error flag (PF) to generate hardware interrupt requests.

- 1 = Hardware interrupt requested when PF = 1.
- 0 = PF interrupts disabled (use polling).

### 18.9.7 SCI1 Data Register (SCIxD)

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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	0	0	0	0	0	0	0	0

**Figure 18-11. SCI1 Data Register (SCIxD)**

# Chapter 19

## Inter-Integrated Circuit (IIC)

### 19.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of  $\text{clock}/20$ , 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.

For additional detail, please refer to volume 1 of the *HCS08 Reference Manual*, (Freescale Semiconductor document order number HCS08RMv1).

#### 19.1.1 Features

The IIC includes these distinctive features:

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

#### 19.1.2 Modes of Operation

The IIC functions the same in Normal and Monitor Modes. A brief description of the IIC in the various MCU modes is given here.

- Run Mode — This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait Mode — The module will continue to operate while the MCU is in Wait Mode and can provide a wake-up interrupt.
- Stop Mode — The IIC is inactive in Stop3 Mode for reduced power consumption. The STOP instruction does not affect IIC register states. Stop1 and Stop2 will reset the register contents.

### 19.1.3 Block Diagram

Figure 19-1 is a block diagram of the IIC.

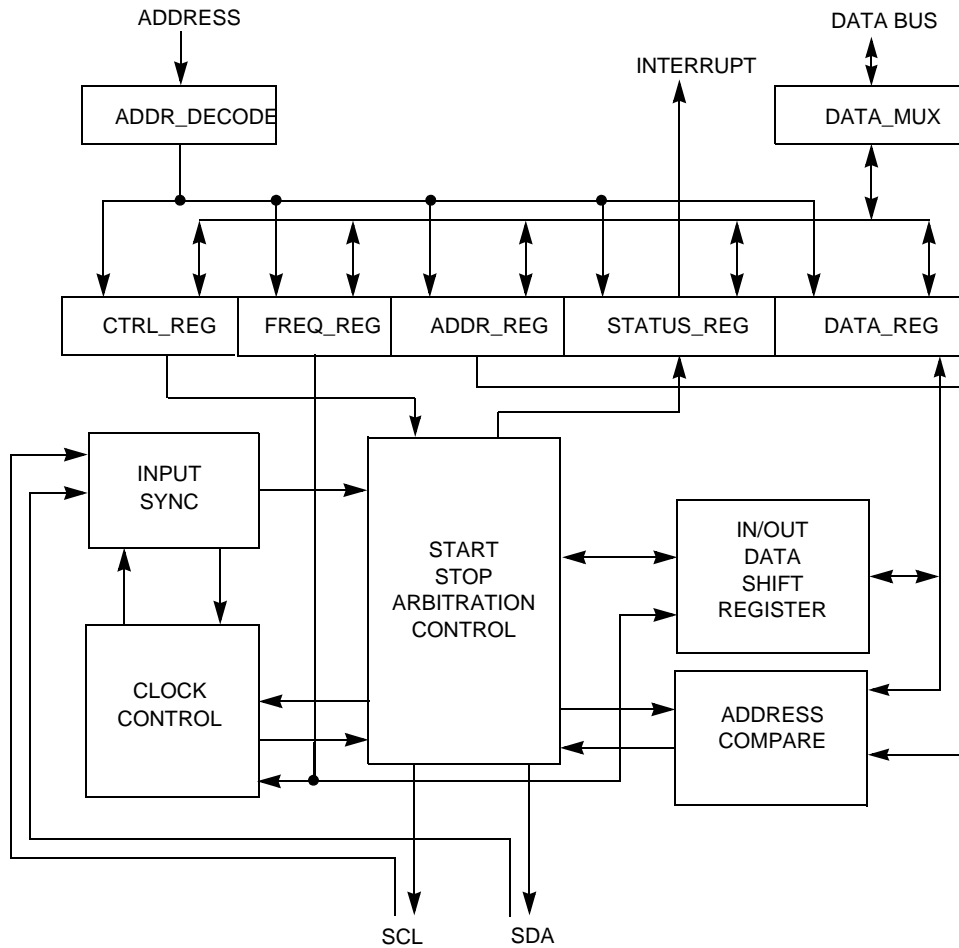


Figure 19-1. IIC Functional Block Diagram

### 19.2 Signal Description

Table 19-1 shows the user-accessible signals for the IIC.

Table 19-1. Signal Properties

Name	Function
SCL	Serial clock line
SDA	Serial data line

## 19.3 External Signal Description

This section describes each user-accessible pin signal.

### 19.3.1 SCL — Serial Clock Line

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

### 19.3.2 SDA — Serial Data Line

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

## 19.4 Register Definitions

This section provides a detailed description of all IIC registers accessible to the end user.

### 19.4.1 Module Memory Map

The IIC has five 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. [Figure 19-2](#) summarizes the IIC module's address space. The following section describes the bit-level arrangement and functionality of each register.

**Table 19-2. Module Memory Map**

Address	Use	Access
Base + \$0000	IIC Address Register (IICA)	read/write
Base + \$0001	IIC Frequency Divider Register (IICF)	read/write
Base + \$0002	IIC Control Register (IICC)	read/write
Base + \$0003	IIC Status Register (IICS)	read
Base + \$0004	IIC Data IO Register (IICD)	read/write

## 19.4.2 Register Descriptions

This section consists of the IIC register descriptions in address order. Refer to the direct-page register summary in [Chapter 11, “MCU 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.

### 19.4.3 IIC Address Register (IIC1A)



Figure 19-2. IIC Address Register (IIC1A)

Table 19-3. IIC1A Register Field Descriptions

Field	Description
7:1 ADDR[7:1]	<b>IIC Address Register</b> — The ADDR contains the specific slave address to be used by the IIC module. This is the address the module will respond to when addressed as a slave.

### 19.4.4 IIC Frequency Divider Register (IIC1F)

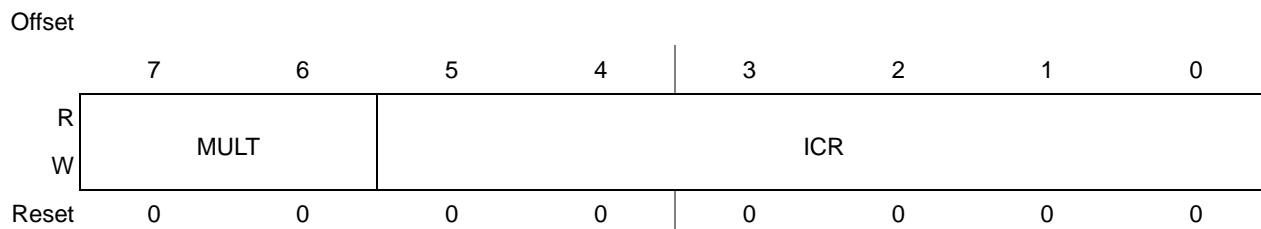


Figure 19-3. IIC Frequency Divider Register (IIC1F)



**Table 19-4. IIC1F Register Field Descriptions**

Field	Description
7:6 MULT	<p><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.</p> <p>00 mul = 01                      01 mul = 02                      10 mul = 04                      11 Reserved</p>
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 are used to define the SCL divider and the SDA hold value. The SCL divider multiplied by the value provided by the MULT register (multiplier factor mul) is used to generate IIC baud rate.</p> <p>IIC baud rate = bus speed (Hz)/(mul * SCL divider)</p> <p>SDA hold time is the delay from the falling edge of the SCL (IIC clock) to the changing of SDA (IIC data). The ICR is used to determine the SDA hold value.</p> <p>SDA hold time = bus period (s) * SDA hold value</p> <p>Table 19-5 provides the SCL divider and SDA hold values for corresponding values of the ICR. These values can be used to set IIC baud rate and SDA hold time. For example:</p> <p>Bus speed = 8 MHz                      MULT is set to 01 (mul = 2)                      Desired IIC baud rate = 100 kbps</p> <p>IIC baud rate = bus speed (Hz)/(mul * SCL divider)                      100000 = 8000000/(2*SCL divider)                      SCL divider = 40</p> <p>Table 19-5 shows that ICR must be set to 0B to provide an SCL divider of 40 and that this will result in an SDA hold value of 9.</p> <p>SDA hold time = bus period (s) * SDA hold value                      SDA hold time = 1/8000000 * 9 = 1.125 μs</p> <p>If the generated SDA hold value is not acceptable, the MULT bits can be used to change the ICR. This will result in a different SDA hold value.</p> <p><b>ICR — IIC Clock Rate Bits</b></p>

**Table 19-5. IIC Divider and Hold Values**

ICRTAP2-TAP1 (hex)	SCL Divider	SDA Hold Value	TAP2-TAP1IC R (hex)	SCL Divider	SDA Hold Value
00	20	7	20	160	17
01	22	7	21	192	17
02	24	8	22	224	33
03	26	8	23	256	33
04	28	9	24	288	49
05	30	9	25	320	49
06	34	10	26	384	65
07	40	10	27	480	65
08	28	7	28	320	33
09	32	7	29	384	33
0A	36	9	2A	448	65
0B	40	9	2B	512	65
0C	44	11	2C	576	97
0D	48	11	2D	640	97
0E	56	13	2E	768	129
0F	68	13	2F	960	129
10	48	9	30	640	65
11	56	9	31	768	65
12	64	13	32	896	129
13	72	13	33	1024	129
14	80	17	34	1152	193
15	88	17	35	1280	193
16	104	21	36	1536	257
17	128	21	37	1920	257
18	80	9	38	1280	129
19	96	9	39	1536	129
1A	112	17	3A	1792	257
1B	128	17	3B	2048	257
1C	144	25	3C	2304	385
1D	160	25	3D	2560	385
1E	192	33	3E	3072	513
1F	240	33	3F	3840	513

## 19.4.5 IIC Control Register (IIC1C)

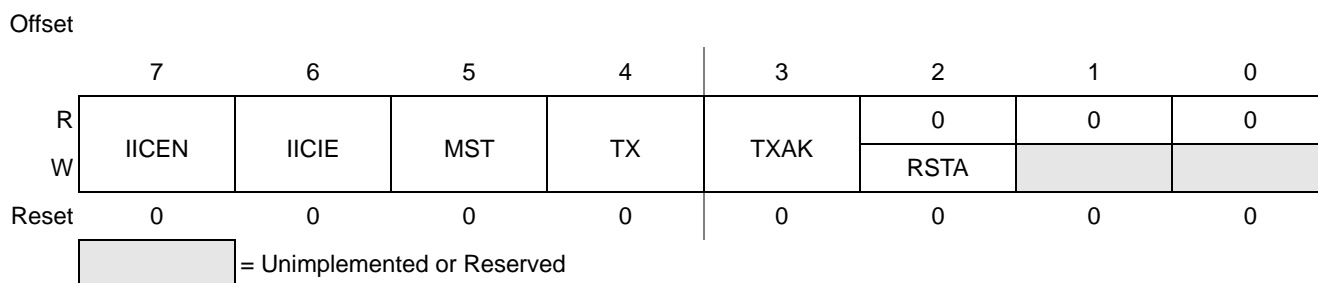


Figure 19-4. IIC Control Register (IIC1C)

Table 19-6. IIC1C Register 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> — The MST bit is changed from a 0 to a 1 when a START signal is generated on the bus and Master Mode is selected. When this bit changes from a 1 to a 0 a STOP signal is generated and the mode of operation changes from Master to Slave. 0 Slave Mode. 1 Master Mode.
4 TX	<b>Transmit Mode Select</b> — The TX bit selects the direction of master and slave transfers. In Master Mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. 0 Receive. 1 Transmit.
3 TXAK	<b>Transmit Acknowledge Enable</b> — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. 0 An acknowledge signal will be sent out to the bus after receiving one data byte. 1 No acknowledge signal response is sent.
2 RSTA	<b>Repeat START</b> — Writing a one to this bit will generate a repeated START condition provided it is the current master. This bit will always be read as a low. Attempting a repeat at the wrong time will result in loss of arbitration.

## 19.4.6 IIC Status Register (IIC1S)

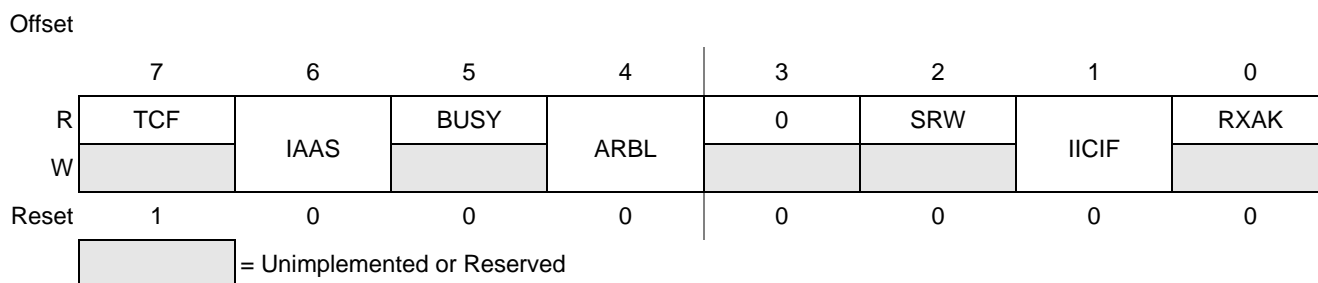


Figure 19-5. IIC Status Register (IIC1S)

Table 19-7. IIC1S Register Field Descriptions

Field	Description
7 TCF	<b>Transfer Complete Flag</b> — This bit is set on the completion of a byte 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 IIC1D register in Receive Mode or writing to the IIC1D in Transmit Mode. 0 Transfer in progress. 1 Transfer complete.
6 IAAS	<b>Addressed as a Slave</b> — The IAAS bit is set when its own specific address is matched with the calling address. Writing the IIC1C register clears this bit. 0 Not addressed. 1 Addressed as a slave.
5 BUSY	<b>Bus Busy</b> — The BUSY bit indicates the status of the bus regardless of Slave or Master Mode. The BUSY bit is set when a START signal is detected and cleared when a STOP signal is detected. 0 Bus is idle. 1 Bus is busy.
4 ARBL	<b>Arbitration Lost</b> — This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing a one to it. 0 Standard bus operation. 1 Loss of arbitration.
2 SRW	<b>Slave Read/Write</b> — When addressed as a slave the SRW bit indicates the value of the R/W command bit of the calling address sent to the master. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.
1 IICIF	<b>IIC Interrupt Flag</b> — The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a one 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 completes</li> <li>• Match of slave address to calling address</li> <li>• Arbitration lost</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.

## 19.4.7 IIC Data I/O Register (IIC1D)

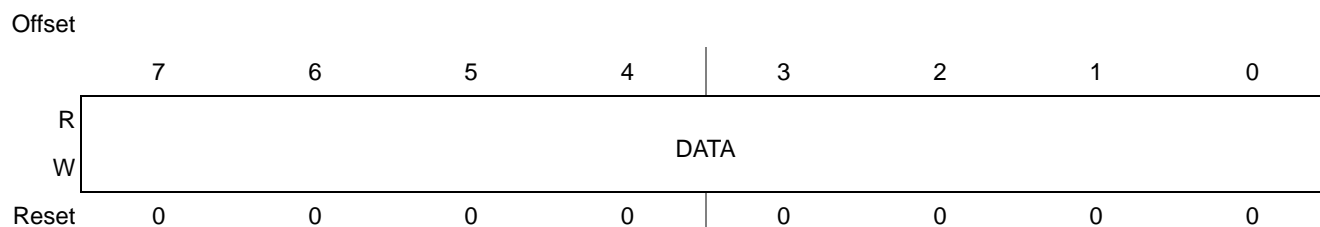


Figure 19-6. IIC Data I/O Register (IIC1D)

Table 19-8. IIC1D Register Field Descriptions

Field	Description
7:0 DATA	<b>Data</b> — In Master Transmit Mode, when data is written to the IIC1D, a data transfer is initiated. The most significant bit is sent first. In Master Receive Mode, reading this register initiates receiving of the next byte of data.

### NOTE

When transitioning out of Master Receive Mode, the IIC Mode should be switched before reading the IIC1D register to prevent an inadvertent initiation of a master receive data transfer.

In Slave Mode, the same functions are available after an address match has occurred.

Note that the TX bit in IIC1C 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 IIC1D will not initiate the receive.

Reading the IIC1D will return the last byte received while the IIC is configured in either Master Receive or Slave Receive Modes. The IIC1D does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IIC1D correctly by reading it back.

In Master Transmit Mode, the first byte of data written to IIC1D following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7–bit 1) concatenated with the required R/W bit (in position bit 0).

## 19.5 Functional Description

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

### 19.5.1 IIC Protocol

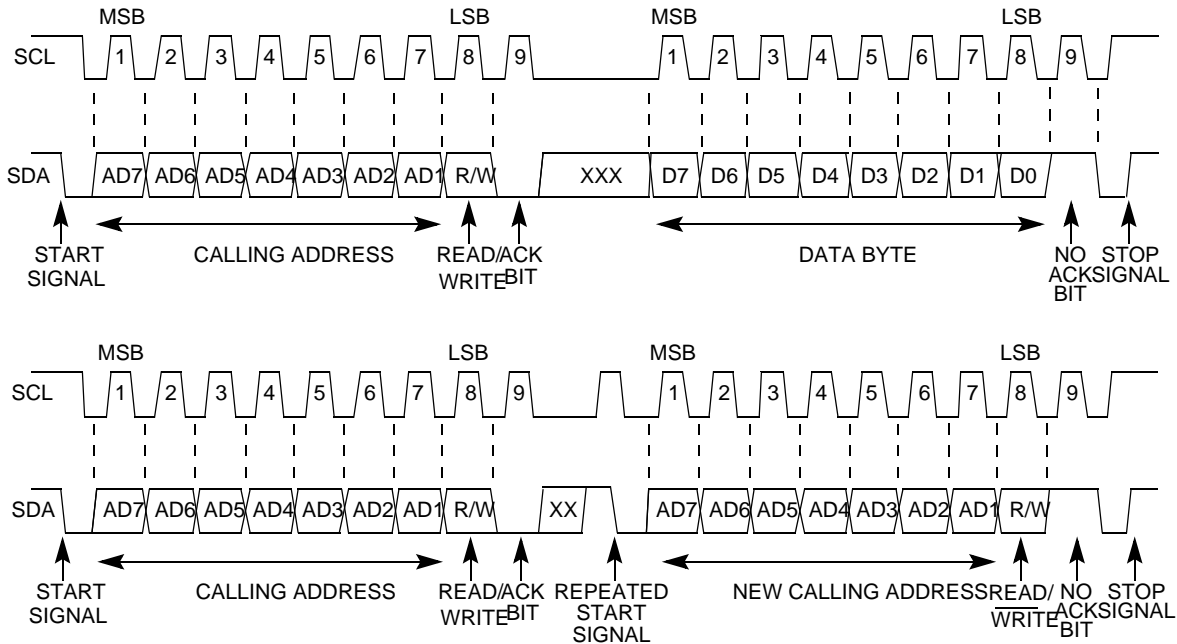
The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts:

- START signal

- Slave address transmission
- Data transfer
- STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The IIC bus system communication is described briefly in the following sections and illustrated in [Figure 19-7](#).



**Figure 19-7. IIC Bus Transmission Signals**

### 19.5.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in [Figure 19-7](#), 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.

### 19.5.1.2 Slave Address Transmission

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

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

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see [Figure 19-7](#)).

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

### 19.5.1.3 Data Transfer

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

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

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in [Figure 19-7](#). There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

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

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

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.

### 19.5.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see [Figure 19-7](#)).

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

### 19.5.1.5 Repeated START Signal

As shown in [Figure 19-7](#), 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.

### 19.5.1.6 Arbitration Procedure

The IIC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus

clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to Slave Receive Mode and stop driving SDA output. In this case, the transition from Master to Slave Mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

### 19.5.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 19-8). 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.

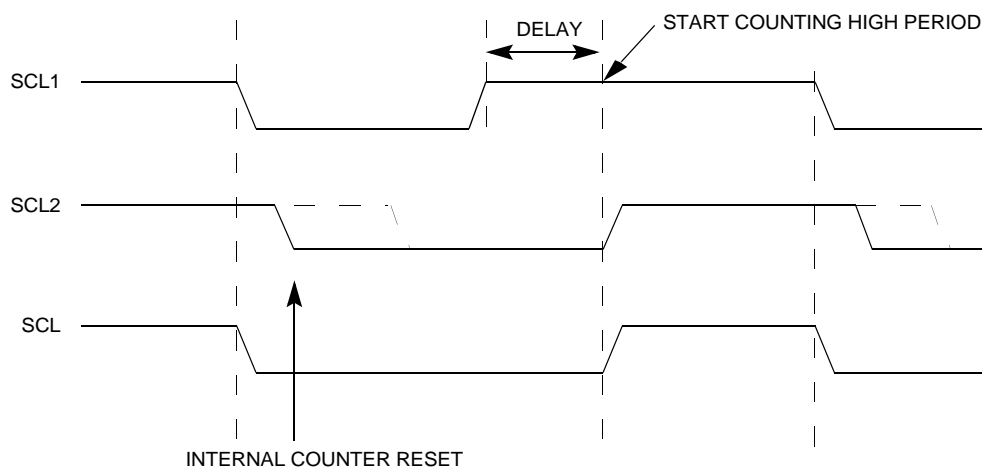


Figure 19-8. IIC Clock Synchronization

### 19.5.1.8 Handshaking

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

### 19.5.1.9 Clock Stretching

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



## 19.6 Resets

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

## 19.7 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in [Table 19-9](#) occur provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a one to it in the interrupt routine. The user can determine the interrupt type by reading the status register.

**Table 19-9. Interrupt Summary**

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

### 19.7.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

### 19.7.2 Address Detect Interrupt

When its own specific address (IIC address register) is matched with the calling address, the IAAS bit in 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.

### 19.7.3 Arbitration Lost Interrupt

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

Arbitration is lost in the following circumstances:

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

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



# Chapter 20

## Analog to Digital (ATD) Module

### 20.1 Introduction

The ATD module is an analog-to-digital converter with a successive approximation register (SAR) architecture with sample and hold.

#### 20.1.1 Features

- 8-/10-bit resolution
- 14.0  $\mu$ sec, 10-bit single conversion time at a conversion frequency of 2 MHz
- Left-/right-justified result data
- Left-justified Signed Data Mode
- Conversion complete flag or conversion complete interrupt generation
- Analog input multiplexer for up to eight analog input channels
- Single or Continuous Conversion Mode

#### 20.1.2 Modes of Operation

The ATD has two modes for low power

- Stop Mode
- Power-Down Mode

##### 20.1.2.1 Stop Mode

When the MCU goes into Stop Mode, the MCU stops the clocks and the ATD analog circuitry is turned off, placing the module into a low-power state. Once in Stop Mode, the ATD module aborts any single or continuous conversion in progress. Upon exiting Stop Mode, no conversions occur and the registers have their previous values. As long as the ATDPU bit is set prior to entering Stop Mode, the module is reactivated coming out of stop.

##### 20.1.2.2 Power Down Mode

Clearing the ATDPU bit in register ATD1C also places the ATD module in a low-power state. The ATD conversion clock is disabled and the analog circuitry is turned off, placing the module in Power-Down Mode. (This mode does not remove power to the ATD module.) Once in Power-Down Mode, the ATD module aborts any conversion in progress. Upon setting the ATDPU bit, the module is reactivated. During Power-Down Mode, the ATD registers are still accessible.

Note that the reset state of the ATDPU bit is zero. Therefore, the module is reset into the power-down state.

### 20.1.3 Block Diagram

Figure 20-1 shows the functional structure of the ATD module.

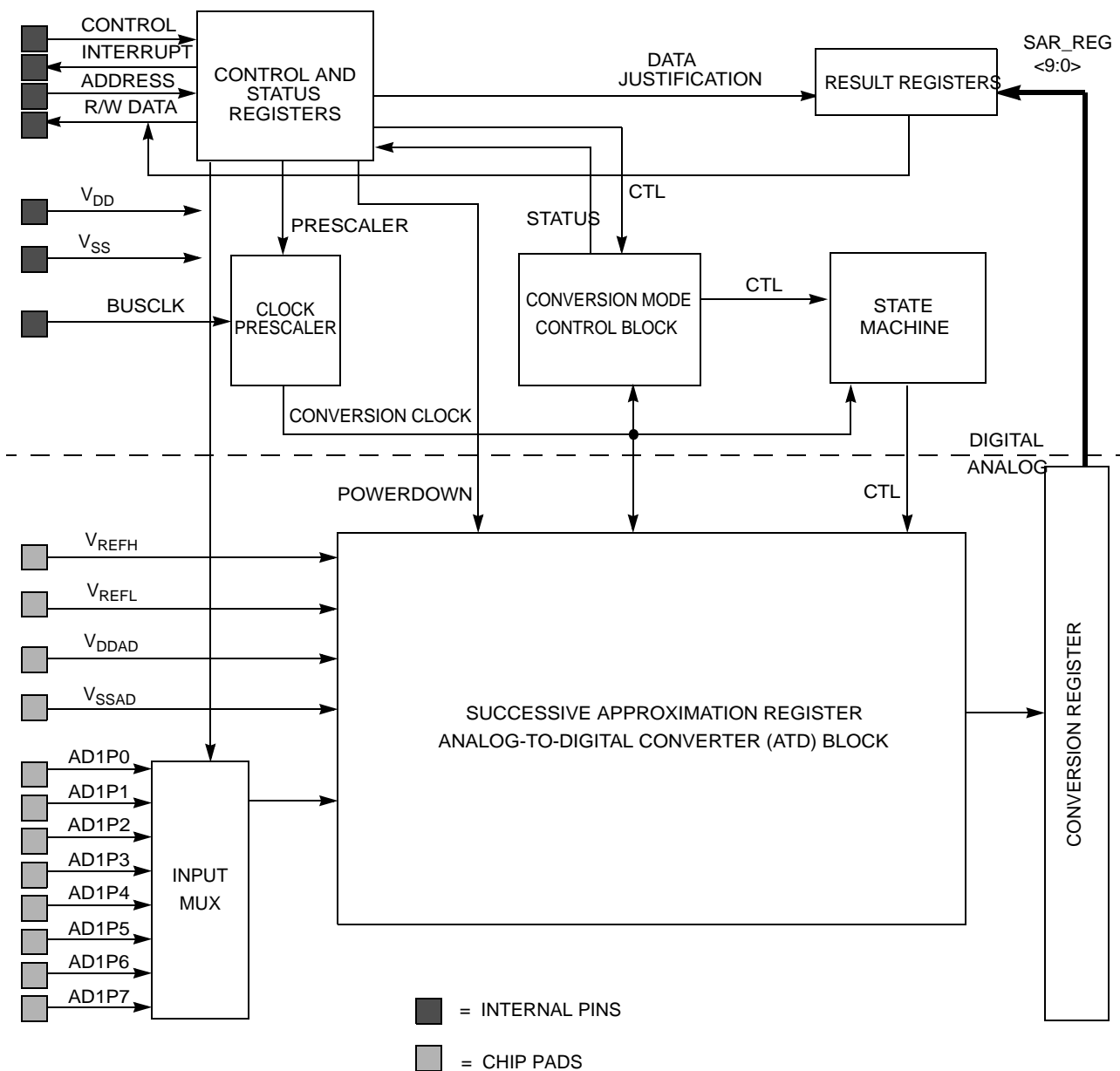


Figure 20-1. ATD Block Diagram

## 20.2 Signal Description

### 20.2.1 Overview

The ATD supports eight input channels and requires 4 supply/reference/ground pins. These pins are listed in [Table 20-1](#).

**Table 20-1. Signal Properties**

Name	Function
AD7–AD0	Channel input pins
V <sub>REFH</sub>	High reference voltage for ATD converter
V <sub>REFL</sub>	Low reference voltage for ATD converter
V <sub>DDAD</sub>	ATD power supply voltage
V <sub>SSAD</sub>	ATD ground supply voltage

#### 20.2.1.1 Channel Input Pins — AD1P7–AD1P0

The channel pins are used as the analog input pins of the ATD. Each pin is connected to an analog switch which serves as the signal gate into the sample sub module.

#### 20.2.1.2 ATD Reference Pins — V<sub>REFH</sub>, V<sub>REFL</sub>

These pins serve as the source for the high and low reference potentials for the converter. Separation from the power supply pins accommodates the filtering necessary to achieve the accuracy of which the system is capable.

#### 20.2.1.3 ATD Supply Pins — V<sub>DDAD</sub>, V<sub>SSAD</sub>

These two pins are used to supply power and ground to the analog section of the ATD. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on digital power supplies.

#### NOTE

V<sub>DDAD1</sub> and V<sub>DD</sub> must be at the same potential. Likewise, V<sub>SSAD1</sub> and V<sub>SS</sub> must be at the same potential.

## 20.3 Functional Description

The ATD uses a successive approximation register (SAR) architecture. The ATD contains all the necessary elements to perform a single analog-to-digital conversion.

A write to the ATD1SC register initiates a new conversion. A write to the ATD1C register will interrupt the current conversion but it will not initiate a new conversion. A write to the ATD1PE register will also abort the current conversion but will not initiate a new conversion. If a conversion is already running when a write to the ATD1SC register is made, it will be aborted and a new one will be started.

## 20.3.1 Mode Control

The ATD has a mode control unit to communicate with the sample and hold (S/H) machine and the SAR machine when necessary to collect samples and perform conversions. The mode control unit signals the S/H machine to begin collecting a sample and for the SAR machine to begin receiving a sample. At the end of the sample period, the S/H machine signals the SAR machine to begin the analog-to-digital conversion process. The conversion process is terminated when the SAR machine signals the end of conversion to the mode control unit. For  $V_{REFL}$  and  $V_{REFH}$ , the SAR machine uses the reference potentials to set the sampled signal level within itself without relying on the S/H machine to deliver them.

The mode control unit organizes the conversion, specifies the input sample channel, and moves the digital output data from the SAR register to the result register. The result register consists of a dual-port register. The SAR register writes data into the register through one port while the module data bus reads data out of the register through the other port.

## 20.3.2 Sample and Hold

The S/H machine accepts analog signals and stores them as capacitor charge on a storage node located in the SAR machine. Only one sample can be held at a time so the S/H machine and the SAR machine can not run concurrently even though they are independent machines. No DC level correction, channel-to-channel gain adjust, or other data acquisition functions are currently performed on the input analog signals. Figure 20-2 shows the placement of the various resistors and capacitors.

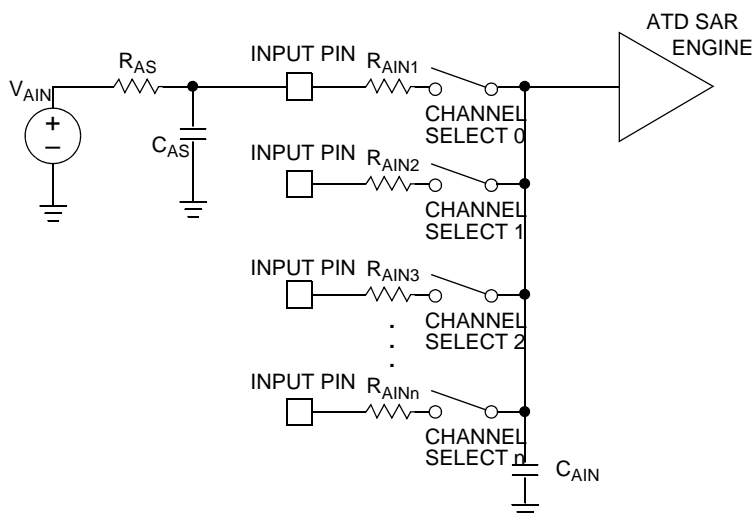


Figure 20-2. Resistor and Capacitor Placement

When the S/H machine is not sampling, it disables its own internal clocks. The input analog signals are unipolar. The signals must fall within the potential range of  $V_{SSAD}$  to  $V_{DDAD}$ . The S/H machine is not required to perform special conversions (i.e., convert  $V_{REFL}$  and  $V_{REFH}$ ).

Proper sampling is dependent on the following factors:

- Analog source impedance (the real portion,  $R_{AS}$ )  
This is the resistive (or real, in the case of high frequencies) portion of the network driving the analog input voltage  $V_{AIN}$ .

- Analog source capacitance ( $C_{AS}$ ) — This is the filtering capacitance on the analog input, which (if large enough) may help the analog source network charge the ATD input in the case of high  $R_{AS}$ .
- ATD input resistance ( $R_{AIN}$  – maximum value 7 k $\Omega$ ) — This is the internal resistance of the ATD circuit in the path between the external ATD input and the ATD sample and hold circuit. This resistance varies with temperature, voltage, and process variation but a worst case number is necessary to compute worst case sample error.
- ATD input capacitance ( $C_{AIN}$  – maximum value 50 pF) — This is the internal capacitance of the ATD sample and hold circuit. This capacitance varies with temperature, voltage, and process variation but a worst case number is necessary to compute worst case sample error.
- ATD conversion clock frequency ( $f_{ATDCLK}$  – maximum value 2 MHz) — This is the frequency of the clock input to the ATD and is dependent on the bus clock frequency and the ATD prescaler. This frequency determines the width of the sample window, which is 14 ATDCLK cycles.
- Input sample frequency ( $f_{SAMP}$ ) — This is the frequency that a given input is sampled.
- Delta-input sample voltage ( $\Delta V_{SAMP}$ ) — This is the difference between the current input voltage (intended for conversion) and the previously sampled voltage (which may be from a different channel). In Non-continuous Convert Mode, this is assumed to be the greater of ( $V_{REFH} - V_{AIN}$ ) and ( $V_{AIN} - V_{REFL}$ ). In Continuous Convert Mode, 5 LSB should be added to the known difference to account for leakage and other losses.
- Delta-analog input voltage ( $\Delta V_{AIN}$ ) — This is the difference between the current input voltage and the input voltage during the last conversion on a given channel. This is based on the slew rate of the input.

In cases where there is no external filtering capacitance, the sampling error is determined by the number of time constants of charging and the change in input voltage relative to the resolution of the ATD:

$$\text{\# of time constants } (\tau) = (14 / f_{ATDCLK}) / ((R_{AS} + R_{AIN}) * C_{AIN}) \quad \text{Eqn. 20-1}$$

$$\text{sampling error in LSB } (E_S) = 2^N * (\Delta V_{SAMP} / (V_{REFH} - V_{REFL})) * e^{-\tau}$$

The maximum sampling error (assuming maximum change on the input voltage) will be:

$$E_S = (3.6/3.6) * e^{-(14/((7 \text{ k} + 10 \text{ k}) * 50 \text{ p} * 2 \text{ M}))} * 1024 = 0.271 \text{ LSB} \quad \text{Eqn. 20-2}$$

In the case where an external filtering capacitance is applied, the sampling error can be reduced based on the size of the source capacitor ( $C_{AS}$ ) relative to the analog input capacitance ( $C_{AIN}$ ). Ignoring the analog source impedance ( $R_{AS}$ ),  $C_{AS}$  will charge  $C_{AIN}$  to a value of:

$$E_S = 2^N * (\Delta V_{SAMP} / (V_{REFH} - V_{REFL})) * (C_{AIN} / (C_{AIN} + C_{AS})) \quad \text{Eqn. 20-3}$$

In the case of a 0.1  $\mu\text{F}$   $C_{AS}$ , a worst case sampling error of 0.5 LSB is achieved regardless of  $R_{AS}$ . However, in the case of repeated conversions at a rate of  $f_{SAMP}$   $R_{AS}$  must re-charge  $C_{AS}$ . This recharge is continuous and controlled only by  $R_{AS}$  (not  $R_{AIN}$ ), and reduces the overall sampling error to:

$$E_S = 2^N * \{(\Delta V_{AIN} / (V_{REFH} - V_{REFL})) * e^{-(1 / (f_{SAMP} * R_{AS} * C_{AS}))} + (\Delta V_{SAMP} / (V_{REFH} - V_{REFL})) * \text{Min}[(C_{AIN} / (C_{AIN} + C_{AS})), e^{-(1 / (f_{ATDCLK} * (R_{AS} + R_{AIN}) * C_{AIN}))}]\} \quad \text{Eqn. 20-4}$$

This is a worst case sampling error which does not account for  $R_{AS}$  recharging the combination of  $C_{AS}$  and  $C_{AIN}$  during the sample window. It does illustrate that high values of  $R_{AS}$  (>10 k $\Omega$ ) are possible if a

large  $C_{AS}$  is used and sufficient time to recharge  $C_{AS}$  is provided between samples. In order to achieve accuracy specified under the worst case conditions of maximum  $\Delta V_{SAMP}$  and minimum  $C_{AS}$ ,  $R_{AS}$  must be less than the maximum value of 10 k $\Omega$ . The maximum value of 10 k $\Omega$  for  $R_{AS}$  is to ensure low sampling error in the worst case condition of maximum  $\Delta V_{SAMP}$  and minimum  $C_{AS}$ .

### 20.3.3 Analog Input Multiplexer

The analog input multiplexer selects one of the eight external analog input channels to generate an analog sample. The analog input multiplexer includes negative stress protection circuitry which prevents cross-talk between channels when the applied input potentials are within specification. Only analog input signals within the potential range of  $V_{REFL}$  to  $V_{REFH}$  (ATD reference potentials) will result in valid ATD conversions.

### 20.3.4 ATD Module Accuracy Definitions

Figure 20-3 illustrates an ideal ATD transfer function. The horizontal axis represents the ATD input voltage in millivolts. The vertical axis the conversion result code. The ATD is specified with the following figures of merit:

- Number of bits (N) — The number of bits in the digitized output
- Resolution (LSB) — The resolution of the ATD is the step size of the ideal transfer function. This is also referred to as the ideal code width, or the difference between the transition voltages to a given code and to the next code. This unit, known as 1LSB, is equal to

$$1\text{LSB} = (V_{REFH} - V_{REFL}) / 2^N \quad \text{Eqn. 20-5}$$

- Inherent quantization error ( $E_Q$ ) — This is the error caused by the division of the perfect ideal straight-line transfer function into the quantized ideal transfer function with  $2^N$  steps. This error is  $\pm 1/2$  LSB.
- Differential non-linearity (DNL) — This is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to the current code and to the next code. A negative DNL means the transfer function spends less time at the current code than ideal; a positive DNL, more. The DNL cannot be less than  $-1.0$ ; a DNL of greater than 1.0 reduces the effective number of bits by 1.
- Integral non-linearity (INL) — This is the difference between the transition voltage to the current code and the transition to the corresponding code on the adjusted transfer curve. INL is a measure of how straight the line is (how far it deviates from a straight line). The adjusted ideal transition voltage is:

$$\text{Adjusted Ideal Trans. } V = \frac{(\text{Current Code} - 1/2)}{2^N} * ((V_{REFH} + E_{FS}) - (V_{REFL} + E_{ZS})) \quad \text{Eqn. 20-6}$$

- Zero scale error ( $E_{ZS}$ ) — This is the difference between the transition voltage to the first valid code and the ideal transition to that code. Normally, it is defined as the difference between the actual and



ideal transition to code \$001, but in some cases the first transition may be to a higher code. The ideal transition to any code is:

**Eqn. 20-7**

$$\text{Ideal Transition } V = \frac{(\text{Current Code} - 1/2)}{2^N} * (V_{\text{REFH}} - V_{\text{REFL}})$$

- Full scale error ( $E_{\text{FS}}$ ) — This is the difference between the transition voltage to the last valid code and the ideal transition to that code. Normally, it is defined as the difference between the actual and ideal transition to code \$3FF, but in some cases the last transition may be to a lower code. The ideal transition to any code is:

**Eqn. 20-8**

$$\text{Ideal Transition } V = \frac{(\text{Current Code} - 1/2)}{2^N} * (V_{\text{REFH}} - V_{\text{REFL}})$$

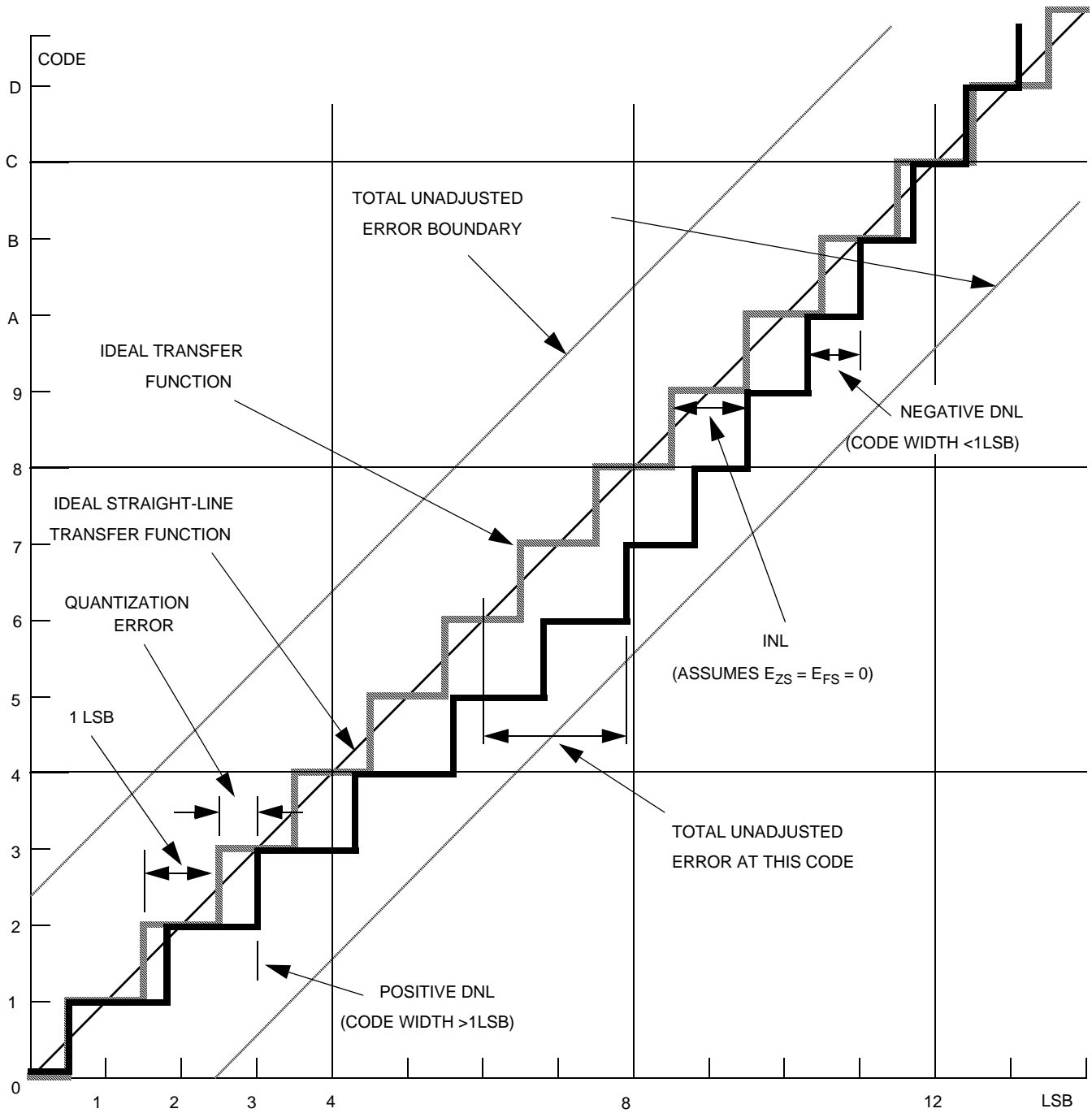
- Total unadjusted error ( $E_{\text{TU}}$ ) — This is the difference between the transition voltage to a given code and the ideal straight-line transfer function. An alternate definition (with the same result) is the difference between the actual transfer function and the ideal straight-line transfer function. This measure of error includes inherent quantization error and all forms of circuit error (INL, DNL, zero-scale, and full-scale) except input leakage error, which is not due to the ATD.
- Input leakage error ( $E_{\text{IL}}$ ) — This is the error between the transition voltage to the current code and the ideal transition to that code that is the result of input leakage across the real portion of the impedance of the network that drives the analog input. This error is a system-observable error which is not inherent to the ATD, so it is not added to total error. This error is:

$$E_{\text{IL}} \text{ (in V)} = \text{input leakage} * R_{\text{AS}}$$

**Eqn. 20-9**

There are two other forms of error which are not specified which can also affect ATD accuracy. These are:

- Sampling error ( $E_{\text{S}}$ ) — The error due to inadequate time to charge the ATD circuitry
- Noise error ( $E_{\text{N}}$ ) — The error due to noise on  $V_{\text{AIN}}$ ,  $V_{\text{REFH}}$ , or  $V_{\text{REFL}}$  due to either direct coupling (noise source capacitively coupled directly on the signal) or power supply ( $V_{\text{DDAD}}$ ,  $V_{\text{SSAD}}$ ,  $V_{\text{DD}}$ , and  $V_{\text{SS}}$ ) noise interfering with the ATD's ability to resolve the input accurately. The error due to internal sources can be reduced (and specified operation achieved) by operating the ATD conversion in Wait Mode and ceasing all IO activity. Reducing the error due to external sources is dependent on system activity and board layout.



NOTES: Graph is for example only and may not represent actual performance

**Figure 20-3. ATD Accuracy Definitions**

## 20.4 Resets

The ATD module is reset on system reset. If the system reset signal is activated, the ATD registers are initialized back to their reset state and the ATD module is powered down. This occurs as a function of the register file initialization; the reset definition of the ATDPU bit (power down bit) is zero or disabled.

The MCU places the module back into an initialized state. If the module is performing a conversion, the current conversion is terminated, the conversion complete flag is cleared, and the SAR register bits are cleared. Any pending interrupts are also cancelled. Note that the control, test, and status registers are initialized on reset; the initialized register state is defined in the register description section of this specification.

Enabling the module (using the ATDPU bit) does not cause the module to reset since the register file is not initialized. Finally, writing to control register ATD1C does not cause the module to reset; the current conversion will be terminated.

## 20.5 Interrupts

The ATD module originates interrupt requests and the MCU handles or services these requests. Details on how the ATD interrupt requests are handled can be found in [Chapter 12, “MCU Resets, Interrupts, and System Configuration”](#).

The ATD interrupt function is enabled by setting the ATDIE bit in the ATD1SC register. When the ATDIE bit is set, an interrupt is generated at the end of an ATD conversion and the ATD result registers (ATD1RH and ATD1RL) contain the result data generated by the conversion. If the interrupt function is disabled (ATDIE = 0), then the CCF flag must be polled to determine when a conversion is complete.

The interrupt will remain pending as long as the CCF flag is set. The CCF bit is cleared whenever the ATD status and control (ATD1SC) register is written. The CCF bit is also cleared whenever the ATD result registers (ATD1RH or ATD1RL) are read.

**Table 20-2. Interrupt Summary**

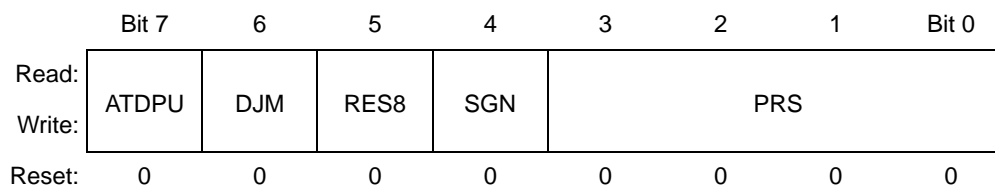
Interrupt	Local Enable	Description
CCF	ATDIE	Conversion complete

## 20.6 ATD Registers and Control Bits

The ATD has seven registers which control ATD functions.

Refer to the direct-page register summary in [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all ATD 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.

## 20.6.1 ATD Control (ATDC)



**Figure 20-4. ATD Control Register (ATD1C)**

Writes to the ATD control register will abort the current conversion, but will not start a new conversion.

### ATDPU — ATD Power Up

This bit provides program on/off control over the ATD, reducing power consumption when the ATD is not being used. When cleared, the ATDPU bit aborts any conversion in progress.

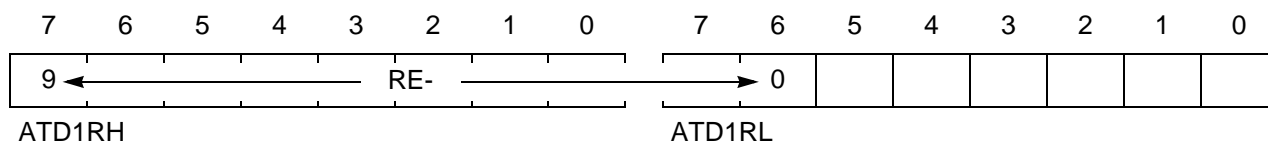
1 = ATD functionality.

0 = Disable the ATD and enter a low-power state.

### DJM — Data Justification Mode

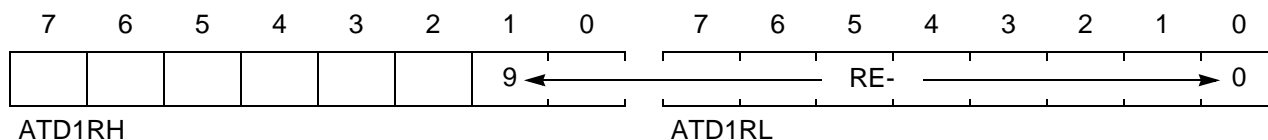
This bit determines how the 10-bit conversion result data maps onto the ATD result register bits. When RES8 is set, bit DJM has no effect and the 8-bit result is always located in ATD1RH.

For Left-justified Mode, result data bits 9–2 map onto bits 7–0 of ATD1RH, result data bits 1 and 0 map onto ATD1RL bits 7 and 6, where bit 7 of ATD1RH is the most significant bit (MSB).



**Figure 20-5. Left-Justified Mode**

For Right-justified Mode, result data bits 9 and 8 map onto bits 1 and 0 of ATD1RH, result data bits 7–0 map onto ATD1RL bits 7–0, where bit 1 of ATD1RH is the most significant bit (MSB).



**Figure 20-6. Right-Justified Mode**

The effect of the DJM bit on the result is shown in [Table 20-3](#).

1 = Result register data is right justified.

0 = Result register data is left justified.

### RES8 — ATD Resolution Select

This bit determines the resolution of the ATD converter, 8-bits or 10-bits. The ATD converter has the accuracy of a 10-bit converter. However, if 8-bit compatibility is required, selecting 8-bit resolution will map result data bits 9–2 onto ATD1RH bits 7–0.

The effect of the RES8 bit on the result is shown in [Table 20-3](#).

- 1 = 8-bit resolution selected.
- 0 = 10-bit resolution selected.

### SGN — Signed Result Select

This bit determines whether the result will be signed or unsigned data. Signed data is represented as 2's complement data and is achieved by complementing the MSB of the result. Signed Data Mode can be used only when the result is left justified (DJM = 0) and is not available for Right-justified Mode (DJM = 1). When a signed result is selected, the range for conversions becomes  $-512$  (\$200) to  $511$  (\$1FF) for 10-bit resolution and  $-128$  (\$80) to  $127$  (\$7F) for 8-bit resolution.

The effect of the SGN bit on the result is shown in [Table 20-3](#).

- 1 = Left justified result data is signed.
- 0 = Left justified result data is unsigned.

**Table 20-3. Available Result Data Formats**

RES8	DJM	SGN	Data Formats of Result	Analog Input $V_{REFH} = V_{DDA}$ , $V_{REFL} = V_{SSA}$ ATD1RH:ATD1RL	
				$V_{DDA}$	$V_{SSA}$
1	0	0	8-bit : left justified : unsigned	\$FF:\$00	\$00:\$00
1	0	1	8-bit : left justified : signed	\$7F:\$00	\$80:\$00
1	1	X <sup>1</sup>	8-bit : left justified <sup>2</sup> : unsigned	\$FF:\$00	\$00:\$00
0	0	0	10-bit : left justified : unsigned	\$FF:\$C0	\$00:\$00
0	0	1	10-bit : left justified : signed	\$7F:\$C0	\$80:\$00
0	1	X <sup>1</sup>	10-bit : right justified : unsigned	\$03:\$FF	\$00:\$00

<sup>1</sup> The SGN bit is only effective when DJM = 0. When DJM = 1, SGN is ignored.

<sup>2</sup> 8-bit results are always in ATD1RH.

### PRS — Prescaler Rate Select

This field of bits determines the prescaled factor for the ATD conversion clock. [Table 20-4](#) illustrates the divide-by operation and the appropriate range of bus clock frequencies.

**Table 20-4. Clock Prescaler Values**

PRS	Factor = (PRS + 1) × 2	Max Bus Clock MHz (2 MHz max ATD Clock) <sup>1</sup>	Max Bus Clock MHz (1 MHz max ATD Clock) <sup>2</sup>	Min Bus Clock <sup>3</sup> MHz (500 kHz min ATD Clock)
0000	2	4	2	1
0001	4	8	4	2
0010	6	12	6	3
0011	8	16	8	4
0100	10	20	10	5
0101	12	20	12	6

**Table 20-4. Clock Prescaler Values (continued)**

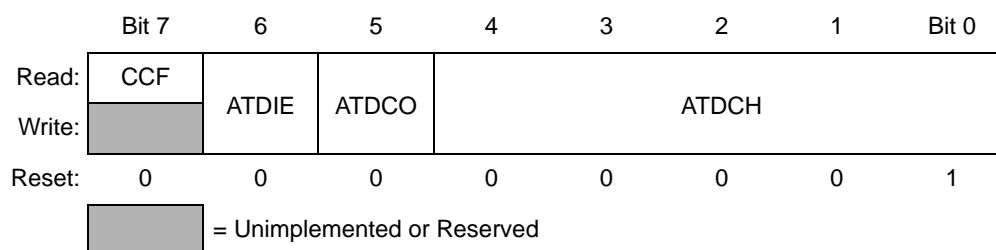
PRS	Factor = (PRS + 1) × 2	Max Bus Clock MHz (2 MHz max ATD Clock) <sup>1</sup>	Max Bus Clock MHz (1 MHz max ATD Clock) <sup>2</sup>	Min Bus Clock <sup>3</sup> MHz (500 kHz min ATD Clock)
0110	14	20	14	7
0111	16	20	16	8
1000	18	20	18	9
1001	20	20	20	10
1010	22	20	20	11
1011	24	20	20	12
1100	26	20	20	13
1101	28	20	20	14
1110	30	20	20	15
1111	32	20	20	16

<sup>1</sup> Maximum ATD conversion clock frequency is 2 MHz. The max bus clock frequency is computed from the max ATD conversion clock frequency times the indicated prescaler setting; i.e., for a PRS of 0, max bus clock = 2 (max ATD conversion clock frequency) × 2 (Factor) = 4 MHz.

<sup>2</sup> Use these settings if the maximum desired ATD conversion clock frequency is 1 MHz. The max bus clock frequency is computed from the max ATD conversion clock frequency times the indicated prescaler setting; i.e., for a PRS of 0, max bus clock = 1 (max ATD conversion clock frequency) × 2 (Factor) = 2 MHz.

<sup>3</sup> Minimum ATD conversion clock frequency is 500 kHz. The min bus clock frequency is computed from the min ATD conversion clock frequency times the indicated prescaler setting; i.e., for a PRS of 1, min bus clock = 0.5 (min ATD conversion clock frequency) × 2 (Factor) = 1 MHz.

## 20.6.2 ATD Status and Control (ATD1SC)


**Figure 20-7. ATD Status and Control Register (ATD1SC)**

Writes to the ATD status and control register clears the CCF flag, cancels any pending interrupts, and initiates a new conversion.

### CCF — Conversion Complete Flag

The CCF is a read-only bit which is set each time a conversion is complete. The CCF bit is cleared whenever the ATD1SC register is written. It is also cleared whenever the result registers, ATD1RH or ATD1RL, are read.

1 = Current conversion is complete.

0 = Current conversion is not complete.

### ATDIE — ATD Interrupt Enabled

When this bit is set, an interrupt is generated upon completion of an ATD conversion. At this time, the result registers contain the result data generated by the conversion. The interrupt will remain pending as long as the conversion complete flag CCF is set. If the ATDIE bit is cleared, then the CCF bit must be polled to determine when the conversion is complete. Note that system reset clears pending interrupts.

- 1 = ATD interrupt enabled.
- 0 = ATD interrupt disabled.

### ATDCO — ATD Continuous Conversion

When this bit is set, the ATD will convert samples continuously and update the result registers at the end of each conversion. When this bit is cleared, only one conversion is completed between writes to the ATD1SC register.

- 1 = Continuous Conversion Mode.
- 0 = Single Conversion Mode.

### ATDCH — Analog Input Channel Select

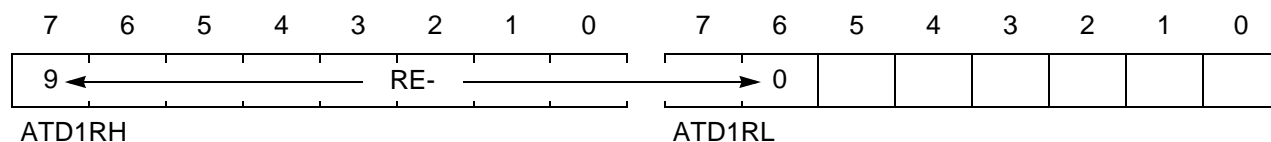
This field of bits selects the analog input channel whose signal is sampled and converted to digital codes. [Table 20-5](#) lists the coding used to select the various analog input channels.

**Table 20-5. Analog Input Channel Select Coding**

ATDCH	Analog Input Channel
00	AD0
01	AD1
02	AD2
03	AD3
04	AD4
05	AD5
06	AD6
07	AD7
08–1D	Reserved (default to $V_{REFL}$ )
1E	$V_{REFH}$
1F	$V_{REFL}$

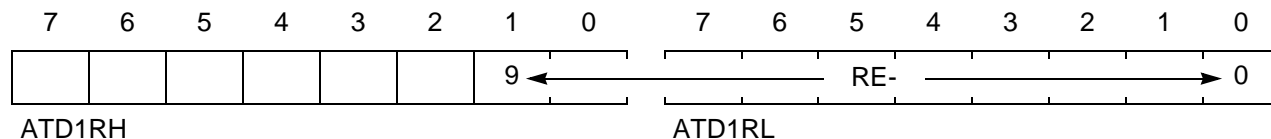
## 20.6.3 ATD Result Data (ATD1RH, ATD1RL)

For Left-justified Mode, result data bits 9–2 map onto bits 7–0 of ATD1RH, result data bits 1 and 0 map onto ATD1RL bits 7 and 6, where bit 7 of ATD1RH is the most significant bit (MSB).



**Figure 20-8. Left-Justified Mode**

For Right-justified Mode, result data bits 9 and 8 map onto bits 1 and 0 of ATD1RH, result data bits 7–0 map onto ATD1RL bits 7–0, where bit 1 of ATD1RH is the most significant bit (MSB).



**Figure 20-9. Right-Justified Mode**

The ATD 10-bit conversion results are stored in two 8-bit result registers, ATD1RH and ATD1RL. The result data is formatted either left or right justified where the format is selected using the DJM control bit in the ATD1C register. The 10-bit result data is mapped either between ATD1RH bits 7–0 and ATD1RL bits 7–6 (left justified), or ATD1RH bits 1–0 and ATD1RL bits 7–0 (right justified).

For 8-bit conversions, the 8-bit result is always located in ATD1RH bits 7–0, and the ATD1RL bits read 0. For 10-bit conversions, the six unused bits always read 0.

The ATD1RH and ATD1RL registers are read-only.

### 20.6.4 ATD Pin Enable (ATD1PE)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:	ATDPE7	ATDPE6	ATDPE5	ATDPE4	ATDPE3	ATDPE2	ATDPE1	ATDPE0
Reset:	0	0	0	0	0	0	0	0

**Figure 20-10. ATD Pin Enable Register (ATD1PE)**

The ATD pin enable register allows the pins dedicated to the ATD module to be configured for ATD usage. A write to this register will abort the current conversion but will not initiate a new conversion. If the ATDPE<sub>x</sub> bit is 0 (disabled for ATD usage) but the corresponding analog input channel is selected via the ATDCH bits, the ATD will not convert the analog input but will instead convert  $V_{REFL}$  placing zeroes in the ATD result registers.

#### ATDPE7

:ATDPE0 — ATD Pin 7–0 Enables

- 1 = Pin enabled for ATD usage.
- 0 = Pin disabled for ATD usage.



# Chapter 21

## Development Support

### 21.1 Features

Features of the Background Debug Controller (BDC) include:

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

Features of the debug module (DBG) include:

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

## 21.2 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.

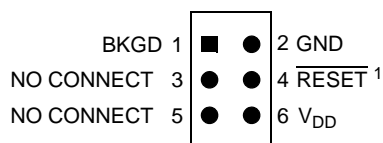
Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer 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  $V_{DD}$ .

### NOTE

On MCUs where the  $\overline{\text{RESET}}$  pin is not input-only, 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 start up of a target system before the on-chip non volatile memory has been programmed.

On MCUs where  $\overline{\text{RESET}}$  pin is input-only, the  $\overline{\text{RESET}}$  pin can only provide a reset into user mode.

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.



<sup>1</sup> On MCUs where  $\overline{\text{RESET}}$  is an input only, the  $\overline{\text{RESET}}$  pin can only provide a reset into user mode. To enter active BDM mode use the background debug force

**Figure 21-1. BDM Tool Connector**

## 21.2.1 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 21.2.2, "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 speed-up pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 21.2.2, "Communication Details"](#), for more detail.

## 21.2.2 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.

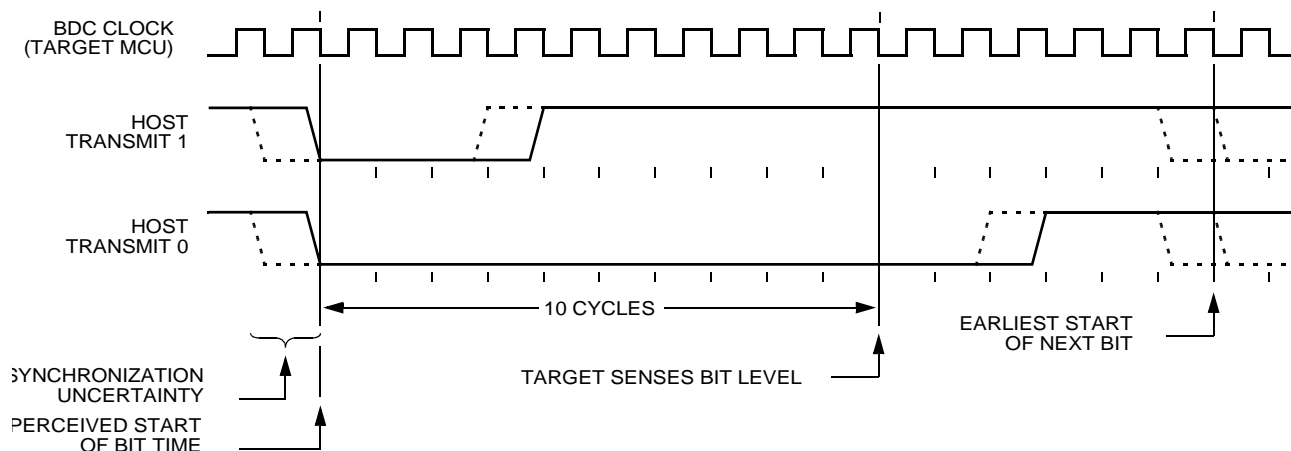
The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

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.

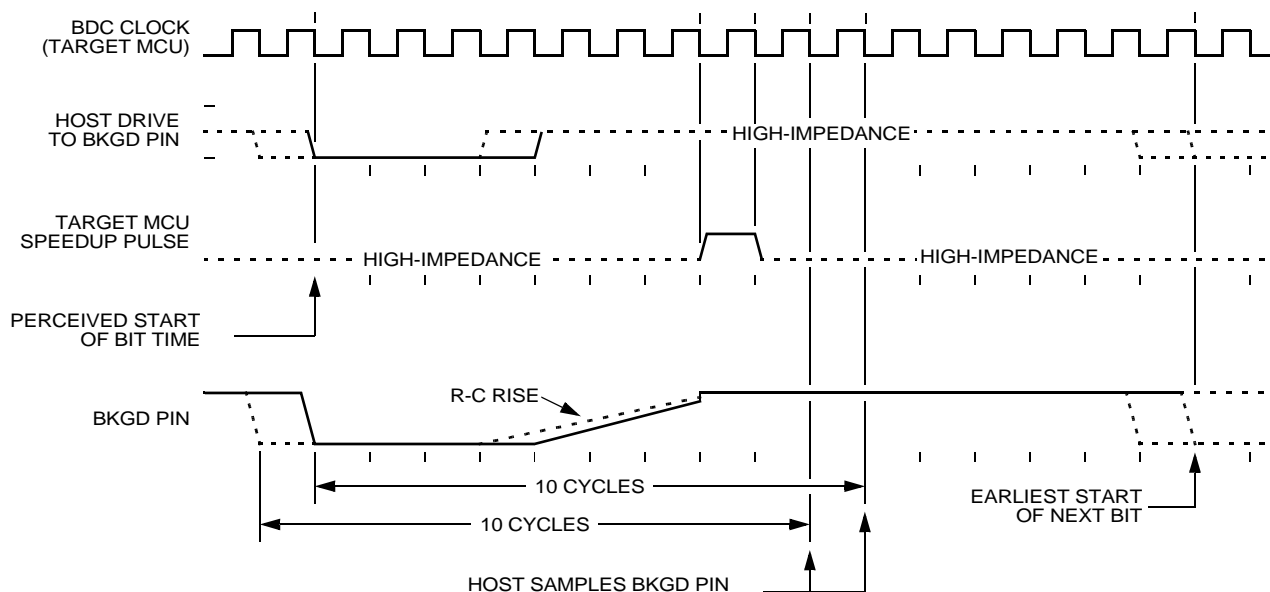
[Figure 21-2](#) shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge

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 21-2. BDC Host-to-Target Serial Bit Timing**

Figure 21-3 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 speed-up 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 21-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 21-4 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.

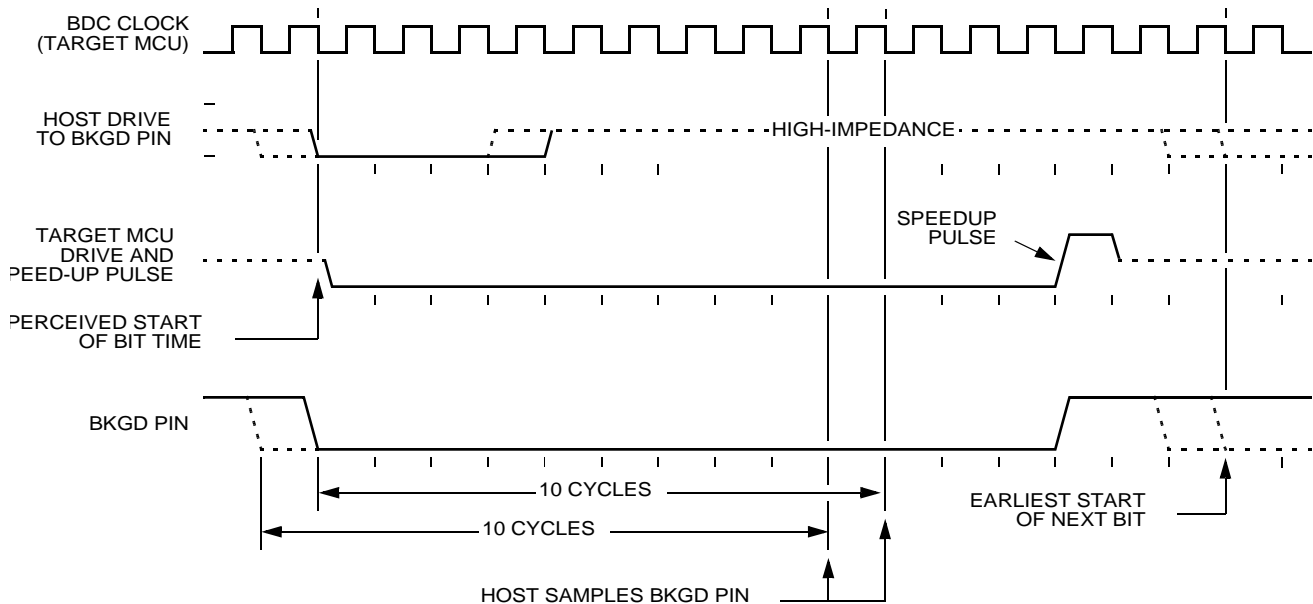


Figure 21-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

### 21.2.3 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 21-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

### 21.2.4 Coding Structure Nomenclature

This nomenclature is used in Table 21-1 to describe the coding structure of the BDC commands.

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

## Development Support

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 21-1. 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
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

**Table 21-1. BDC Command Summary (continued)**

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
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.

## 21.2.5 BDC Hardware Breakpoint

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

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

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

## 21.3 On-Chip Debug System (DBG)

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

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

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is covered in greater detail in [Section 21.3.6, "Hardware Breakpoints"](#).

### 21.3.1 Comparators A and B

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

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an



additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If  $RWAEN = 1$  (enabled) and  $RWA = 0$  (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

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

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

### 21.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, users would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full ( $CNT = 1:0:0:0$ ), the information is shifted by one position and the host must perform  $((8 - CNT) - 1)$  dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see [Section 21.3.5, "Trigger Modes"](#)), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

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

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

### 21.3.3 Change-of-Flow Information

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

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

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

### 21.3.4 Tag vs. Force Breakpoints and Triggers

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

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

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

### 21.3.5 Trigger Modes

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

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

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

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following Trigger Mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

- **A-Only** — Trigger when the address matches the value in comparator A
- **A OR B** — Trigger when the address matches either the value in comparator A or the value in comparator B
- **A Then B** — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.
- **A AND B Data (Full Mode)** — This is called a Full Mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.
- In Full Trigger Mode it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if users do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.
- **A AND NOT B Data (Full Mode)** — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.
- In Full Trigger Mode it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if users do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.
- **Event-Only B (Store Data)** — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.
- **A Then Event-Only B (Store Data)** — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

- **Inside Range ( $A \leq \text{Address} \leq B$ )** — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.
- **Outside Range ( $\text{Address} < A$  or  $\text{Address} > B$ )** — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

### 21.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGCR register may be set to 1 to allow any of the trigger conditions described in [Section 21.3.5, “Trigger Modes”](#), to be used to generate a hardware breakpoint request to the CPU. TAG in DBGCR controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to Active Background Mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to Active Background Mode.

If the Background Mode has not been enabled ( $\text{ENBDM} = 1$ ) by a serial WRITE\_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to Active Background Mode.

## 21.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits. Refer to the high-page register summary in [Chapter 11, “MCU Memory”](#) for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 21.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in Active Background Mode. (This prevents the ambiguous condition of the control bit forbidding Active Background Mode while the MCU is already in Active Background Mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE\_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

### 21.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ\_STATUS and WRITE\_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

Offset

	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

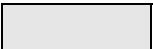
 = Unimplemented or Reserved

Figure 21-5. BDC Status and Control Register (BDCSCR)

Table 21-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	<b>Enable BDM (Permit Active Background Mode)</b> — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow Active Background Mode commands
6 BDMACT	<b>Background Mode Active Status</b> — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	<b>BDC Breakpoint Enable</b> — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	<b>Force/Tag Select</b> — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters Active Background Mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter Active Background Mode if CPU attempts to execute that instruction 1 Breakpoint match forces Active Background Mode at next instruction boundary (address need not be an opcode)
3 CLKSW	<b>Select Source for BDC Communications Clock</b> — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

**Table 21-2. BDCSCR Register Field Descriptions (continued)**

Field	Description
2 WS	<p><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.</p> <p>0 Target CPU is running user application code or in Active Background Mode (was not in wait or Stop Mode when background became active)</p> <p>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</p>
1 WSF	<p><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.)</p> <p>0 Memory access did not conflict with a wait or stop instruction</p> <p>1 Memory access command failed because the CPU entered wait or Stop Mode</p>
0 DVF	<p><b>Data Valid Failure Status</b> — This status bit is not used in the HCS08 because it does not have any slow access memory.</p> <p>0 Memory access did not conflict with a slow memory access</p> <p>1 Memory access command failed because CPU was not finished with a slow memory access</p>

### 21.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

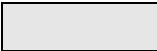
This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ\_BKPT and WRITE\_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in Active Background Mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to [Section 21.2.5, “BDC Hardware Breakpoint”](#).

## 21.4.2 System Background Debug Force Reset Register (SBD FR)

This register contains a single write-only control bit. A serial Active 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.

Offset

	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 Active Background Mode debug commands, not from user programs.

Figure 21-6. System Background Debug Force Reset Register (SBD FR)

## 21.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

### 21.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 21.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 21.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 21.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 21.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

### 21.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It is not necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.



### 21.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

Offset

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

Figure 21-7. Debug Control Register (DBGC)

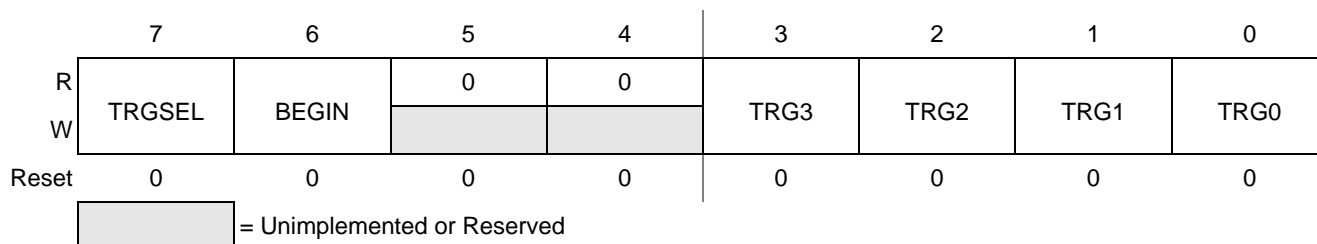
Table 21-3. DBGC Register Field Descriptions

Field	Description
7 DBGEN	<b>Debug Module Enable</b> — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure. 0 DBG disabled 1 DBG enabled
6 ARM	<b>Arm Control</b> — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN. 0 Debugger not armed 1 Debugger armed
5 TAG	<b>Tag/Force Select</b> — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect. 0 CPU breaks requested as force type requests 1 CPU breaks requested as tag type requests
4 BRKEN	<b>Break Enable</b> — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests. 0 CPU break requests not enabled 1 Triggers cause a break request to the CPU
3 RWA	<b>R/W Comparison Value for Comparator A</b> — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A. 0 Comparator A can only match on a write cycle 1 Comparator A can only match on a read cycle
2 RWAEN	<b>Enable R/W for Comparator A</b> — Controls whether the level of R/W is considered for a comparator A match. 0 R/W is not used in comparison A 1 R/W is used in comparison A
1 RWB	<b>R/W Comparison Value for Comparator B</b> — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B. 0 Comparator B can match only on a write cycle 1 Comparator B can match only on a read cycle
0 RWBEN	<b>Enable R/W for Comparator B</b> — Controls whether the level of R/W is considered for a comparator B match. 0 R/W is not used in comparison B 1 R/W is used in comparison B

### 21.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

Offset



**Figure 21-8. Debug Trigger Register (DBGT)**

**Table 21-4. DBGT Register Field Descriptions**


Field	Description
7 TRGSEL	<p><b>Trigger Type</b> — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed.</p> <p>0 Trigger on access to compare address (force)                      1 Trigger if opcode at compare address is executed (tag)</p>
6 BEGIN	<p><b>Begin/End Trigger Select</b> — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces.</p> <p>0 Data stored in FIFO until trigger (end trace)                      1 Trigger initiates data storage (begin trace)</p>
3:0 TRG[3:0]	<p><b>Select Trigger Mode</b> — Selects one of nine triggering modes, as described below.</p> <p>0000 A-only                      0001 A OR B                      0010 A Then B                      0011 Event-only B (store data)                      0100 A then event-only B (store data)                      0101 A AND B data (full mode)                      0110 A AND NOT B data (full mode)                      0111 Inside range: <math>A \leq \text{address} \leq B</math>                      1000 Outside range: <math>\text{address} &lt; A</math> or <math>\text{address} &gt; B</math>                      1001 – 1111 (No trigger)</p>

### 21.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

Offset

	7	6	5	4	3	2	1	0
R	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
W								
Reset	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 21-9. Debug Status Register (DBGS)**

**Table 21-5. DBGS Register Field Descriptions**

Field	Description
7 AF	<b>Trigger Match A Flag</b> — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	<b>Trigger Match B Flag</b> — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	<b>Arm Flag</b> — While DBGEN = 1, this status bit is a read-only image of ARM in DBGIC. This bit is set by writing 1 to the ARM control bit in DBGIC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGIC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	<b>FIFO Valid Count</b> — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

