

MC12311

Sub 1 GHz Low Power Transceiver plus Microcontroller Reference Manual

Document Number: MC12311RM
Rev. 1.0
11/2011

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
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

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

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

Asia/Pacific:
Freescale Semiconductor 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

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

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. 2006, 2007, 2008, 2009, 2010, 2011. All rights reserved.

Contents

About This Book

Audience	xv
Organization	xv
Revision History	xvi
References	xviii

Chapter 1 MC12311 Introduction

1.1	Ordering Information	1-1
1.2	General Platform Features	1-1
1.3	9S08QE32 Microcontroller Features	1-2
1.4	RF Transceiver Features	1-3
1.5	Software Solutions	1-3
1.6	System Overview	1-4
1.6.1	Transceiver Overview	1-4
1.6.2	MCU Block Diagram	1-5
1.6.3	QE32 Central Processing Unit (CPU)	1-6
1.6.4	MCU Peripheral Modules	1-7
1.6.5	MCU Internal Clock Distribution	1-8

Chapter 2 MC12311 Pins and Connections

2.1	Device Pin Assignment	2-1
2.2	Pin Definitions	2-2
2.3	Internal Functional Interconnects	2-5

Chapter 3 System Considerations

3.1	Introduction	3-1
3.2	Power Connections	3-1
3.3	System Functional Interconnects	3-3
3.3.1	On board Connections (SPI Channel and Status)	3-3
3.3.2	System Reset	3-3
3.3.3	External Clock Connections	3-6
3.3.4	Additional Transceiver Status Signals	3-6
3.4	System Clock Sources and Configurations	3-6
3.4.1	Transceiver Oscillator	3-7
3.4.2	MCU Clock Sources	3-9
3.4.3	System Clock Configurations	3-10
3.4.4	MCU Background/Mode Select (BKGD/MS)	3-12
3.5	MC12311 GPIO (Mixed I/O from Transceiver and MCU)	3-12
3.5.1	MCU GPIO Characteristics	3-13

3.5.2	Transceiver DIOX Characteristics	3-14
3.6	MC12311 Digital Signal Summary	3-14
3.7	Transceiver RF Configurations and External Connections	3-16
3.7.1	RF Interface Pins	3-16
3.7.2	Standard Output Power RF Configuration (Single, Bidirectional Port)	3-16
3.7.3	Higher Output Power RF Configuration (Dual Port with Optional External Power Amplifier) 3-17	
3.7.4	Filter and Matching Network Component Values	3-18

Chapter 4

Sub 1-Ghz Transceiver Architecture Description

4.1	Overview	4-1
4.2	Simplified Block Diagram	4-1
4.3	Transceiver Power Supply	4-2
4.4	Low Battery Detector	4-2
4.5	Frequency Synthesis	4-2
4.5.1	Reference Oscillator	4-3
4.5.2	CLKOUT Output	4-3
4.5.3	PLL Architecture	4-3
4.5.4	Lock Time	4-4
4.5.5	Lock Detect Indicator	4-5
4.6	Transmitter Description	4-5
4.6.1	Bit Rate Setting	4-5
4.6.2	FSK Modulation	4-6
4.6.3	OOK Modulation	4-7
4.6.4	Modulation Shaping	4-7
4.6.5	Power Amplifiers	4-7
4.6.6	Over Current Protection	4-8
4.7	Receiver Description	4-8
4.7.1	LNA - Single to Differential Buffer	4-9
4.7.2	Automatic Gain Control	4-9
4.7.3	Continuous-Time DAGC	4-11
4.7.4	Quadrature Mixer - ADCs - Decimators	4-12
4.7.5	Channel Filter	4-12
4.7.6	DC Cancellation	4-13
4.7.7	Complex Filter - OOK	4-14
4.7.8	RSSI	4-14
4.7.9	Cordic	4-14
4.7.10	FSK Demodulator	4-15
4.7.11	OOK Demodulator	4-15
4.7.12	Bit Synchronizer	4-17
4.7.13	Frequency Error Indicator (FEI)	4-19
4.7.14	Automatic Frequency Correction (AFC)	4-20
4.7.15	Optimized Setup for Low Modulation Index Systems	4-20

4.7.16	Temperature Sensor	4-21
4.7.17	Timeout Function	4-22

Chapter 5 Transceiver Operating Modes

5.1	Basic Modes	5-1
5.2	Automatic Sequencer and Wake-Up Times	5-1
5.2.1	Transmitter Startup Time	5-2
5.2.2	Tx Start Procedure	5-2
5.2.3	Receiver Startup Time	5-3
5.2.4	Rx Start Procedure	5-4
5.2.5	Optimized Frequency Hopping Sequences	5-4
5.3	Listen Mode	5-5
5.3.1	Timing	5-5
5.3.2	Criteria	5-6
5.3.3	End of Cycle Actions	5-6
5.3.4	RC Timer Accuracy	5-7
5.4	AutoModes	5-7

Chapter 6 Transceiver Digital Control and Communications

6.1	Overview	6-1
6.1.1	Data Operation Modes	6-1
6.2	Control Block Description	6-2
6.2.1	SPI Interface	6-2
6.2.2	FIFO	6-3
6.2.3	Sync Word Recognition	6-5
6.2.4	Packet Handler	6-6
6.2.5	Control	6-6
6.3	Digital IO Pins Mapping	6-6
6.3.1	DIO Pins Mapping in Continuous Mode	6-7
6.3.2	DIO Pins Mapping in Packet Mode	6-7
6.4	Continuous Mode	6-8
6.4.1	General Description	6-8
6.4.2	Tx Processing	6-8
6.4.3	Rx Processing	6-9
6.5	Packet Mode	6-9
6.5.1	General Description	6-9
6.5.2	Packet Format	6-10
6.5.3	Tx Processing (without AES)	6-12
6.5.4	Rx Processing (without AES)	6-13
6.5.5	AES	6-14
6.5.6	Handling Large Packets	6-15
6.5.7	Packet Filtering	6-15

6.5.8	DC-Free Data Mechanisms	6-17
6.6	Register Summary	6-18
6.7	Common Configuration Registers	6-21
6.8	Transmitter Registers	6-25
6.9	Receiver Registers	6-26
6.10	IRQ and Pin Mapping Registers	6-29
6.11	Packet Engine Registers	6-32
6.12	Temperature Sensor Registers	6-36
6.13	Test Registers	6-36

Chapter 7 MC12311 Transceiver - MCU SPI Interface

7.1	SiP Level SPI Pin Connections	7-1
7.2	Features	7-2
7.3	SPI System Block Diagram	7-2
7.3.1	SPI Signal Definitions	7-3
7.3.2	MC12311 SPI Transaction Protocol	7-3
7.3.3	MC12311 SPI Transaction Timing	7-4
7.4	QE32 MCU SPI Module Overview	7-6
7.5	MCU SPI Block Diagrams	7-6
7.5.1	MCU SPI Module Block Diagram	7-6
7.5.2	MCU SPI Baud Rate Generation	7-7
7.6	MCU SPI Functional Description	7-7
7.6.1	SPI Clock Formats	7-8
7.6.2	SPI Clock Gating	7-8
7.6.3	MCU SPI Pin Controls	7-9
7.6.4	MCU SPI Interrupts	7-10
7.6.5	Mode Fault Detection	7-10
7.7	MCU SPI Registers and Control Bits	7-10
7.7.1	SPI Control Register 1 (SPI1C1)	7-11
7.7.2	SPI Control Register 2 (SPI1C2)	7-12
7.7.3	SPI Baud Rate Register (SPI1BR)	7-13
7.7.4	SPI Status Register (SPI1S)	7-14
7.7.5	SPI Data Register (SPI1D)	7-15
7.8	Configuring MCU Registers for Proper SPI Operation	7-15
7.8.1	Set SPI Module Mode	7-15
7.8.2	SPI Baud Rate Control	7-16

Chapter 8 MCU Modes of Operation

8.1	Introduction	8-1
8.2	Features	8-1
8.3	Run Mode	8-1
8.3.1	Low Power Run Mode (LPRun)	8-1

8.4	Active Background Mode	8-2
8.5	Wait Mode	8-3
8.5.1	Low-Power Wait Mode (LPWait)	8-4
8.6	Stop Modes	8-4
8.6.1	Stop2 Mode	8-5
8.6.2	Stop3 Mode	8-6
8.6.3	Active BDM Enabled in Stop Mode	8-7
8.6.4	LVD Enabled in Stop Mode	8-8
8.6.5	Stop modes in Low Power Run Mode	8-8
8.7	Mode selection	8-8
8.7.1	On-Chip Peripheral Modules in Stop and Low Power Modes	8-12

Chapter 9 Memory

9.1	9S08QE32 Memory Map	9-1
9.2	Reset and Interrupt Vector Assignments	9-2
9.3	Register Addresses and Bit Assignments	9-3
9.4	RAM	9-10
9.5	Flash	9-11
9.5.1	Features	9-11
9.5.2	Program and Erase Times	9-11
9.5.3	Program and Erase Command Execution	9-12
9.5.4	Burst Program Execution	9-13
9.5.5	Access Errors	9-15
9.5.6	Flash Block Protection	9-15
9.5.7	Vector Redirection	9-16
9.6	Security	9-16
9.7	Flash Registers and Control Bits	9-17
9.7.1	Flash Clock Divider Register (FCDIV)	9-18
9.7.2	Flash Options Register (FOPT and NVOPT)	9-19
9.7.3	Flash Configuration Register (FCNFG)	9-20
9.7.4	Flash Protection Register (FPROT and NVPROT)	9-20
9.7.5	Flash Status Register (FSTAT)	9-22
9.7.6	Flash Command Register (FCMD)	9-23

Chapter 10 Resets, Interrupts, and General System Control

10.1	Introduction	10-1
10.2	Features	10-1
10.3	MCU Reset	10-1
10.4	Computer Operating Properly (COP) Watchdog	10-2
10.5	Interrupts	10-3
10.5.1	Interrupt Stack Frame	10-4
10.5.2	External Interrupt Request (IRQ) Pin	10-4

10.5.3	Interrupt Vectors, Sources, and Local Masks	10-5
10.6	Low-Voltage Detect (LVD) System	10-7
10.6.1	Power-On Reset Operation	10-7
10.6.2	Low-Voltage Detection (LVD) Reset Operation	10-7
10.6.3	Low-Voltage Detection (LVD) Interrupt Operation	10-7
10.6.4	Low-Voltage Warning (LVW) Interrupt Operation	10-7
10.7	Peripheral Clock Gating	10-7
10.8	Reset, Interrupt, and System Control Registers and Control Bits	10-8
10.8.1	Interrupt Pin Request Status and Control Register (IRQSC)	10-8
10.8.2	System Reset Status Register (SRS)	10-10
10.8.3	System Background Debug Force Reset Register (SBDFR)	10-11
10.8.4	System Options Register 1 (SOPT1)	10-12
10.8.5	System Options Register 2 (SOPT2)	10-14
10.8.6	System Device Identification Register (SDIDH, SDIDL)	10-15
10.8.7	System Power Management Status and Control 1 Register (SPMSC1)	10-16
10.8.8	System Power Management Status and Control 2 Register (SPMSC2)	10-17
10.8.9	System Power Management Status and Control 3 Register (SPMSC3)	10-18
10.8.10	System Clock Gating Control 1 Register (SCGC1)	10-20
10.8.11	System Clock Gating Control 2 Register (SCGC2)	10-21

Chapter 11 MCU Parallel Input/Output

11.1	Port Data and Data Direction	11-1
11.2	Pullup, Slew Rate, and Drive Strength	11-2
11.2.1	Port Internal Pullup Enable	11-2
11.2.2	Port Slew Rate Enable	11-2
11.2.3	Port Drive Strength Select	11-2
11.3	Pin Behavior in Stop Modes	11-3
11.4	Parallel I/O and Pin Control Registers	11-3
11.4.1	Port A Registers	11-4
11.4.2	Port A Drive Strength Selection Register (PTADS)	11-6
11.4.3	Port B Registers	11-7
11.4.4	Port C Registers	11-9
11.4.5	Port D Registers	11-12
11.4.6	Port E Registers	11-14

Chapter 12 MCU Internal Clock Source (ICS)

12.1	Introduction	12-1
12.1.1	External Oscillator	12-1
12.1.2	Stop2 Mode Considerations	12-1
12.1.3	Features	12-1
12.1.4	Block Diagram	12-2
12.1.5	Modes of Operation	12-3

12.2	External Signal Description	12-4
12.3	Register Definition	12-4
12.3.1	ICS Control Register 1 (ICSC1)	12-4
12.3.2	ICS Control Register 2 (ICSC2)	12-6
12.3.3	ICS Trim Register (ICSTRM)	12-7
12.3.4	ICS Status and Control (ICSSC)	12-7
12.3.5	ICS Test Register (ICST)	12-9
12.4	Functional Description	12-10
12.4.1	Operational Modes	12-10
12.4.2	Mode Switching	12-12
12.4.3	Bus Frequency Divider	12-13
12.4.4	Low Power Bit Usage	12-13
12.4.5	DCO Maximum Frequency with 32.768 kHz Oscillator	12-13
12.4.6	Internal Reference Clock	12-13
12.4.7	External Reference Clock	12-14
12.4.8	Fixed Frequency Clock	12-14
12.4.9	BDC Clock Operation	12-14
12.4.10	Scan Coverage	12-14
12.5	S08ICSV2 Electricals	12-15
12.5.1	External Oscillator (XOSC) and Internal Clock Source (ICS) Characteristics	12-15
12.5.2	Local Clock	12-17

Chapter 13 Central Processor UnitS08CPUV4

13.1	Introduction	13-1
13.1.1	Features	13-1
13.2	Programmer's Model and CPU Registers	13-2
13.2.1	Accumulator (A)	13-2
13.2.2	Index Register (H:X)	13-2
13.2.3	Stack Pointer (SP)	13-3
13.2.4	Program Counter (PC)	13-3
13.2.5	Condition Code Register (CCR)	13-3
13.3	Addressing Modes	13-5
13.3.1	Inherent Addressing Mode (INH)	13-5
13.3.2	Relative Addressing Mode (REL)	13-5
13.3.3	Immediate Addressing Mode (IMM)	13-5
13.3.4	Direct Addressing Mode (DIR)	13-6
13.3.5	Extended Addressing Mode (EXT)	13-6
13.3.6	Indexed Addressing Mode	13-6
13.4	Special Operations	13-7
13.4.1	Reset Sequence	13-7
13.4.2	Interrupt Sequence	13-7
13.4.3	Wait Mode Operation	13-8
13.4.4	Stop Mode Operation	13-8

13.4.5	BGND Instruction	13-9
13.5	QE32 Instruction Set Summary	13-10

Chapter 14 MCU Keyboard Interrupt (KBI)

14.1	Introduction	14-1
14.1.1	KBI Clock Gating	14-1
14.1.2	Features	14-2
14.1.3	Modes of Operation	14-2
14.1.4	Block Diagram	14-2
14.2	External Signal Description	14-3
14.3	Register Definition	14-4
14.3.1	Register Descriptions	14-4
14.3.2	KBI Interrupt Status and Control Register (KBISC)	14-4
14.3.3	KBI Interrupt Pin Select Register (KBIPE)	14-5
14.3.4	KBI Interrupt Edge Select Register (KBIES)	14-5
14.4	Functional Description	14-5
14.4.1	Edge Only Sensitivity	14-6
14.4.2	Edge and Level Sensitivity	14-6
14.4.3	Pullup/Pulldown Resistors	14-6
14.4.4	Keyboard Interrupt Initialization	14-6

Chapter 15 Timer/Pulse-Width Modulator

15.0.1	Features	15-1
15.0.2	ACMP/TPM Configuration Information	15-1
15.0.3	TPM Clock Gating	15-1
15.0.4	Modes of Operation	15-1
15.0.5	Block Diagram	15-2
15.1	Signal Description	15-4
15.1.1	Detailed Signal Descriptions	15-4
15.2	Memory Map and Register Definition	15-8
15.2.1	Module Memory Map	15-8
15.2.2	TPM Status and Control Register (TPM1SC)	15-8
15.2.3	TPM-Counter Registers (TPM1CNTH:TPM1CNTL)	15-9
15.2.4	TPM Counter Modulo Registers (TPM1MODH:TPM1MODL)	15-10
15.2.5	TPM Channel n Status and Control Register (TPM1CnSC)	15-11
15.2.6	TPM Channel Value Registers (TPM1CnVH:TPM1CnVL)	15-13
15.3	Functional Description	15-14
15.3.1	Counter	15-15
15.3.2	Channel Mode Selection	15-17
15.4	Reset Overview	15-20
15.4.1	General	15-20
15.4.2	Description of Reset Operation	15-20

15.5	Interrupts	15-20
15.5.1	General	15-20
15.5.2	Description of Interrupt Operation	15-21

Chapter 16 MCU Serial Communications Interface (SCI)

16.1	Introduction	16-1
16.1.1	Features	16-1
16.1.2	Modes of Operation	16-1
16.1.3	Block Diagram	16-2
16.2	Register Definition	16-4
16.2.1	SCI Baud Rate Registers (SCIxBDH, SCIxBDL)	16-4
16.2.2	SCI Control Register 1 (SCIxC1)	16-5
16.2.3	SCI Control Register 2 (SCIxC2)	16-6
16.2.4	SCI Status Register 1 (SCIxS1)	16-7
16.2.5	SCI Status Register 2 (SCIxS2)	16-9
16.2.6	SCI Control Register 3 (SCIxC3)	16-10
16.2.7	SCI Data Register (SCIxD)	16-11
16.3	Functional Description	16-11
16.3.1	Baud Rate Generation	16-11
16.3.2	Transmitter Functional Description	16-12
16.3.3	Receiver Functional Description	16-13
16.3.4	Interrupts and Status Flags	16-15
16.3.5	Additional SCI Functions	16-16
16.3.6	SCI Clock Gating	16-17

Chapter 17 Inter-Integrated Circuit (IIC)

17.1	Introduction	17-1
17.1.1	Module Configuration	17-1
17.1.2	IIC Clock Gating	17-1
17.1.3	Features	17-1
17.1.4	Modes of Operation	17-2
17.1.5	Block Diagram	17-3
17.2	Signal Description	17-3
17.3	External Signal Description	17-3
17.3.1	SCL — Serial Clock Line	17-4
17.3.2	SDA — Serial Data Line	17-4
17.4	Register Definition	17-4
17.4.1	Module Memory Map	17-4
17.4.2	Register Descriptions	17-4
17.4.3	IIC Address Register (IIC1A)	17-4
17.4.4	IIC Frequency Divider Register (IIC1F)	17-5
17.4.5	IIC Control Register (IIC1C1)	17-9

17.4.6	IIC Status Register (IIC1S).....	17-10
17.4.7	IIC Data I/O Register (IIC1D)	17-11
17.4.8	IIC Control Register 2 (IIC1C2).....	17-11
17.5	Functional Description	17-12
17.5.1	IIC Protocol.....	17-12
17.5.2	10-bit Address.....	17-16
17.5.3	General Call Address	17-17
17.6	Resets	17-17
17.7	Interrupts.....	17-17
17.7.1	Byte Transfer Interrupt	17-17
17.7.2	Address Detect Interrupt.....	17-17
17.7.3	Arbitration Lost Interrupt	17-17
17.8	Initialization/Application Information	17-19

Chapter 18 Analog to Digital (ATD) Module

18.1	Introduction.....	18-1
18.1.1	ADC Clock Gating	18-1
18.1.2	Module Configurations.....	18-1
18.1.3	Features.....	18-3
18.1.4	ADC Module Block Diagram.....	18-3
18.2	External Signal Description	18-4
18.2.1	Analog Power (V_{DDA})	18-5
18.2.2	Analog Ground (V_{SSA}).....	18-5
18.2.3	Voltage Reference High (V_{REFH}).....	18-5
18.2.4	Voltage Reference Low (V_{REFL})	18-5
18.2.5	Analog Channel Inputs (ADx)	18-5
18.3	Register Definition	18-5
18.3.1	Status and Control Register 1 (ADCSC1)	18-5
18.3.2	Status and Control Register 2 (ADCSC2)	18-7
18.3.3	Data Result High Register (ADCRH).....	18-8
18.3.4	Data Result Low Register (ADCRL)	18-8
18.3.5	Compare Value High Register (ADCCVH)	18-9
18.3.6	Compare Value Low Register (ADCCVL)	18-9
18.3.7	Configuration Register (ADCCFG)	18-9
18.3.8	Pin Control 1 Register (APCTL1)	18-11
18.3.9	Pin Control 2 Register (APCTL2)	18-12
18.3.10	Pin Control 3 Register (APCTL3)	18-13
18.4	Functional Description	18-14
18.4.1	Clock Select and Divide Control	18-14
18.4.2	Input Select and Pin Control.....	18-15
18.4.3	Hardware Trigger	18-15
18.4.4	Conversion Control.....	18-15
18.4.5	Automatic Compare Function.....	18-18

18.4.6	MCU Wait Mode Operation	18-19
18.4.7	MCU Stop3 Mode Operation	18-19
18.4.8	MCU Stop2 Mode Operation	18-20
18.5	Initialization Information	18-20
18.5.1	ADC Module Initialization Example	18-20
18.6	Application Information	18-22
18.6.1	External Pins and Routing	18-22
18.6.2	Sources of Error	18-24

Chapter 19 Analog Comparator 3V

19.1	Introduction	19-1
19.1.1	Features	19-1
19.1.2	ACMP Configuration Information	19-1
19.1.3	ACMP/TPM Configuration Information	19-1
19.1.4	ACMP Clock Gating	19-1
19.1.5	Interrupt Vectors	19-2
19.1.6	Modes of Operation	19-2
19.1.7	Block Diagram	19-3
19.2	External Signal Description	19-3
19.3	Memory Map/Register Definition	19-4
19.3.1	ACMPx Status and Control Register (ACMPxSC)	19-4
19.4	Functional Description	19-5

Chapter 20 Development Support

20.1	Introduction	20-1
20.1.1	Forcing Active Background	20-1
20.1.2	Module Configuration	20-1
20.1.3	Features	20-1
20.2	Background Debug Controller (BDC)	20-2
20.2.1	BKGD Pin Description	20-3
20.2.2	Communication Details	20-3
20.2.3	BDC Commands	20-7
20.2.4	BDC Hardware Breakpoint	20-9
20.3	On-Chip Debug System (DBG)	20-10
20.3.1	Comparators A and B	20-10
20.3.2	Bus Capture Information and FIFO Operation	20-10
20.3.3	Change-of-Flow Information	20-11
20.3.4	Tag vs. Force Breakpoints and Triggers	20-11
20.3.5	Trigger Modes	20-12
20.3.6	Hardware Breakpoints	20-14
20.4	Register Definition	20-14
20.4.1	BDC Registers and Control Bits	20-14

20.4.2	System Background Debug Force Reset Register (SBD FR)	20-16
20.4.3	DBG Registers and Control Bits.	20-17

About This Book

This manual details the MC12311, which is a highly-integrated, cost-effective, system-in-package (SIP), sub-1GHz wireless node solution with an FSK, GFSK, MSK, or OOK modulation-capable transceiver and low-power QE32 8-bit microcontroller. The highly integrated RF transceiver operates over a wide frequency range including 315 MHz, 433 MHz, 470 MHz, 868 MHz, 915 MHz, 928 MHz, and 955 MHz in the license-free Industrial, Scientific and Medical (ISM) frequency bands.

Audience

This manual is intended for system designers.

Organization

This document is organized into 21 chapters.

Chapter 1	Introduction — Briefly introduces the MC12311.
Chapter 2	MC12311 Pins and Connections — Describes device pinout and functionality.
Chapter 3	System Considerations — Describes system level considerations of the MC12311 transceiver and MCU.
Chapter 4	Sub 1-Ghz Transceiver Architecture Description — This chapter describes the architecture and operation of the MC12311 low-power, highly integrated transceiver chip.
Chapter 5	Transceiver Operating Modes — Describes the operating modes of the MC12311 transceiver.
Chapter 6	Transceiver Digital Control and Communications — Details the MC12311 data processing circuit. Its role is to interface the data to/from the modulator/demodulator and the uC access points (SPI and DIO pins). It also controls all the configuration registers.
Chapter 7	MC12311 Transceiver-MCU Serial Peripheral Interface — Details how the MC12311 transceiver and CPU communicate primarily through the on board SPI interface.
Chapter 8	MCU Modes of Operation — Describes QE32 operating modes and describes entry into each mode, exit from each mode, and details the functionality while in each of the modes.
Chapter 9	MCU Memory — Describes on-chip memory in the QE32 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 10	MCU Resets, Interrupts, and System Configuration — This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the QE32.
Chapter 11	MCU Parallel Input/Output — This section explains software controls related to parallel input/output (I/O).

Chapter 12	MCU Internal Clock Source (ICS) — Shows the functional organization of the internal clock source (ICS) module and includes a general description and features list.
Chapter 13	MCU Central Processor Unit (CPU) — This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the QE32 Family.
Chapter 14	MCU Keyboard Interrupt (KBI) — Describes how the KBI module allows up to eight pins to act as additional interrupt sources.
Chapter 15	MCU Timer/PWM (TPM Module) — 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 16	MCU Serial Communications Interface (SCI) — This chapter describes the SCI which allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs.
Chapter 17	IIC Module — 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 18	Analog to Digital (ATD) Module — Details the ATD module which is an analog-to-digital converter with a successive approximation register (SAR) architecture with sample and hold.
Chapter 19	Analog Comparator 3V — Details the ACMP features and operation.
Chapter 20	Development Support — Describes the features of the background debug controller (BDC).

Revision History

The following table summarizes revisions to this document since the previous release (Rev 0.0).

Revision History

Location	Revision
Chapter 3	Application circuit updates.

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 MC12311 Data Sheet

Chapter 1

MC12311 Introduction

The MC12311 is a highly-integrated, cost-effective, system-in-package (SIP), sub-1GHz wireless node solution with an FSK, GFSK, MSK, or OOK modulation-capable transceiver and low-power QE32 8-bit microcontroller. The highly integrated RF transceiver operates over a wide frequency range including 315 MHz, 433 MHz, 470 MHz, 868 MHz, 915 MHz, 928 MHz, and 955 MHz in the license-free Industrial, Scientific and Medical (ISM) frequency bands. This configuration allows users to minimize the use of external components.

The MC12311 is targeted for the following low-power wireless applications:

- Automated Meter Reading
- Wireless Sensor Networks
- Home and Building Automation
- Wireless Alarm and Security Systems
- Industrial Monitoring and Control
- Wireless MBUS Standard (EN13757-4:2005)

Freescale supplements the MC12311 with tools and software that include hardware evaluation and development boards, software development IDE and applications, drivers, custom PHY usable with Freescale's IEEE 802.15.4 compatible MAC, and an available wireless MBUS solution.

1.1 Ordering Information

Table 1-1 lists the available devices in the MC12311 family.

Table 1-1. Devices in the MC12311 Family

Device	Operating Temp Range (TA.)	Package	Memory Options	Description
MC12311	-40° to 85° C	LGA	1KB RAM, 16KB Flash	Intended for proprietary applications and Freescale Simple MAC (SMAC)
MC12311	-40° to 85° C	LGA Tape and Reel	1KB RAM, 16KB Flash	Intended for proprietary applications and Freescale Simple MAC (SMAC)

1.2 General Platform Features

- Sub-1 GHz on board transceiver
- Multiple power saving modes
- 1.8 V to 3.6 V operating voltage with on-chip voltage regulators
- -40°C to +85°C temperature range

- Low external component count
- Supports single 32 MHz crystal clock source operation or dual crystal operation
- Versatile software solutions
- 60-pin LGA (8x8 mm) Package

1.3 9S08QE32 Microcontroller Features

- 8-Bit QE32 Central Processor Unit (CPU) with CPU clock rate up to:
 - 50.33 MHz at 3.6 V to 2.4V
 - 40 MHz at 2.4V to 2.1V
 - 20 MHz at 2.1V to 1.8V
- HC08 instruction set with added BGND instruction
- Support for up to 32 interrupt/reset sources
- 32 KB Flash and 2 KB RAM
- Low power modes (Wait plus Stop2 and Stop3 modes)
- Dedicated serial peripheral interface (SPI) connected internally to sub-1 GHz transceiver
- Multiple clock source options
- Internal clock generator (ICG) with 243 kHz oscillator that has +/-0.2% trimming resolution and +/-0.5% deviation across voltage.
- Startup oscillator of approximately 8 MHz
- External crystal or resonator
- External source from transceiver clock for very high accuracy source or system low-cost option
- In-circuit debug and FLASH programming available via on-chip background debug module (BDM)
- System protection features
 - Programmable low voltage interrupt (LVI)
 - Optional watchdog timer (COP)
 - Illegal opcode detection
- Peripherals
 - ADC — 10-channel, 12-bit resolution
 - ACMPx — Two analog comparators with selectable interrupt on rising, falling, or either edge of comparator output
 - SCIx — Two serial communications interface modules with optional 13-bit break.
 - IIC — One IIC; up to 100 kbps
 - TPMx — One 6-channel (TPM3) and two 3-channel (TPM1 and TPM2)
 - RTC — (Real-time counter) 8-bit modulus counter with binary or decimal based prescaler
- KBI — Two 8-bit port keyboard interrupt modules
- Input/Output
 - Up to 33 GPIO including dedicated GPIO supporting transceiver

- 13 KBI interrupts with selectable polarity
- Hysteresis and configurable pull-up device on all input pins; Configurable slew rate and drive strength on all output pins.
- Hysteresis and configurable pull up device on all input pins; Configurable slew rate and drive strength on all output pins.

1.4 RF Transceiver Features

- High Sensitivity: down to -120 dBm at 1.2 kbps
- High Selectivity: 16-tap FIR Channel Filter
- Bullet-proof front end: IIP3 = -18 dBm, IIP2 = +35 dBm, 80 dB Blocking Immunity, no Image Frequency response
- Low current: Rx = 16mA, 100nA register retention
- Programmable Pout : -18 to +17 dBm in 1 dB steps
- Constant RF performance over voltage range of chip
- FSK bit rates up to 300 kbps
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK and OOK modulations
- Built-in Bit Synchronizer performing Clock recovery
- Incoming Sync Word Recognition
- Automatic RF Sense with ultra-fast AFC
- Packet engine with CRC, AES-128 encryption and 66-byte FIFO
- Built-in temperature sensor and Low battery indicator
- 32 MHz crystal oscillator clock source

1.5 Software Solutions

Freescale will support the MC12311 platform with several software solutions:

- SMAC (Simple Media Access Controller) - This codebase provides simple communication and test applications based on drivers/PHY utilities available as source code. This environment is useful for hardware and RF debug, hardware standards certification, and developing proprietary applications.
- IEEE 802.15.4 MAC with custom PHY layer - The Freescale MAC is a robust, mature codebase useful for developing networking solutions. Freescale is implementing an IEEE 802.15.4 MAC-compatible custom sub-1 GHz PHY template that can be used across different frequency bands. This capability allows users to build powerful networking solutions on a known, stable codebase.
- Wireless MBUS stack - Freescale is porting an existing wireless MBUS codebase to the MC12311 platform which will be available through an external partner.

The Freescale MC12311 solutions are provided through 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. The MC12311 products are available as codebases within BeeKit; for the wireless MBUS stack, BeeKit will have simple demo applications only.

1.6 System Overview

Figure 1-1 shows a simplified block diagram of the MC12311.

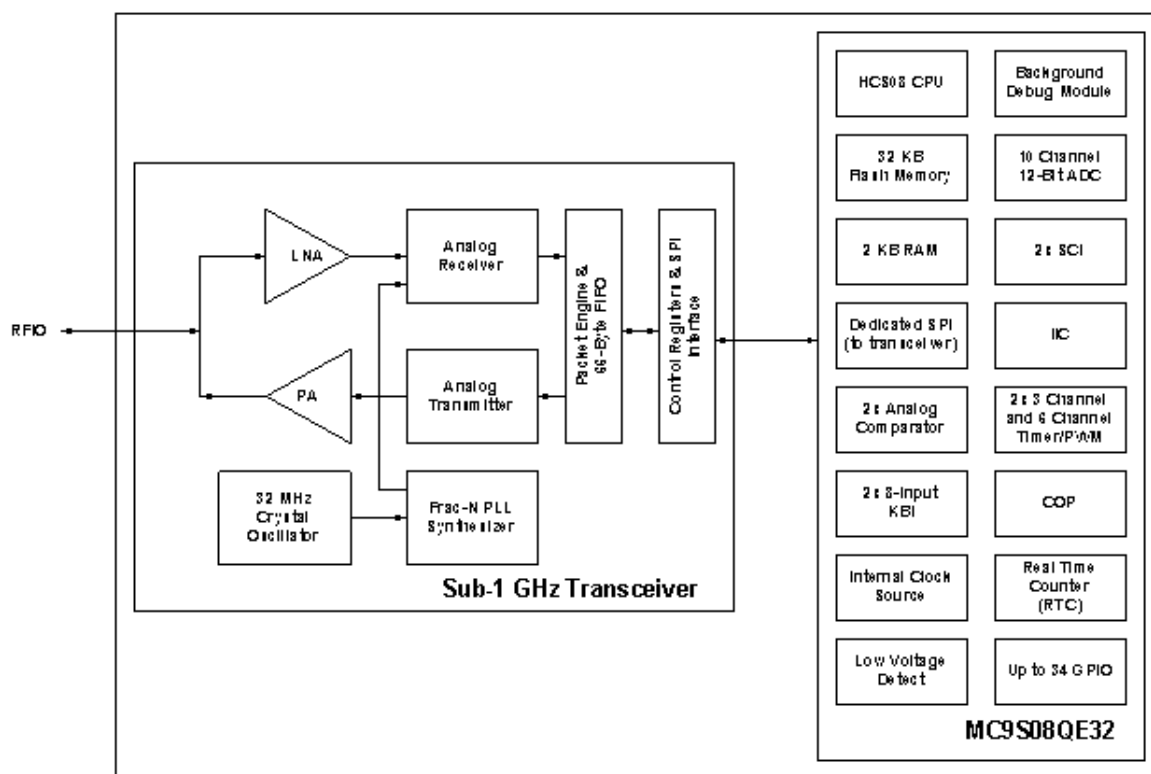


Figure 1-1. MC12311 System Level Block Diagram

1.6.1 Transceiver Overview

The transceiver (see Figure 1-1) is a single-chip integrated circuit ideally suited for today's high performance ISM band RF applications. Its advanced features set, including state of the art packet engine, greatly simplifies system design while the high level of integration reduces the external RF component bill of material (BOM) to a handful of passive de-coupling and matching components. It is intended for use as a high-performance, low-cost FSK and OOK RF transceiver for robust, frequency agile, half-duplex bi-directional RF links.

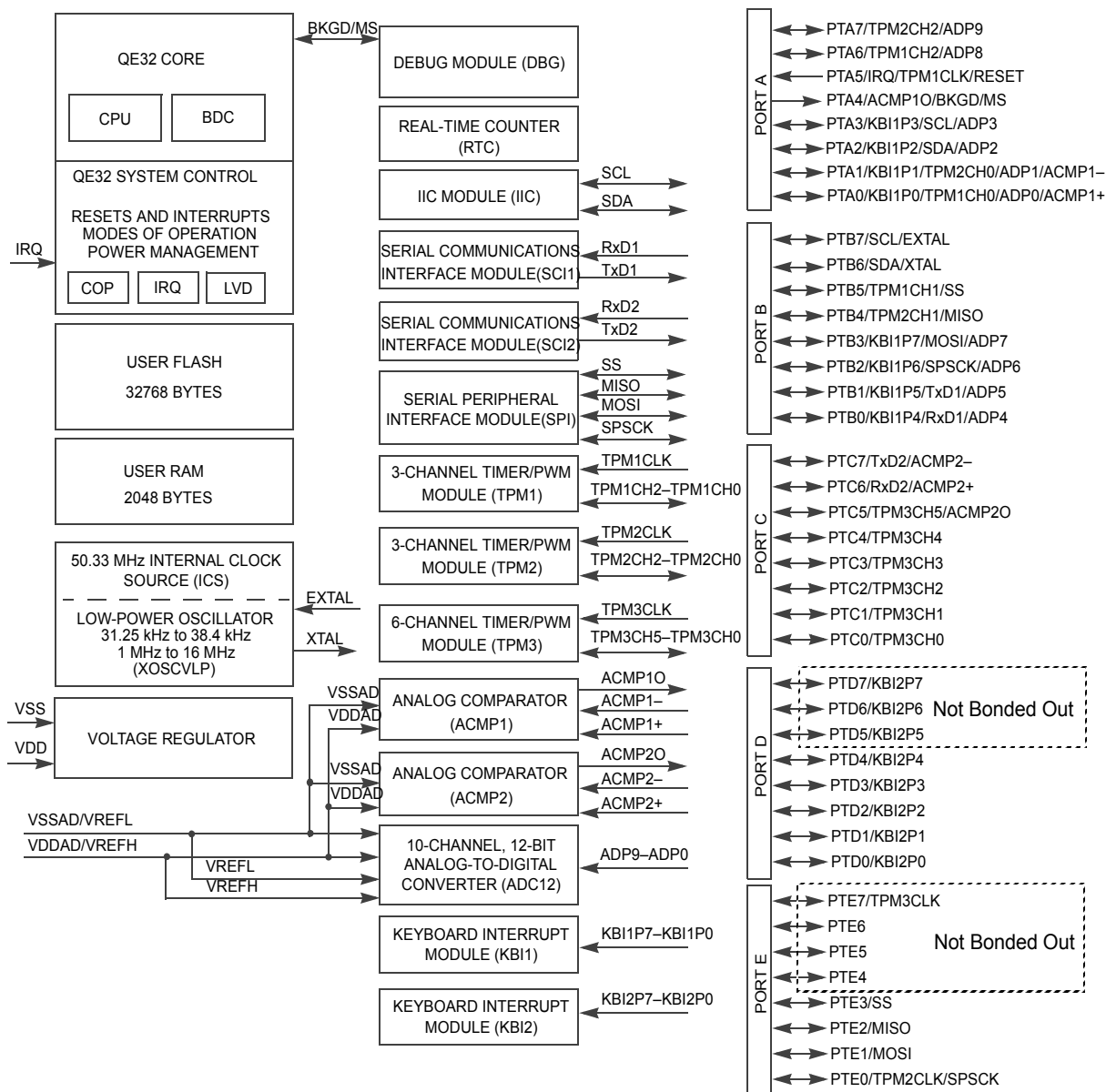
The MC12311 is intended for applications over a wide frequency range, including the 433 MHz and 868 MHz European and the 902-928 MHz North American ISM bands. Coupled with a link budget in excess of 135 dB, the transceiver advanced system features include a 66 byte TX/RX FIFO, configurable automatic packet handler, listen mode, temperature sensor and configurable DIOs which greatly enhance

system flexibility while at the same time significantly reducing MCU requirements. The transceiver complies with both ETSI and FCC regulatory requirements.

The major RF communication parameters of the MC12311 transceiver are programmable and most can be dynamically set. This feature offers the unique advantage of programmable narrow-band and wide-band communication modes without the need to modify external components. The transceiver is also optimized for low power consumption while offering high RF output power and channelized operation.

1.6.2 MCU Block Diagram

The on board 9S08QE32 MCU integrated circuit features an HC08 8-bit CPU, 2 KB RAM, 32 KB Flash memory, and a rich set of peripherals. The RF transceiver is controlled through the MCU SPI port which is dedicated to the RF device interface. Two of the transceiver status IO lines are also directly connected to the MCU GPIO to monitor the transceiver operation. In addition, the transceiver reset and additional status can be connected to the MCU through external connections.



Notes:

1. When PTA5 is configured as RESET, pin becomes bi-directional with output being open-drain drive containing an internal pull-up device.

Figure 1-2. MCU Block Diagram

1.6.3 QE32 Central Processing Unit (CPU)

The on board CPU is a 40 MHz 8-bit QE32 CPU. It executes the HC08 instruction set with added BGND instruction and supports for up to 32 interrupt/reset sources.

1.6.4 MCU Peripheral Modules

The MCU includes the following peripheral modules:

- Dedicated serial peripheral interface (SPI) connected internally to the transceiver - 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 transceiver and internally tied to the transceiver SPI port, the MCU SPI port has to be programmed as a master and meet the timing requirement of the transceiver SPI interface
- One 6-channel (TPM3) 16-bit timer/pulse width modulator module and two 3-channel (TPM1 and TPM2) 16-bit timer/pulse width modulator modules, each with selectable input capture, output capture, and PWM capability.
- Two 8-bit port keyboard interrupt (KBI) modules with programmable edge and level IRQ capability
- 10-channel, 12-bit resolution ADC - 2.5 ms conversion time; automatic compare function; 1.7 mV/°C temperature sensor; internal bandgap reference channel; operation in stop3; fully functional from 3.6 V to 1.8 V
- Two analog comparators with selectable interrupt on rising, falling, or either edge of comparator output; compare option to fixed internal bandgap reference voltage; outputs can be optionally routed to TPM module; operation in Stop3
- Two independent serial communication interfaces (SCI1 and SCI2) with optional 13-bit break. Full duplex non-return to zero (NRZ); LIN master extended break generation; LIN slave extended break detection; wake on active edge
- Internal Clock Source (ICS) - Supports external clock source or external crystal oscillator and has an trimmable internal reference clock (32 kHz nominal). In addition to supporting two clock sources, the ICS includes an FLL that can be used to supply higher frequency system clocks based on the selected source. Default out of reset is the internal reference selected with the FLL engaged which provides a nominal 17 MHz CPU clock for boot.
- Real-time counter (RTC) - 8-bit modulus counter with binary or decimal based prescaler; can be clocked from:
 - Optional external clock source (ERCLK) for precise time base, time-of-day, calendar or task scheduling functions
 - The trimmable 32 kHz internal reference clock (ICSIRCLK)
 - 1 kHz internal low power oscillator (LPO).
- Inter-integrated circuit (IIC) with up to 100 kbps with maximum bus loading; multi-master operation; programmable slave address; interrupt driven byte-by-byte data transfer; supports broadcast mode and 10-bit addressing
- In-circuit debug and flash programming available via on-chip background debug module (BDM)
- **Up to 32 MCU GPIO with programmable pullups - The QE32 MCU has a total of 7 ports with each having 8 GPIO. Because of the MC12311 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 11, “MCU Parallel Input/Output”. Unused GPIO and GPIO not pinned-out should be initialized to a known condition.**

1.6.5 MCU Internal Clock Distribution

Figure 1-3 shows a simplified MCU clock distribution diagram. Some modules in the MCU have selectable clock inputs. The clock inputs to the modules indicate the clock(s) used to drive the module function. All memory-mapped registers associated with the modules are clocked with BUSCLK.

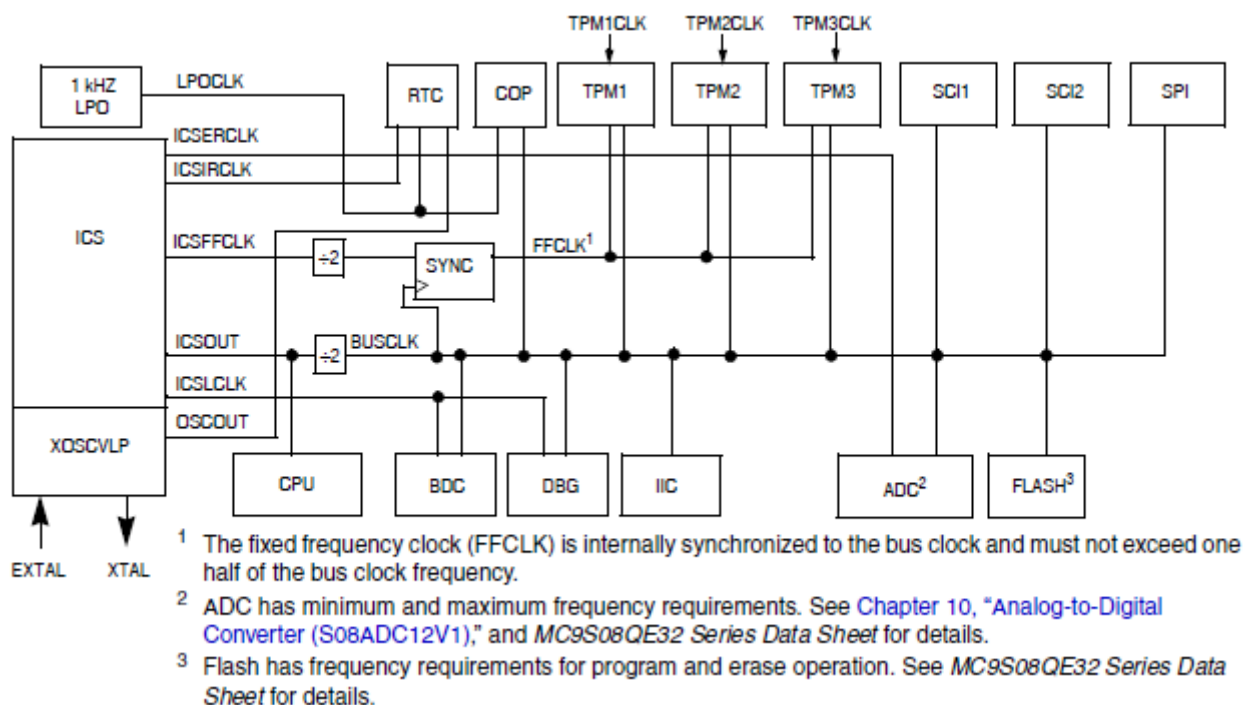


Figure 1-3. MCU Internal Clock Distribution

The clocks to the MCU are all derived from the ICS module:

- ICSOUT — This clock source is used as the CPU clock and is divided by two to generate the peripheral bus clock, BUSCLK. Control bits in the ICS control registers determine which of three clock sources is connected:
 - Internal reference clock
 - External reference clock
 - Frequency-locked loop (FLL) output
- ICSLCLK — This clock source is derived from the digitally controlled oscillator, DCO, of the ICS when the ICS is configured to run off the internal or external reference clock. Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICSERCLK — This is the external reference clock and can be selected as the alternate clock for the ADC module. The “Optional External Reference Clock” section in Chapter 11, “Internal Clock Source (S08ICSV3),” explains the ICSERCLK in more detail. See Chapter 13, “Real-Time Counter (S08RTCV1)” and Chapter 10, “Analog-to-Digital Converter (S08ADC12V1)” for more information regarding the use of ICSERCLK with these modules.

- **ICSIRCLK** — This is the internal reference clock and can be selected as the real-time counter clock source. Chapter 11, “Internal Clock Source (S08ICSV3)” explains the ICSECLK in more detail. See Chapter 13, “Real-Time Counter (S08RTCV1),” for more information regarding the use of ICSIRCLK.
- **ICSFFCLK** — This generates the fixed frequency clock (FFCLK) after being synchronized to the bus clock. It can be selected as clock source for the TPM modules. The ICSFFCLK frequency is determined by the ICS settings. See the “Fixed Frequency Clock” section in Chapter 11, “Internal Clock Source (S08ICSV3).”
- **LPOCLK** — This clock is generated from an internal low power oscillator completely independent of the ICS module. The LPOCLK can be selected as the clock source to the RTC and COP modules. To use the LPOCLK with these modules, see Chapter 13, “Real-Time Counter (S08RTCV1),” and Section 5.4, “Computer Operating Properly (COP) Watchdog.”
- **OSCOUT** — This is the XOSCVLP module output and can be selected as the real-time counter clock source.
- **TPM_xCLK** — TPM_xCLK is the optional external clock source for the TPM modules. The TPM_xCLK must be limited to one fourth the frequency of the bus clock for synchronization. See Section 16.2.1.1, “EXTCLK — External Clock Source.”



Chapter 2

MC12311 Pins and Connections

2.1 Device Pin Assignment

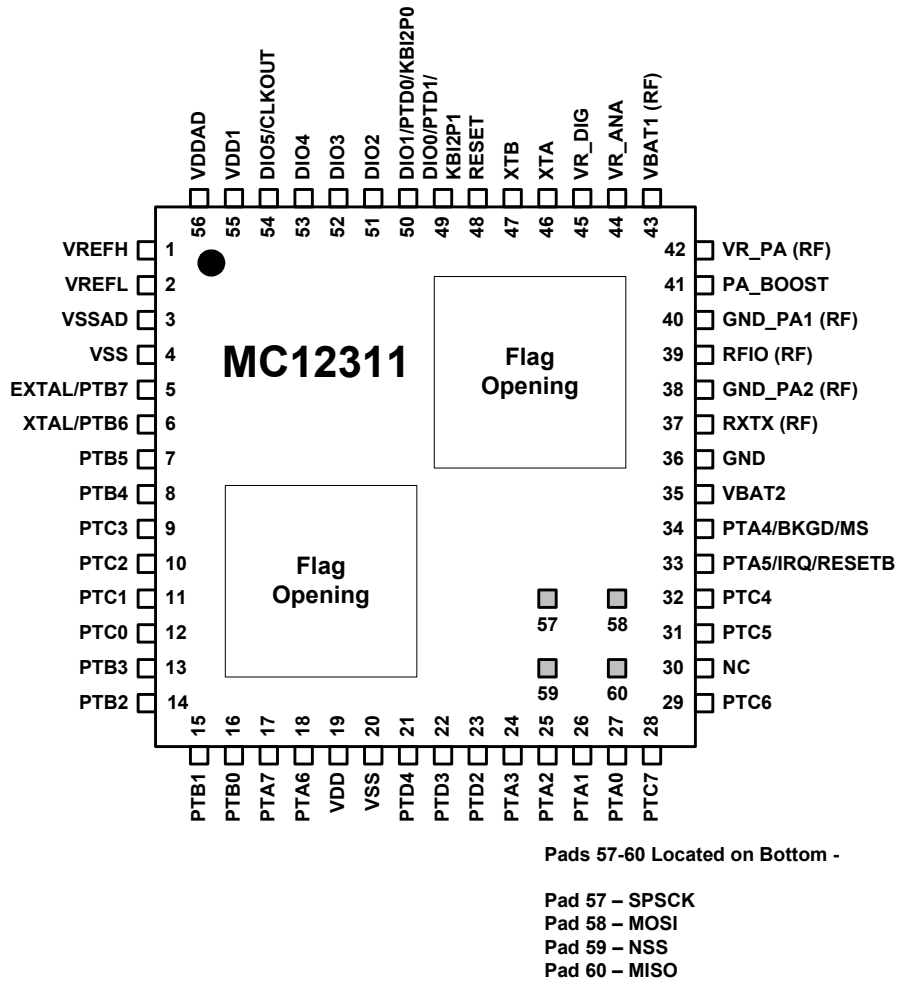


Figure 2-1. MC12311 Pinout

2.2 Pin Definitions

Table 2-1 details the MC12311 pinout and functionality.

Table 2-1. Pin Function Description

Pin #	Pin Name	Type	Description	Functionality
1	VREFH	Input	MCU high reference voltage for ATD	VREFH
2	VREFL	Input	MCU low reference voltage for ATD	VREFL
3	VSSAD	Power Input	MCU ADC Ground	Connect to ground
4	VSS	Power Input	MCU Ground	Connect to ground
5	EXTAL/SCL/PTB7 ¹	Input / Digital Input/Output	MCU Crystal Oscillator input / IIC bus clock / Port B Bit 7	<ul style="list-style-type: none"> Normally used as MCU clock source Driven from transceiver ClkOut (DIO5)
6	XTAL/SCA/PTB6 ¹	Output / Digital Input/Output	MCU Crystal Oscillator output / IIC bus data / Port B Bit 6	Normally used as MCU clock source
7	PTB5/TPM1CH1 ²	Digital Input/Output	MCU Port B Bit 5 / TPM1 Channel 1	See Footnote 1
8	PTB4/TPM2CH1 ²	Digital Input/Output	MCU Port B Bit 4 / TPM2 Channel 1	See Footnote 1
9	PTC3/TPM3CH3	Digital Input/Output	MCU Port C Bit 3 / TPM3 Channel 3	
10	PTC2/TPM3CH2	Digital Input/Output	MCU Port C Bit 2 / TPM3 Channel 2	
11	PTC1/TPM3CH1	Digital Input/Output	MCU Port C Bit 1 / TPM3 Channel 1	
12	PTC0/TPM3CH0	Digital Input/Output	MCU Port C Bit 0 / TPM3 Channel 0	
13	PTB3/KBI1P7/ ADP7 ²	Digital Input/Output / Analog Input	MCU Port B Bit 3 / KBI1 Input Bit 7 / ADC Analog Channel 7	See Footnote 1
14	PTB2/KBI1P6/ ADP6 ²	Digital Input/Output / Analog Input	MCU Port B Bit 2 / KBI1 Input Bit 6 / ADC Analog Channel 6	See Footnote 1
15	PTB1/KBI1P5/ ADP5	Digital Input/Output / Analog Input	MCU Port B Bit 1 / KBI1 Input Bit 5 / ADC Analog Channel 5	
16	PTB0/KBI1P4/ ADP4	Digital Input/Output / Analog Input	MCU Port B Bit 0 / KBI1 Input Bit 4 / ADC Analog Channel 4	
17	PTA7/TPM2CH2/ ADP9	Digital Input/Output / Analog Input	MCU Port A Bit 7 / TPM2 Channel 2 / ADC Analog Channel 9	

Table 2-1. Pin Function Description (continued)

Pin #	Pin Name	Type	Description	Functionality
18	PTA6/TPM1CH2/ ADP8	Digital Input/Output / Analog Input	MCU Port A Bit 6 / TPM1 Channel 2 / ADC Analog Channel 8	
19	VDD	Power Input	MCU VDD	Connect to MC12311 VDD supply
20	VSS	Power Input	MCU Ground	Connect to ground
21	PTD4/KBI2P4	Digital Input/Output	MCU Port D Bit 4 / KBI2 Input Bit 4	
22	PTD3/KBI2P3	Digital Input/Output	MCU Port D Bit 3 / KBI2 Input Bit 3	
23	PTD2/KBI2P2	Digital Input/Output	MCU Port D Bit 2 / KBI2 Input Bit 2	
24	PTA3/KBI1P3/ SCL/ADP3 ¹	Digital Input/Output / Analog Input	MCU Port A Bit 3 / KBI1 Input Bit 3 / IIC bus clock / ADC Analog Channel 3	
25	PTA2/KBI1P2/ SCA/ADP2 ²	Digital Input/Output / Analog Input	MCU Port A Bit 2 / KBI1 Input Bit 2 / IIC bus data / ADC Analog Channel 2	
26	PTA1/KBI1P1/ TPM2CH0/ADP1/ ACMP1- ³	Digital Input/Output / Analog Input	MCU Port A Bit 1 / KBI1 Input Bit 1 / TPM2 Channel 0 / ADC Analog Channel 1 / ACMP1 Analog Input Minus	
27	PTA0/KBI1P0/ TPM1CH0/ADP0/ ACMP1+ ³	Digital Input/Output / Analog Input	MCU Port A Bit 0 / KBI1 Input Bit 0 / TPM1 Channel 0 / ADC Analog Channel 0 / ACMP1 Analog Input Plus	
28	PTC7/TXD2/ ACMP2-	Digital Input/Output / Analog Input	MCU Port C Bit 7 / SCI2 TXD / ACMP2 Analog Input Minus	
29	PTC6/RXD2/ ACMP2+	Digital Input/Output	MCU Port C Bit 6 / SCI2 RXD / ACMP2 Analog Input Plus	
30	NC		No Connect	
31	PTC5/TPM3CH5/ ACMP2O	Digital Input/Output	MCU Port C Bit 5 / TPM3 Channel 5 / ACMP2 Output	
32	PTC4/TPM3CH4	Digital Input/Output	MCU Port C Bit 5 / TPM3 Channel 5 / ACMP2 Output	
33	PTA5/IRQ/ TPM1CLK/ RESETB	Digital Input/Output	MCU Port A Bit 5 / MCU IRQ / TPM1 Clock / MCU RESET	
34	PTA4/ACMP1O/ BKGD/MS	Digital Input/Output	MCU Port A Bit 4 / ACMP1 Output / Background Debug Port / Mode Select	
35	VBAT2	Power Input	Transceiver VDD	Connect to MC12311 VDD supply

Table 2-1. Pin Function Description (continued)

Pin #	Pin Name	Type	Description	Functionality
36	GND	Power Input	Transceiver Ground	Connect to ground
37	RXTX (RF)	Digital Output	Transceiver Rx/Tx RF Switch Control Output; high when in TX	
38	GND_PA2 (RF)	Power Input	Transceiver RF Ground	Connect to ground
39	RFIO (RF)	RF Input/Output	Transceiver RF Input/Output	
40	GND_PA1 (RF)	Power Input	Transceiver RF Ground	Connect to ground
41	PA_BOOST	RF Output	Transceiver Optional High-Power PA Output	
42	VR_PA	Power Output	Transceiver regulated output voltage for VR_PA use.	
43	VBAT1 (RF)	Power Input	Transceiver VDD for RF circuitry	Connect to MC12311 VDD supply
44	VR_ANA	Power Output	Transceiver regulated output voltage for analog circuitry.	Decouple to ground with 100 nF capacitor
45	VR_DIG	Power Output	Transceiver regulated output voltage for digital circuitry.	Decouple to ground with 100 nF capacitor
46	XTA	Xtal Osc	Transceiver crystal reference oscillator	Connect to 32 MHz crystal and load capacitor
47	XTB	Xtal Osc	Transceiver crystal reference oscillator	Connect to 32 MHz crystal and load capacitor
48	RESET	Digital Input	Transceiver hardware reset input	<ul style="list-style-type: none"> • Active high • Typically driven from MCU GPIO
49	DIO0/PTD1/KBI2P1	Digital Input/Output	Transceiver GPIO Bit 0 / MCU Port D Bit 1 / MCU KBI2 Bit 1	MCU IO and Transceiver IO connected on board MC12311
50	DIO1/PTD0/KBI2P0	Digital Input/Output	Transceiver GPIO Bit 1 / MCU Port D Bit 0 / MCU KBI2 Bit 0	MCU IO and Transceiver IO connected on board MC12311
51	DIO2	Digital Input/Output	Transceiver GPIO Bit 2	
52	DIO3	Digital Input/Output	Transceiver GPIO Bit 3	
53	DIO4	Digital Input/Output	Transceiver GPIO Bit 4	
54	DIO5/CLKOUT	Digital Input/Output	Transceiver GPIO Bit 5 / ClkOut	Commonly programmed as ClkOut to supply MCU clock; connect to Pin 5
55	VDD1	Power Input	MCU VDD supply	Connect to MC12311 VDD supply
56	VDDAD	Power Input	MCU ADC VDD	Connect to MC12311 VDD supply
57	SPSCK ²	Digital Input/Output	SPI Port Clock driven from MCU Port E Bit 0	<ul style="list-style-type: none"> • MCU IO and Transceiver IO connected on board MC12311 • MCU IO must be configured for this connection

Table 2-1. Pin Function Description (continued)

Pin #	Pin Name	Type	Description	Functionality
58	MOSI ²	Digital Input/Output	SPI Port MOSI signal connected to MCU Port E Bit 1	<ul style="list-style-type: none"> MCU IO and Transceiver IO connected on board MC12311 MCU IO must be configured for this connection
59	NSS ²	Digital Input/Output	SPI Port \overline{SS} signal connected to MCU Port E Bit 3	<ul style="list-style-type: none"> MCU IO and Transceiver IO connected on board MC12311 MCU IO must be configured for this connection
60	MISO ²	Digital Input/Output	SPI Port MISO signal connected to MCU Port E Bit 2	<ul style="list-style-type: none"> MCU IO and Transceiver IO connected on board MC12311 MCU IO must be configured for this connection
FLAG	VSS	Power input	External package flag. Common VSS	Connect to ground.

¹ IIC pins, SCL and SDA can be repositioned using the IICPS bit in SOPT2; default locations are PTA3 and PTA2.

² MCU GPIO PTB7, PTB6, PTB3, and PTB2 default to SPI Port Pins. THEY MUST BE OVERPROGRAMMED to position the SPI Port on port pins PTE3, PTE2, PTE1, and PTE0 because the SPI Port is reserved for transceiver communication and is connected on board the package. Use SPIPS bit in Register SOPT2.

³ If ADC and ACMP1 are both enabled, both modules have access to the pin.

2.3 Internal Functional Interconnects

The MCU provides control to the transceiver through the SPI Port and receives status from the transceiver from the DIOx pins. Certain interconnects between the devices are routed on board the SiP. In addition, the signals are brought out to external pads.

Table 2-2. MC12311 Internal Functional Interconnects

Pin #	MCU Signal	Transceiver Signal	Description
49	PTD1/KBI2P1	DIO0	Transceiver DIO0 can be programmed to provide status to the MCU
50	PTD0/KBI2P0	DIO1	Transceiver DIO1 can be programmed to provide status to the MCU
57	PTE0/SPSCK	SCK	MCU SPI connection must be initiated, not default
58	PTE1/MOSI	MOSI	MCU SPI connection must be initiated, not default
59	PTE3/SS	NSS	MCU SPI connection must be initiated, not default
60	PTE2/MISO	MISO	MCU SPI connection must be initiated, not default

NOTE

- As shown in [Table 2-2](#), the MCU SPI Port pin selection must be configured by software by writing the SPIPS bit in Register SOPT2
- The transceiver DIO pins must be programmed to provide desired status

Chapter 3

System Considerations

3.1 Introduction

The MC12311 is the embodiment of a sub-1 GHz wireless node in a single SiP package. All control of the node is done through the on board 9S08QE32 processor, and all MCU peripherals, MCU GPIO, transceiver functionality, and transceiver 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 transceiver is through the common SPI bus and several MCU GPIO lines. Primary interface with the transceiver is through the SPI command structure that allows reading/writing registers and provides initialization of parameters, reading of status, and control of transceiver operation. The transceiver also has two status signals tied to MCU GPIO internally.

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 MC12311 device and the following chapters present individual functions in detailed descriptions.

3.2 Power Connections

The MC12311 power connections at the SiP level are listed in [Table 3-1](#).

Table 3-1. Power Pin Descriptions

Pin #	Pin Name	Type	Description	Functionality
3	VSSAD	Power Input	MCU ADC Ground	Connect to ground
4	VSS	Power Input	MCU Ground	Connect to ground
19	VDD	Power Input	MCU VDD	Connect to MC12311 VDD supply
20	VSS	Power Input	MCU Ground	Connect to ground
35	VBAT2	Power Input	Transceiver VDD	Connect to MC12311 VDD supply
36	GND	Power Input	Transceiver Ground	Connect to ground
38	GND_PA2 (RF)	Power Input	Transceiver RF Ground	Connect to ground
40	GND_PA1 (RF)	Power Input	Transceiver RF Ground	Connect to ground
42	VR_PA	Power Output	Transceiver regulated output voltage for VR_PA use.	

Table 3-1. Power Pin Descriptions (continued)

Pin #	Pin Name	Type	Description	Functionality
43	VBAT1 (RF)	Power Input	Transceiver VDD for RF circuitry	Connect to MC12311 VDD supply
44	VR_ANA	Power Output	Transceiver regulated output voltage for analog circuitry.	Decouple to ground with 100 nF capacitor
45	VR_DIG	Power Output	Transceiver regulated output voltage for digital circuitry.	Decouple to ground with 100 nF capacitor
55	VDD1	Power Input	MCU VDD supply	Connect to MC12311 VDD supply
56	VDDAD	Power Input	MCU ADC VDD	Connect to MC12311 VDD supply
FLAG ¹	VSS	Power input	External package flag. Common VSS	Connect to ground.

¹ Flags on bottom of package are electrically separate. Both must be connected to ground.

When designing power to the MC12311 SiP, the following points need to be considered:

- The SiP package has two ground flags (VSS) on board the package. These are separated and both must be connected to system ground.
- The MCU VDD power supply connections include -
 - VDD (Pin 19)
 - VDD1 (Pin 55)
 - The VDDAD (Pin 56) analog supply to the MCU ATD is also normally wired to the common source supply.
- For the transceiver the primary power inputs include -
 - VBAT2 (Pin 35)
 - VBAT1 (Pin 43) for the RF circuitry
- The transceiver provides on board voltage regulator outputs for bypassing -
 - VR_ANA (Pin 44) regulated voltage to analog circuitry; bypass to ground
 - VR_DIG (Pin 45) regulated voltage to digital circuitry; bypass to ground
- The transceiver provides a regulated output VR_PA (Pin 42) for with RF power boost mode.
- Additional system ground pins include -
 - VSS (Pin 4 and Pin 20) - MCU ground
 - VSSAD (Pin 3) - MCU ATD ground
 - GND (Pin 36) - transceiver ground
 - GND_PA1 (Pin 40) and GND_PA2 (Pin 38) - transceiver RF grounds

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

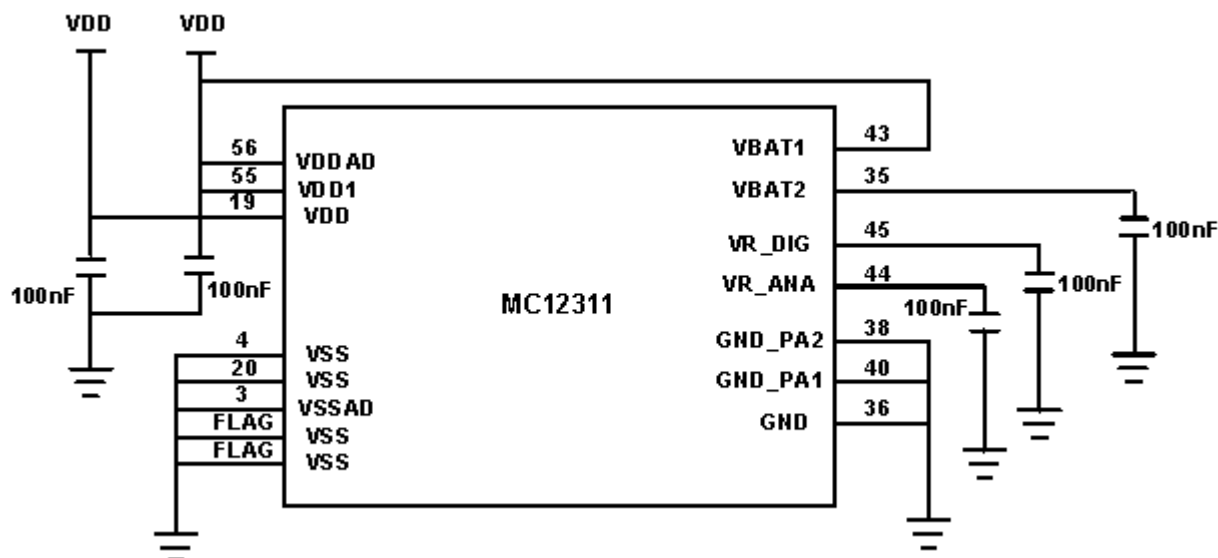


Figure 3-1. MC12311 Power Supply Connections

NOTE

Depending on the application, the VREFH high reference voltage for the ATD module is commonly also tied to the VDD common supply and the VREFL low reference voltage is tied to ground.

3.3 System Functional Interconnects

The MC12311 is comprised of two separate devices in a single package. The MCU controls the transceiver and there can be connections between the devices for several functions.

- Some connections are provided on board
- Additional external connections may also be used depending on the application needs.
 - Transceiver reset
 - Clock interconnect
 - Additional transceiver status

3.3.1 On board Connections (SPI Channel and Status)

The internal (on board) device connections are listed in [Table 2-2](#). These include:

- SPI communication channel - see [Chapter 7, “MC12311 Transceiver - MCU SPI Interface”](#)
- Transceiver DIO0 and DIO1 outputs as status - these status outputs are used to manage the data flow and transceiver sequencer during radio packet mode operation.

3.3.2 System Reset

The MC12311 system does not have a master input that resets the entire SiP:

- The MCU reset input is PTA5/IRQ/RESETB - use of this pin as an external reset is optional, and the MCU has a separate power-on reset (POR).
- The transceiver has an independent reset pin (RESET) - this signal is typically connected to an MCU GPIO to provide total software control of the transceiver

NOTE

It is recommended that the MCU be connected to the transceiver reset via an MCU GPIO pin to provide best overall hardware control.

3.3.2.1 MCU Reset Input PTA5/IRQ/RESETB

After a power-on reset (POR), the PTA5/IRQ/TPM1CLK/RESET pin defaults to a general-purpose input port pin, PTA5. Setting RSTPE in SOPT1 configures the pin to be the $\overline{\text{RESET}}$ pin with an open-drain drive containing an internal pullup device. After configured as $\overline{\text{RESET}}$, the pin remains $\overline{\text{RESET}}$ until the next LVD or POR. The $\overline{\text{RESET}}$ pin when enabled can be used to reset the MCU from an external source when the pin is driven low.

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

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

NOTE

- This pin does not contain a clamp diode to VDD and must not be driven above VDD
- The voltage on the internally pulled up RESET pin when measured is below VDD. The internal gates connected to this pin are pulled to VDD. If the RESET pin is required to drive to a VDD level an external pullup must be used.
- In EMC-sensitive applications, an external RC filter is recommended on the RESET pin, if enabled

3.3.2.2 Transceiver Reset

The transceiver can be reset via two means:

- A power-on reset (POR) of the MC12311 transceiver is triggered when VDD is applied.
- A hardware reset can be issued by controlling Pin 48 (transceiver RESET).

3.3.2.2.1 Transceiver POR

Similar to the MCU, a transceiver POR is internally generated when power is applied to VDD. See Figure 3-2 for the transceiver POR timing diagram.

- The transceiver hardware RESET pin is bidirectional and is first driven to high as the result of the POR.
- The RESET signal is then driven low, and the application must wait for 10 ms from the end of the POR cycle before commencing communications over the SPI bus.
- RESET should be left floating during the POR sequence.

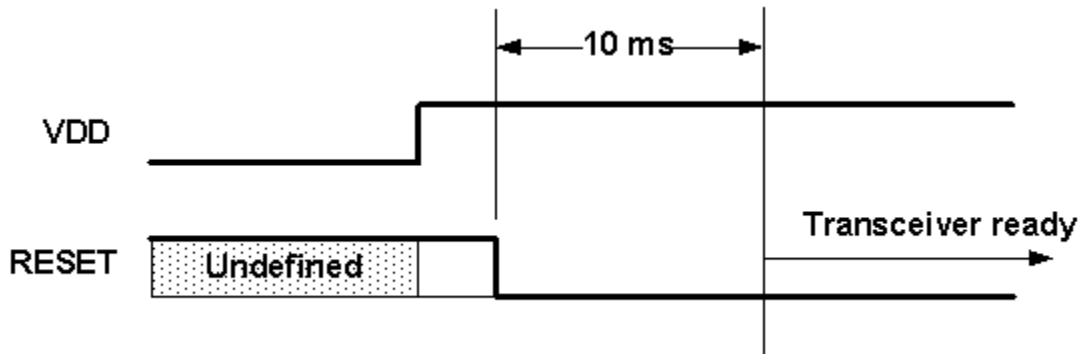


Figure 3-2. POR Timing Diagram

NOTE

Any CLKOUT activity can also be used to detect when the chip is ready.

3.3.2.2.2 Transceiver Hardware Reset

A hardware reset of the MC12311 is also possible by asserting RESET high for a minimum of one hundred microseconds, and then releasing. The application must wait 5 ms before using the chip.

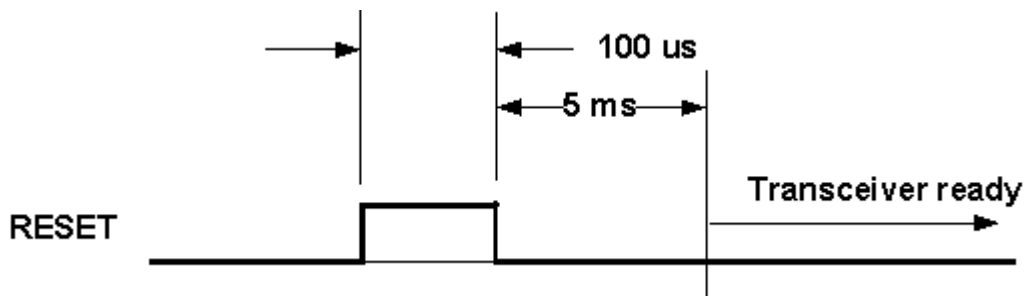


Figure 3-3. Manual Reset Timing Diagram

NOTE

While RESET is driven high, additional current consumption of up to ten milliamps may be seen on VDD.

3.3.2.3 MCU Control of Transceiver Reset

It is recommended to provide hardware reset capability of the transceiver via an MCU GPIO externally connected to the transceiver RESET pin. For Freescale applications software, MCU signal PTC0 is the preferred GPIO to control the RESET.

3.3.3 External Clock Connections

It is possible that the transceiver can supply a clock source to the MCU through external connections. See [Section 3.3.3, “External Clock Connections”](#).

3.3.4 Additional Transceiver Status Signals

The MC12311 transceiver has a total of six outputs (DIO5:DIO0) that can be programmed as status indicators:

- DIO1 and DIO0 are connected to MCU GPIO on board the package - these are utilized by applications software from transmitting and receiving data packets.
- At the user’s discretion, the additional DIO5:DIO2 can be connected externally to the MCU GPIO and programmed as status indicators -
 - Use of selected signals must be programmed on the transceiver
 - If interrupt request (IRQ) capability is desired, any status must be connected to a KBI input or to the MCU IRQ input. If the MCU IRQ signal is thus used, the MCU cannot have a hardware reset as these functions share the same pin. The KBI inputs must be programmed for IRQ use.

3.4 System Clock Sources and Configurations

The MC12311 clock connections are shown in [Figure 3-3](#). The 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 32 MHz crystal for the transceiver reference oscillator
- The transceiver optionally provides a ClkOut 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
- The MCU contains an internal nominal 32 kHz clock oscillator (which can be trimmed) that can be used to run the MCU for low power operation
- Out of reset, the MCU uses the internal oscillator and the on board FLL to generate an approximately 8 MHz clock for start-up. This allows recovery from stop or reset without a long crystal start-up delay

In addition, the transceiver has an on board RC oscillator that is used only internally for triggering periodic listen modes.

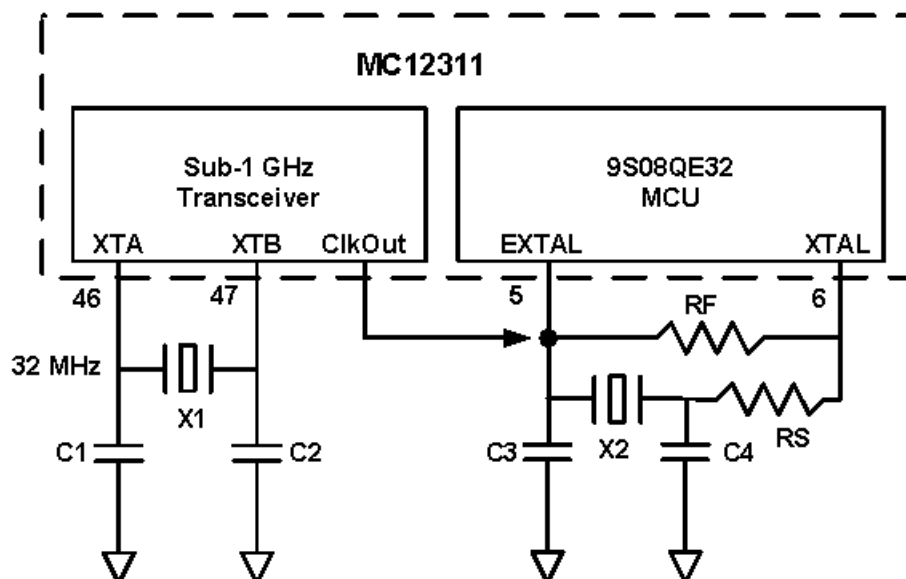


Figure 3-4. MC12311 Clock Connections

3.4.1 Transceiver Oscillator

The transceiver crystal oscillator is the main timing reference of the device. The transceiver oscillator source must always be present and an external crystal is typically used to implement the oscillator, although an external TCXO may also be used (see [Section 4.5.1, “Reference Oscillator”](#)). The source frequency is normally 32 MHz.

In [Figure 3-4](#) crystal X1 and capacitors C1 and C2 form the transceiver crystal oscillator circuit. An off board feedback resistor between input XTA and output XTAL is not required. An important parameter for the crystal X1 is the load capacitance. The oscillator needs to see a balanced load capacitance at each terminal, and as a result, the sum of the stray capacitance of the pcb board, device pin (XTA or XTAL), and load capacitor (C1 or C2) at each terminal should be equal. The amount of external load capacitance is determined by the specific crystal specification.

NOTE

- There is no on board trim capacitance, therefore the user should evaluate and size the external load capacitors to center the oscillator frequency within the cut tolerance of the crystal.
- The frequency accuracy of the crystal (cut tolerance plus temperature variation) must be matched to the required specification of the application.

3.4.1.1 Crystal Resonator Specification

[Table 3-2](#) shows the crystal resonator specification for the crystal reference oscillator circuit of the MC12311 transceiver. This specification covers the full range of operation and is employed in the reference design.

Table 3-2. Crystal Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
FXOSC	XTAL Frequency		26	-	32	MHz
RS	XTAL Serial Resistance		-	30	140	ohms
C0	XTAL Shunt Capacitance		-	2.8	7	pF
CLOAD	External Foot Capacitance	On each pin XTA and XTB	8	16	22	pF

NOTE

- The initial frequency tolerance (cut tolerance), temperature stability and aging performance should be chosen in accordance with the target operating temperature range and the receiver bandwidth selected.
- The loading capacitance should be applied externally, and adapted to the actual C_{load} specification of the crystal.
- A minimum crystal frequency of 28 MHz is required to cover the 863-870 MHz band and 29 MHz for the 902-928 MHz band.

3.4.1.2 Transceiver ClkOut Output (DIO5)

The reference frequency, or a fraction of it, can be provided as an output on DIO5. Use of the ClkOut output allows:

- Driving the clock source to the QE32 MCU -
 - Saves the cost of an additional crystal.
 - CLKOUT can be made available in any operation mode except Sleep mode and is automatically enabled at power-on reset.
- Trimming of the reference oscillator frequency - ClkOut can be provided as a test point for a frequency counter to allow trimming of the external load capacitance during design/evaluation.

The ClkOut functionality is controlled by programming transceiver Register RegDioMapping2 (0x20) (see [Section 6.10, “IRQ and Pin Mapping Registers”](#)):

- Bits 5:4 - control the function of DIO5 and enable ClkOut. See Table 6-2 and Table 6-3.
- Bits 2:0 - control the ClkOut frequency.

Table 3-3 lists ClkOut frequency versus Bits 2:0 setting using a 32 MHz reference source.

Table 3-3. ClkOut Frequency Using 32 MHz Reference Oscillator

ClkOut Bits 2:0	Divide Ratio	ClkOut Frequency (MHz)
000	1	32
001	1/2	16
010	1/4	8
011	1/8	4

Table 3-3. ClkOut Frequency Using 32 MHz Reference Oscillator

ClkOut Bits 2:0	Divide Ratio	ClkOut Frequency (MHz)
100	1/16	2
101	1/32	1
110	RC	62.5 kHz (Automatically enabled)
111	-	OFF (default)

3.4.2 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.

- The ICS module has an on board 32 kHz (nominal) oscillator and FLL -
 - Provides start-up run clock
 - The oscillator can be used with or without the FLL
 - The FLL can be used with an external clock/crystal
- External pins are provided to accommodate an external crystal, resonator, or clock source.
- A separate low power 1 kHz oscillator (LPO) can be used for the real time counter (RTC) or the COP timer.

3.4.2.1 MCU External Clock Source

As shown in [Figure 3-4](#), the external pins associated with the MCU clock source are output XTAL and input EXTAL. An external clock source (such as ClkOut) 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.

3.4.2.2 MCU External Crystal Oscillator

Referring again to [Figure 3-4](#), another choice is the use of an external crystal or ceramic resonator. The MCU oscillator is a Pierce type that can accommodate crystals or resonators in any of three modes:

- Low range (32 -38.4 kHz)
- High range/high gain (1-16 MHz)
- High range/low gain (1-8 MHz)

RS (when used) and RF must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C3 and C4 normally must be high-quality ceramic capacitors specifically designed for high-frequency applications.

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

C3 and C4 are typically in the 5-pF to 25-pF range and are chosen to match the requirements of a specific crystal or resonator. Take into account printed circuit board (PCB) capacitance and MCU pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

When using the oscillator in low range and low gain mode, the external components RS, RF, C3 and C4 are not required.

3.4.2.3 MCU Internal Clock Source

The QE32 MCU has an internal reference clock (nominally 32.768 kHz) that is used in a number of ways:

- Default MCU clock out of reset - the internal ICS clock module defaults to the internal oscillator enabled and the FLL engaged. The resulting default CPU clock frequency is a nominal 8 MHz (4 MHz bus clock).
- Programmable system clock - either used with or without the FLL, the internal oscillator can remain as the normal RUN frequency source
 - Nominal maximum CPU clock of 60 MHz
 - Total trimmed frequency deviation of +/-2% maximum
- Wake-up clock for low power modes - the internal oscillator can be enabled to clock the RTC to provide a wake-up timer from Stop2 or Stop3

3.4.2.4 LPO 1 kHz Oscillator

The LPO is independent of the ICS module and can be used to clock the RTC or COP. Its period can vary greatly from 0.7 - 1.3 mS.

3.4.3 System Clock Configurations

Because of the multiple clock configurations of the MCU, the availability of external clock source pins of both the transceiver and the MCU, and the ClkOut output from the transceiver, there are a number of variations for MC12311 system clock configurations. Key considerations for any system clock configuration are:

- The transceiver 32 MHz source (typically the reference crystal oscillator) must always be present.
- 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 may impact the required system clock rate.

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

NOTE

In the following sub-sections, it is assumed that the MCU GPIO is connected to drive the transceiver reset.

3.4.3.1 Single crystal with ClkOut driving MCU EXTAL input

The single crystal (transceiver) with ClkOut driving the MCU EXTAL input (external clock) is the most common configuration for low cost and excellent frequency accuracy. The ClkOut frequency is programmable from 1 to 32 Mhz and drives the MCU external source.

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 should reset and then release reset to the transceiver (MCU still running on start-up clock)
- Wait for transceiver start-up.
- Program ClkOut to desired frequency
- Wait for the ClkOut source plus FLL to lock, and then switch MCU clock to external source

If the transceiver is forced to a low power condition, the MCU can revert to the internal oscillator and FLL.

3.4.3.2 Single Crystal with MCU Using Internal Clock Only

The single crystal (transceiver) with the MCU using internal clock only has no real advantage over using the ClkOut output, except for slightly lower power.

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 should assert reset and then release reset to the transceiver (MCU still running on start-up clock)
- Program ICS to desired clock rate if the default is not the preferred choice
- Program transceiver to disable ClkOut.

The MCU does not have a high accuracy time base when using the internal reference.

3.4.3.3 Dual Crystal Operation

The transceiver crystal can be augmented by the use of a second crystal on the MCU. The typical application would use a 32.768 kHz 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 should assert reset and then release reset to the transceiver (MCU still running on start-up clock)
- The MCU clock can be switched to the external source after the FLL is locked

- Program transceiver to disable ClkOut

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 time tick for the RTC at very low power.

3.4.4 MCU Background/Mode Select (BKGD/MS)

During a power-on-reset (POR) or background debug force reset (see [Section 10.8.3, “System Background Debug Force Reset Register \(SBD FR\)”](#) for more information), the PTA4/ACMP10/BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin (BKGDPE = 1), an internal pullup device is automatically enabled.

The background debug communication function is enabled when BKGDPE in SOPT1 is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTA4/ACMP10/BKGD/MS pin’s alternative pin functions.

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

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

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

3.5 MC12311 GPIO (Mixed I/O from Transceiver and MCU)

The MC12311 SiP supports a total of 36 GPIO pins that originate from the transceiver and/or the MCU:

- The transceiver provides six pins (DIO5-DIO0) that can be programmed as status -
 - Use of the DIOX are programmed as listed in [Table 6-2](#) and [Table 6-3](#)
 - DIO1 and DIO0 are connected internally to MCU GPIO as well as routed to package pins.
 - Freescale software commonly uses DIO4 externally connected to PTA5/IRQ/TPM1CLK/RESETB
 - DIO5 is also ClkOUT and most commonly externally connected to EXTAL for the MCU clock source
 - DIO3 and DIO2 are uncommitted.
- Four-signal SPI Bus port connected internally and reserved for MCU and transceiver use.
- Twenty-six MAXIMUM additional MCU GPIO are available for use in the SiP package.

NOTE

Four additional MCU GPIO are not pinned-out (see [Figure 1-2](#)) and cannot be used. However, they must be initialized for low power operation.

- If a crystal is used with the MCU, PTB6/XTAL and PTB7/EXTAL pins cannot be used as GPIO. These are available only using the internal clock reference.
- If ClkOut is used to drive the external clock source to the MCU, EXTAL is not available as GPIO
- PTA4/BKGD/MS is not commonly used as GPIO because it is dedicated to the MCU debug port (BDM)

3.5.1 MCU GPIO Characteristics

The internal MCU GPIO hardware consists of 5 ports with 8 signals per port for a total of 40 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, and some pins are internally connected to the transceiver. 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 11, “MCU Parallel Input/Output”](#).
- For information about how and when on-chip peripheral systems use these pins, refer to the appropriate peripheral chapter of this document.
- Not all of the pins are available for external use.

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 11, “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 PTA3–PTA0 and PTD7–PTD0 pins are controlled by a 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.

3.5.2 Transceiver DIOX Characteristics

The transceiver status/GPIO consists of 6 signals total (DIO5-DIO0). DIO2 is bidirectional and all others are output only. Immediately after reset, all the DIO are configured as outputs. Use of the DIO is controlled by transceiver registers RegDioMapping1 and RegDioMapping2, see [Section 6.10, “IRQ and Pin Mapping Registers”](#).

3.6 MC12311 Digital Signal Summary

Table 3-4 summarizes digital I/O pin usage.

Table 3-4. MC12311 Pin Assignment and Pin-Sharing Priority

Pin Number	<-- Lowest Priority --> Highest					Comments
	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4	
17	PTA7	TPM2CH2			ADP9	
18	PTA6	TPM1CH2			ADP8	
33	PTA5	IRQ	TPM1CLK	RESET		
34	PTA4	ACMP1O	BKGD	MS		
24	PTA3	KBI1P3	SCL ¹		ADP3	
25	PTA2	KBI1P2	SDA ¹		ADP2	
26	PTA1	KBI1P1	TPM2CH0	ADP1 ¹	ACMP1- ³	
27	PTA0	KBI1P0	TPM1CH0	ADP0 ³	ACMP1+ ³	
5	PTB7	SCL ²			EXTAL	
6	PTB6	SDA ¹			XTAL	
7	PTB5	TPM1CH1	SS ³			
8	PTB4	TPM2CH1	MISO ²			
13	PTB3	KBI1P7	MOSI ²		ADP7	
14	PTB2	KBI1P6	SPSCK ²		ADP6	
15	PTB1	KBI1P5	TxD1		ADP5	
16	PTB0	KBI1P4	RxD1		ADP4	
28	PTC7	TxD2			ACMP2-	
29	PTC6	RxD2			ACMP2+	
31	PTC5	TPM3CH5			ACMP2O	
32	PTC4	TPM3CH4				
9	PTC3	TPM3CH3				
10	PTC2	TPM3CH2				
11	PTC1	TPM3CH1				

Table 3-4. MC12311 Pin Assignment and Pin-Sharing Priority (continued)

Pin Number	<-- Lowest Priority --> Highest					Comments
	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4	
12	PTC0	TPM3CH0				
-	PTD7	KBI2P7				Not pinned-out
-	PTD6	KBI2P6				
-	PTD5	KBI2P5				
21	PTD4	KBI2P4				
22	PTD3	KBI2P3				
23	PTD2	KBI2P2				
50	PTD1	KBI2P1				Shared w/DIO0
49	PTD0	KBI2P0				Sharedw/DIO1
-	PTE7	TPM3CLK				Not pinned-out
-	PTE6					
-	PTE5					
-	PTE4					
59	PTE3	\overline{SS}^2				SPI Port signal
60	PTE2	MISO ²				SPI Port signal
58	PTE1	MOSI ²				SPI Port signal
57	PTE0	TPM2CLK	SPSCK ²			SPI Port signal
54	DIO5					Used as ClkOut
53	DIO4					
52	DIO3					
51	DIO2					
49	DIO1					Shared w/PTD0
50	DIO0					Shared w/PTD1

¹ If ADC and ACMP1 are enabled, both modules have access to the pin.

² IIC pins, SCL and SDA, can be repositioned using IICPS in SOPT2; default reset locations are PTA3 and PTA2.

³ SPI pins (SS, MISO, MOSI, and SPSCK) can be repositioned using SPIPS in SOPT2. Default locations are PTB5, PTB4, PTB3, and PTB2.

3.7 Transceiver RF Configurations and External Connections

The MC12311 transceiver radio has features that allow for a flexible as well as low cost RF interface:

- Sensitivity down to -120 dBm at 1.2 kbps
- High selectivity w/16-tap FIR channel filter
- Robust receiver front end:
 - IIP3 = -18 dBm
 - IIP2 = +35 dBm
 - 80 dB blocking immunity
 - No image frequency response
- Programmable Pout from -18 to +17 dBm in 1 dB steps
- FSK bit rates up to 300 kbps
- FSK, GFSK, MSK, GMSK, and OOK modulators
- Two TX output power configurations

3.7.1 RF Interface Pins

The MC12311 transceiver has the following pins associated with the RF interface:

- VR_PA - regulated output voltage to be used with TX PA
- PA_BOOST - optional secondary high-power TX PA
- RFIO - RF bidirectional input/output; used as input only for PA boost mode
- RXTX - digital output to control RF switch
- GND_PA1- RF ground
- GND_PA2 - RF ground

NOTE

The following descriptions for RF operation are not meant as complete applications information, but rather general information. Freescale supplies evaluation boards as well as complete reference designs, and the user is directed to these designs upon which to build target systems.

There are two basic RF configurations as described in the following sections.

3.7.2 Standard Output Power RF Configuration (Single, Bidirectional Port)

The standard RF configuration for the transceiver is a single, bidirectional port mode (shown in [Figure 3-5](#)):

- Only the bidirectional RF pin RFIO is used
- Maximum output power of typically +13 dBm into a 50 ohm load.
- The output power is programmable in approximately 1 dB steps.

- Inductor L8 and capacitor C19 provide an ac blocking network for the PA power supply.
- The remaining external components are a generalized RF network -
 - Provides impedance match between the antenna and the transceiver RFIO
 - Supports an elliptic-function low pass filter to provide TX harmonic trapping and out-of-band suppression
 - The topology for the RF matching network can be used over the various bands of interest with changes in component values. Not all indicated components are used at all frequencies

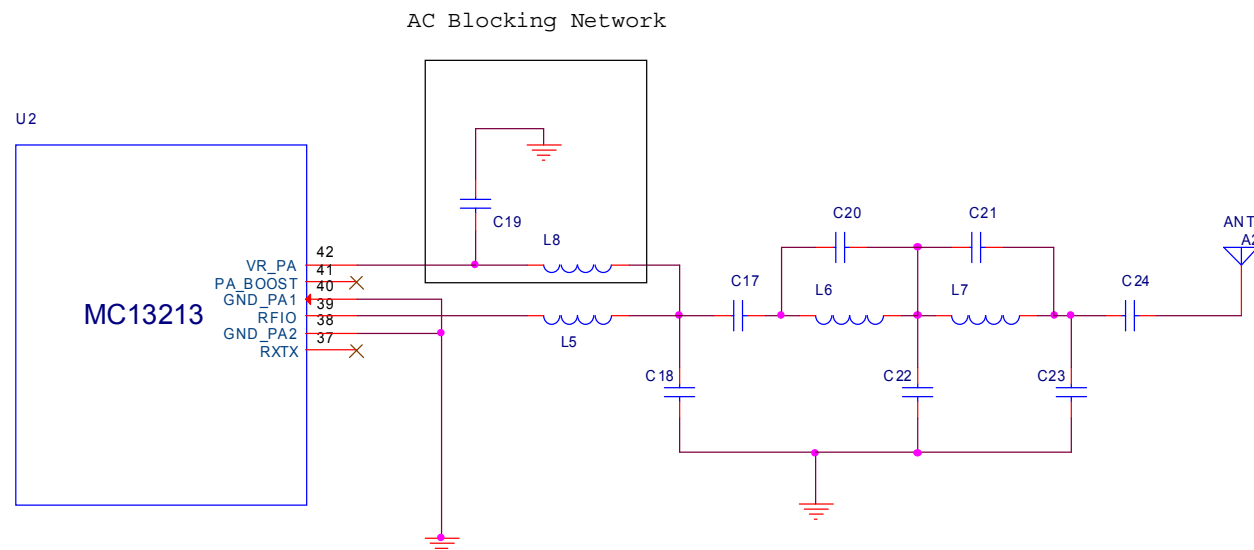


Figure 3-5. Single Port RF Schematic (+13dBm)

3.7.3 Higher Output Power RF Configuration (Dual Port with Optional External Power Amplifier)

The secondary RF configuration for the transceiver is a dual port mode (shown in [Figure 3-6](#)):

- Secondary PA output (PA_BOOST) is used as the TX port and the standard RFIO pin is used as the RX port.
- Maximum output power from the PA_BOOST output is typically +17 dBm into a 50 ohm load.
- Inductor L1 and capacitor C1 provide an ac blocking network for the PA_BOOST power supply.
- An external RF switch is required to enable the appropriate path to the antenna -
 - The TX port (PA_BOOST network) is selected for transmit
 - The RFIO port network is selected for receive
 - Device control output RXTX can be used to control the antenna switch
- An optional external power amplifier can be inserted between the device TX output and the antenna switch to increase transmitted power up to typically about 1 W maximum.

- Similar to the single-port configuration, generalized RF circuit topologies are shown for both the TX and RX paths -
 - The TX path network provides both impedance matching and low pass filtering for TX harmonic trapping and out-of-band suppression
 - The RX path network typically provides only impedance match between the antenna and the transceiver RFIO and can normally be greatly simplified
 - The topology for the RF matching networks can be used over the various bands of interest with changes in component values. Not all indicated components are used at all frequencies.

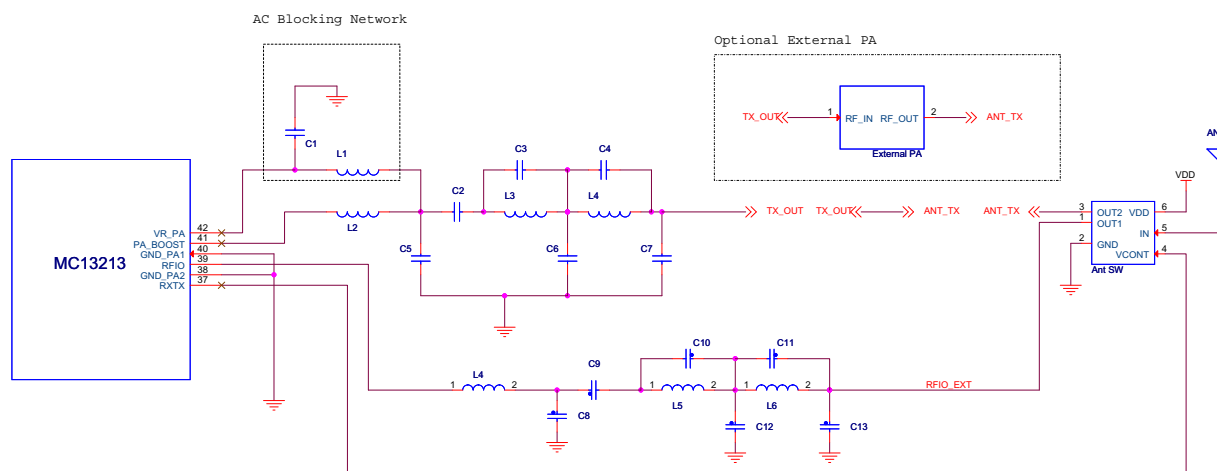


Figure 3-6. Dual Port RF Schematic (+17dBm to 1W)

3.7.4 Filter and Matching Network Component Values

The generalized filter/matching network shown in [Figure 3-5](#) and [Figure 3-6](#) must be evaluated and tuned to its application use and frequency:

- Impedance matching is always required
- Only a transmission path typically uses any low pass filtering elements
- Not all indicated components are used at all frequencies

To provide an initial design configuration for several popular frequency bands, [Table 3-5](#) lists component values versus frequency and use for the generalized component topology of [Figure 3-7](#). For each frequency, the filter components for the PA_BOOST output filter (Dual Port TX) and the Single Port (TX/RX) are given.

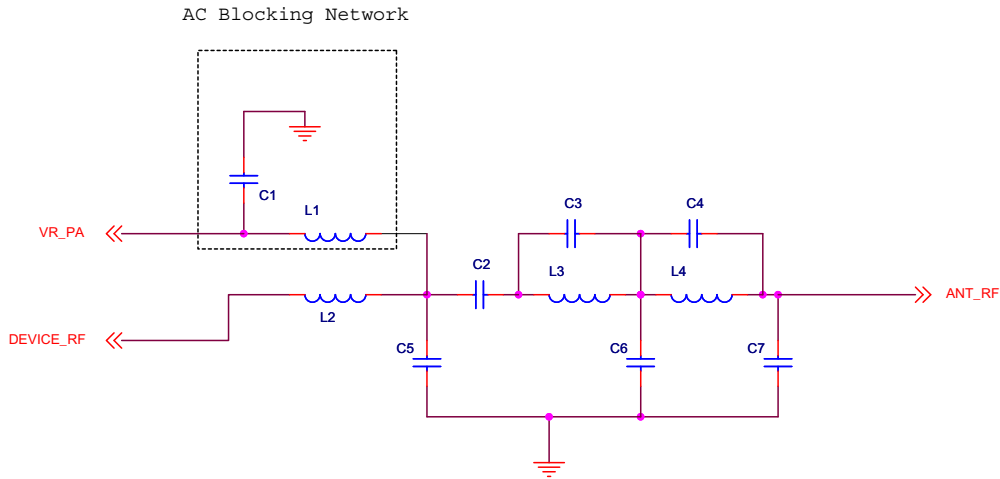


Figure 3-7. General RF Filter/Matching Network Topology

NOTE

These component values are given only as suggested initial design values. The user must evaluate his particular design and adjust/change components as required to meet targeted specifications.

Table 3-5. RF Component Values vs. Frequency Band

Component Designation	434 MHz		868 MHz or 915 MHz	
	Dual Port TX	Single Port TX/RX	Dual Port TX	Single Port TX/RX
L1	22nH	33nH	33nH	33nH
L2	Short	10nH	Short	3.9nH
L3	12nH	12nH	5.6nH	6.8nH
L4	15nH	10nH	5.6nH	6.8nH
C2	33pF	22pF	47pF	8.2pF
C3	2.7pF	2.7pF	10pF	10pF
C4	10pF	10pF	10pF	10pF
C5	18pF	15pF	10pF	4.7pF
C6	15pF	15pF	6.8pF	6.8pF
C7	8.2 pF	8.2pF	3.3pF	5.6pF



Chapter 4

Sub 1-GHz Transceiver Architecture Description

This chapter describes the architecture and operation of the MC12311 low-power, highly integrated transceiver chip.

4.1 Overview

The MC12311 transceiver is a single-chip integrated circuit ideally suited for today's high performance ISM band RF applications. The transceiver's advanced features set, including state of the art packet engine greatly simplifies system design while the high level of integration reduces the external BOM to a handful of passive decoupling and matching components. It is intended for use as high-performance, low-cost FSK and OOK RF transceiver for robust frequency agile, half-duplex bidirectional RF links, and where stable and constant RF performance is required over the full operating voltage range of the device.

The transceiver is intended for applications over a wide frequency range, including the 433 MHz and 868 MHz European and the 902-928 MHz North American ISM bands. Coupled with a link budget in excess of 135 dB, the advanced system features include a 66-byte TX/RX FIFO, configurable automatic packet handler, listen mode, temperature sensor and configurable DIOs which greatly enhance system flexibility whilst at the same time significantly reducing MCU requirements.

The transceiver complies with both ETSI and FCC regulatory requirements

4.2 Simplified Block Diagram

Figure 4-1 shows a simplified block diagram of the MC12311 transceiver.

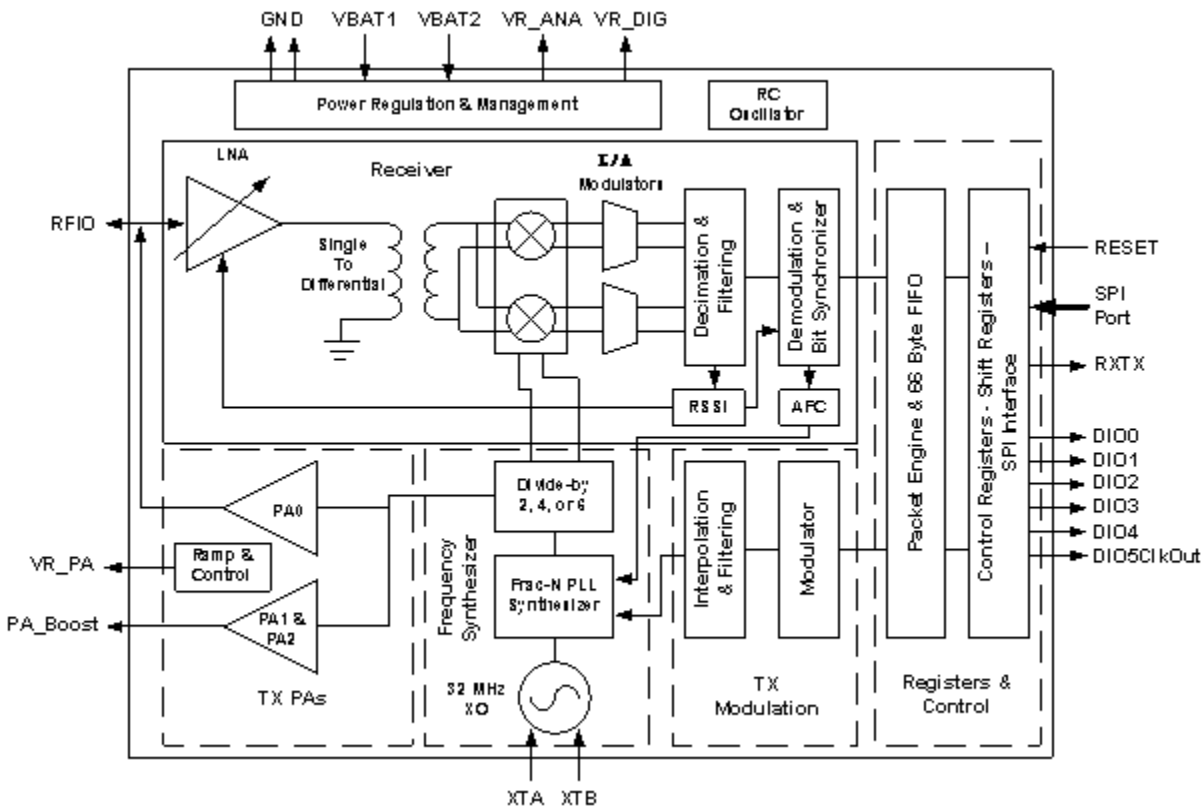


Figure 4-1. MC12311 Transceiver Block Diagram

4.3 Transceiver Power Supply

The MC12311 has separate power pins for both the MCU and the transceiver. The transceiver employs on board regulation and management that provides stable operating characteristics over the full temperature and voltage range of operation. This includes the full output power of +17dBm which is maintained from 1.8 to 3.6 V.

The transceiver voltage source is supplied via pins VBAT1 and VBAT2. (see [Section 3.2, “Power Connections”](#)) for power supply connection and bypassing details.

4.4 Low Battery Detector

A low battery detector is also included allowing the generation of an interrupt signal in response to passing a programmable threshold adjustable through the register *RegLowBat*. The interrupt signal can be mapped to any of the DIO pins, through the programming of *RegDioMapping*.

4.5 Frequency Synthesis

The LO generation on the MC12311 transceiver is based on a state-of-the-art fractional-N PLL. The PLL is fully integrated with automatic calibration.

4.5.1 Reference Oscillator

The crystal oscillator is the main timing reference of the MC12311. It is used as a reference for the transceiver frequency synthesizer and as a clock for the digital processing. In turn, the transceiver ClkOut signal can be used to provide an external reference clock for the MCU (see [Section 3.4, “System Clock Sources and Configurations”](#)).

The XO startup time, TS_OSC , depends on the actual XTAL being connected on pins XTA and XTB. When using the built-in sequencer, the device optimizes the startup time and automatically triggers the PLL when the XO signal is stable. To manually control the startup time, the user should either wait for TS_OSC max, or monitor the signal CLKOUT which will only be made available on the output buffer when a stable XO oscillation is achieved.

An external clock can be used to replace the crystal oscillator, for instance a tight tolerance TCXO. To do so, Bit 4 at transceiver Address 0x59 should be set to 1, and the external clock has to be provided on XTA (Pin 4). XTB (pin 5) should be left open. The peak-peak amplitude of the input signal must never exceed 1.8 V. Please consult your TCXO supplier for an appropriate value of decoupling capacitor, C_D .

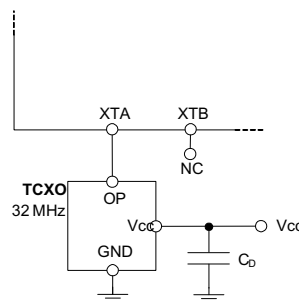


Figure 4-2. TCXO Connection

4.5.2 CLKOUT Output

The reference frequency, or a fraction of it, can be provided on DIO5 (pin 12) by modifying bits *ClkOut* in *RegDioMapping2*. Two typical applications of the CLKOUT output include:

- Provide a clock output for the MCU (see [Section 3.4, “System Clock Sources and Configurations”](#)) - saves the cost of an additional crystal. CLKOUT can be made available in any operation mode except Sleep mode and is automatically enabled at power-on reset.
- Provides an oscillator reference output - allows simple software trimming of the initial crystal tolerance.

If the transceiver is put into low power mode, ClkOut may be disabled for lowest power.

4.5.3 PLL Architecture

The frequency synthesizer generating the LO frequency for both the receiver and the transmitter is a fractional-N sigma-delta PLL. The PLL incorporates a third order loop capable of fast auto-calibration, and it has a fast switching-time. The VCO and the loop filter are both fully integrated, removing the need for an external tight-tolerance, high-Q inductor in the VCO tank circuit.

4.5.3.1 VCO

The VCO runs at 2, 4 or 6 times the RF frequency (respectively in the 915, 434 and 315 MHz bands) to reduce any LO leakage in receiver mode, to improve the quadrature precision of the receiver, and to reduce the pulling effects on the VCO during transmission.

The VCO calibration is fully automated. A coarse adjustment is carried out at power-on reset, and fine tuning is performed each time the transceiver PLL is activated. Automatic calibration times are fully transparent to the end-user, as this processing time is included in the *TS_TE* and *TS_RE* specifications.

4.5.3.2 PLL Bandwidth

The bandwidth of the Fractional-N PLL is wide enough to allow for:

- High speed FSK modulation, up to 300 kb/s, inside the PLL bandwidth
- Very fast PLL lock times - enabling both short startup and fast hop times required for frequency agile applications

4.5.3.3 Carrier Frequency and Resolution

The transceiver PLL embeds a 19-bit sigma-delta modulator and its frequency resolution, constant over the whole frequency range, and is given by:

$$F_{\text{STEP}} = \frac{F_{\text{XOSC}}}{2^{19}}$$

The carrier frequency is programmed through *RegFrf*, split across Addresses 0x07 to 0x09:

$$F_{\text{RF}} = F_{\text{STEP}} \times \text{Frf}(23,0)$$

NOTE

The Frf setting is split across 3 bytes. A change in the center frequency will only be taken into account when the least significant byte *FrfLsb* in *RegFrfLsb* is written. This allows for more complex modulation schemes such as m-ary FSK, where frequency modulation is achieved by changing the programmed RF frequency.

4.5.4 Lock Time

PLL lock time *TS_FS* is a function of a number of technical factors, such as synthesized frequency, frequency step, etc. When using the built-in sequencer, the transceiver optimizes the startup time and automatically starts the receiver or the transmitter when the PLL has locked. To manually control the startup time, the user should either wait for *TS_FS* max given in the specification, or monitor the signal PLL lock detect indicator, which is set when the PLL has is within its locking range.

When performing an AFC, which usually corrects very small frequency errors, the PLL response time is approximately:

$$T_{\text{PLL AFC}} = \frac{5}{\text{PLL BW}}$$

In a frequency hopping scheme, the timings *TS_HOP* given in the table of specifications give an order of magnitude for the expected lock times.

4.5.5 Lock Detect Indicator

A lock indication signal can be made available on some of the DIO pins, and is toggled high when the PLL reaches its locking range.

4.6 Transmitter Description

The transmitter of MC12311 transceiver is comprised of the frequency synthesizer, modulator and power amplifier blocks.

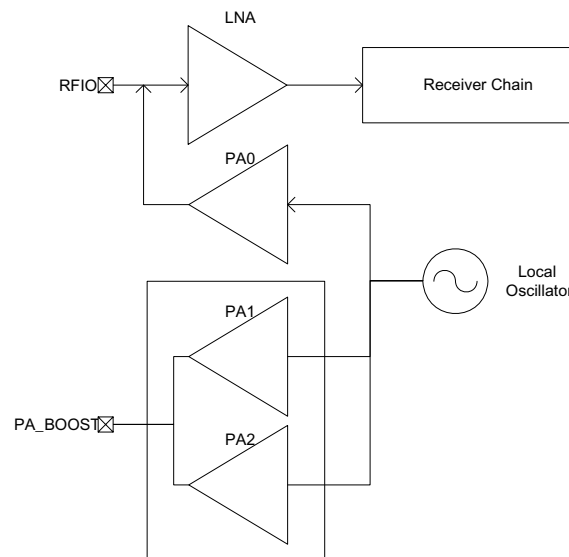


Figure 4-3. Transmitter Block Diagram

4.6.1 Bit Rate Setting

When using the transceiver in Continuous mode, the data stream to be transmitted can be input directly to the modulator via pin DIO2/DATA in an asynchronous manner, unless Gaussian filtering is used, in which case the DCLK signal on pin 10 (DIO1/DCLK) is used to synchronize the data stream.

In Packet mode or in Continuous mode with Gaussian filtering enabled, the Bit Rate (BR) is controlled by bits *BitRate* in *RegBitrate*:

$$\text{BR} = \frac{F_{\text{XOSC}}}{\text{BitRate}}$$

Among others, the following Bit Rates are accessible:

Table 4-1. Bit Rate Examples

Type	BitRate (15:8)	BitRate (7:0)	(G)FSK (G)MSK	OOK	Actual BR (b/s)
Classical transceiver baud rates (multiples of 1.2 kbps)	0x68	0x2B	1.2 kbps	1.2 kbps	1200.015
	0x34	0x15	2.4 kbps	2.4 kbps	2400.060
	0x1A	0x0B	4.8 kbps	4.8 kbps	4799.760
	0x0D	0x05	9.6 kbps	9.6 kbps	9600.960
	0x06	0x83	19.2 kbps	19.2 kbps	19196.16
	0x03	0x41	38.4 kbps		38415.36
	0x01	0xA1	76.8 kbps		76738.60
	0x00	0xD0	153.6 kbps		153846.1
Classical transceiver baud rates (multiples of 0.9 kbps)	0x02	0x2C	57.6 kbps		57553.95
	0x01	0x16	115.2 kbps		115107.9
Round bit rates (multiples of 12.5, 25 and 50 kbps)	0x0A	0x00	12.5 kbps	12.5 kbps	12500.00
	0x05	0x00	25 kbps	25 kbps	25000.00
	0x02	0x80	50 kbps		50000.00
	0x01	0x40	100 kbps		100000.0
	0x00	0xD5	150 kbps		150234.7
	0x00	0xA0	200 kbps		200000.0
	0x00	0x80	250 kbps		250000.0
	0x00	0x6B	300 kbps		299065.4
Watch Xtal frequency	0x03	0xD1	32.768 kbps	32.768 kbps	32753.32

4.6.2 FSK Modulation

FSK modulation is performed inside the PLL bandwidth, by changing the fractional divider ratio in the feedback loop of the PLL. The large resolution of the sigma-delta modulator, allows for very narrow frequency deviation. The frequency deviation F_{DEV} is given by:

$$F_{DEV} = F_{STEP} \times Fdev(13,0)$$

To ensure a proper modulation, the following limit applies:

$$F_{DEV} + \frac{BR}{2} \leq 500\text{kHz}$$

NOTE

No constraint applies to the modulation index of the transmitter, but the frequency deviation F_{DEV} must exceed 600 Hz.

4.6.3 OOK Modulation

OOK modulation is applied by switching on and off the Power Amplifier. Digital control and smoothing are available to improve the transient power response of the OOK transmitter.

4.6.4 Modulation Shaping

Modulation shaping can be applied in both OOK and FSK modulation modes, to improve the narrowband response of the transmitter. Both shaping features are controlled with *PaRamp* bits in *RegPaRamp*.

- In FSK mode, a Gaussian filter with $BT = 0.3, 0.5$ or 1 is used to filter the modulation stream, at the input of the sigma-delta modulator. If the Gaussian filter is enabled when the MC12311 is in Continuous mode, DCLK signal on pin 10 (DIO1/DCLK) will trigger an interrupt on the uC each time a new bit has to be transmitted.
- When OOK modulation is used, the PA bias voltages are ramped up and down smoothly when the PA is turned on and off, to reduce spectral splatter.

NOTE

The transmitter must be restarted if the ModulationShaping setting is changed, in order to recalibrate the built-in filter.

4.6.5 Power Amplifiers

Three power amplifier blocks are embedded in the transmitter. The first one (PA0) can generate up to +13 dBm into a 50 Ohm load. PA0 shares a common front-end pin RFIO (pin 21) with the receiver LNA.

PA1 and PA2 are both connected to pin PA_BOOST (pin 23), allowing for two distinct power ranges:

- Low power mode - where power out is $-18 \text{ dBm} < P_{out} < 13 \text{ dBm}$, with PA1 enabled
- Higher power mode - when PA1 and PA2 are combined, providing up to +17 dBm to a matched load.

NOTE

When PA1 and PA2 are combined to deliver +17 dBm to the antenna, a specific impedance matching / harmonic filtering design is required to ensure impedance transformation and regulatory compliance.

All PA settings are controlled by *RegPaLevel*, and the truth table of settings is given in [Table 4-2](#).

Table 4-2. Power Amplifier Mode Selection Truth Table

Pa0On	Pa1On	Pa2On	Mode	Power Range	Pout Formula
1	0	0	PA0 output on pin RFIO	-18 to +13 dBm	-18 dBm + <i>OutputPower</i>
0	1	0	PA1 enabled on pin PA_BOOST	-18 to +13 dBm	-18 dBm + <i>OutputPower</i>

Table 4-2. Power Amplifier Mode Selection Truth Table

Pa0On	Pa1On	Pa2On	Mode	Power Range	Pout Formula
0	1	1	PA1 and PA2 combined on pin PA_BOOST	-14 to +17 dBm	-14 dBm + <i>OutputPower</i>
Other combinations			Reserved		

NOTE

- To ensure correct operation at the highest power levels, adjust the Over Current Protection Limit accordingly in *RegOcp*.
- If PA_BOOST pin is not used, the pin can be left floating.

4.6.6 Over Current Protection

An over-current protection block is built-in the chip that helps prevent surge currents when the transmitter is used at its highest power levels, thus protecting the battery that may power the application. The current clamping value is controlled by *OcpTrim* bits in *RegOcp*, and is calculated with the following formula:

$$I_{max} = 45 + 5 \times OcpTrim(mA)$$

NOTE

I_{max} sets the maximum current drawn by the final PA stage, and does not account for the PA drivers and frequency synthesizer. Global current drain on *V_{batt}* will be higher

4.7 Receiver Description

The MC12311 transceiver features a digital receiver with the analog to digital conversion process being performed directly following the LNA-Mixers block. The zero-IF receiver is able to handle (G)FSK and (G)MSK modulation. ASK and OOK modulation is, however, demodulated by a low-IF architecture. All the filtering, demodulation, gain control, synchronization and packet handling is performed digitally, and this allows a very wide range of bit rates and frequency deviations to be selected. The receiver is also capable of automatic gain calibration in order to improve precision of RSSI measurements.

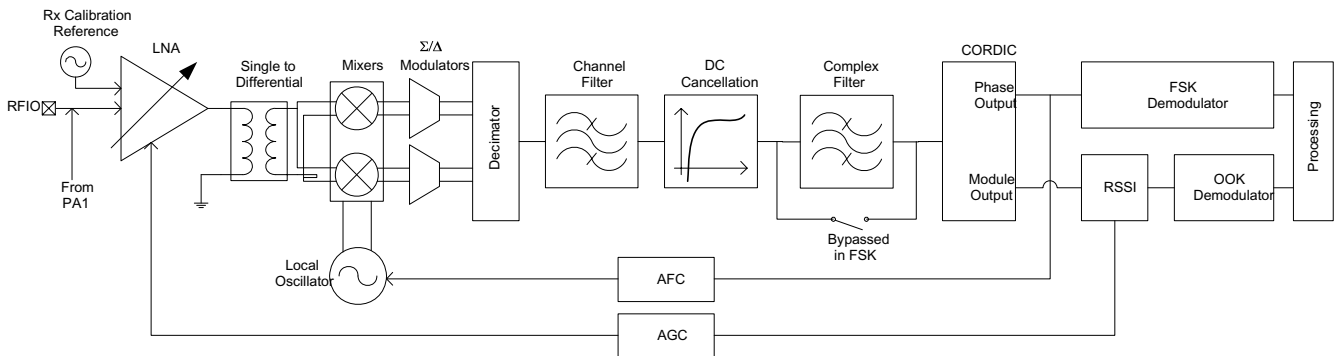


Figure 4-4. Receiver Block Diagram

The following sections give a brief description of each of the receiver blocks.

4.7.1 LNA - Single to Differential Buffer

The LNA uses a common-gate topology, which allows for a flat characteristic over the whole frequency range. It is designed to have an input impedance of 50 Ohms or 200 Ohms (as selected with bit *LnaZin* in *RegLna*), and the parasitic capacitance at the LNA input port is cancelled with an external RF choke. A single to differential buffer is implemented to improve the second order linearity of the receiver.

The LNA gain, including the single-to-differential buffer, is programmable over a 48 dB dynamic range, and control is either manual or automatic with the embedded AGC function.

NOTE

In the specific case where the LNA gain is manually set by the user, the receiver will not be able to properly handle FSK signals with a modulation index smaller than 2 at an input power greater than the 1dB compression point, [tabulated in section 4.7.2](#).

Table 4-3. LNA Gain Settings

LnaGainSelect	LNA Gain	Gain Setting
000	Any of the below, set by the AGC loop	-
001	Max gain	G1
010	Max gain - 6 dB	G2
011	Max gain - 12 dB	G3
100	Max gain - 24 dB	G4
101	Max gain - 36 dB	G5
110	Max gain - 48 dB	G6
111	Reserved	-

4.7.2 Automatic Gain Control

By default (*LnaGainSelect* = 000), the LNA gain is controlled by a digital AGC loop in order to obtain the optimal sensitivity/linearity trade-off.

Regardless of the data transfer mode (Packet or Continuous), the following series of events takes place when the receiver is enabled:

- The receiver stays in WAIT mode, until *RssiValue* exceeds *RssiThreshold* for two consecutive samples. Its power consumption is the receiver power consumption.
- When this condition is satisfied, the receiver automatically selects the most suitable LNA gain, optimizing the sensitivity/linearity trade-off.
- The programmed LNA gain, read-accessible with *LnaCurrentGain* in *RegLna*, is carried on for the whole duration of the packet, until one of the following conditions is fulfilled:

- **Packet mode:** if *AutoRxRestartOn* = 0, the LNA gain will remain the same for the reception of the following packet. If *AutoRxRestartOn* = 1, after the controller has emptied the FIFO the receiver will re-enter the WAIT mode described above, after a delay of *InterPacketRxDelay*, allowing for the distant transmitter to ramp down, hence avoiding a false RSSI detection. In both cases (*AutoRxRestartOn*=0 or *AutoRxRestartOn*=1), the receiver can also re-enter the WAIT mode by setting *RestartRx* bit to 1. The user can decide to do so, to manually launch a new AGC procedure.
- **Continuous mode:** upon reception of valid data, the user can decide to either leave the receiver enabled with the same LNA gain, or to restart the procedure, by setting *RestartRx* bit to 1, resuming the WAIT mode of the receiver, described above.

NOTE

- The AGC procedure must be performed while receiving preamble in FSK mode.
- In OOK mode, the AGC will give better results if performed while receiving a constant “1” sequence

The following figure illustrates the AGC behavior:

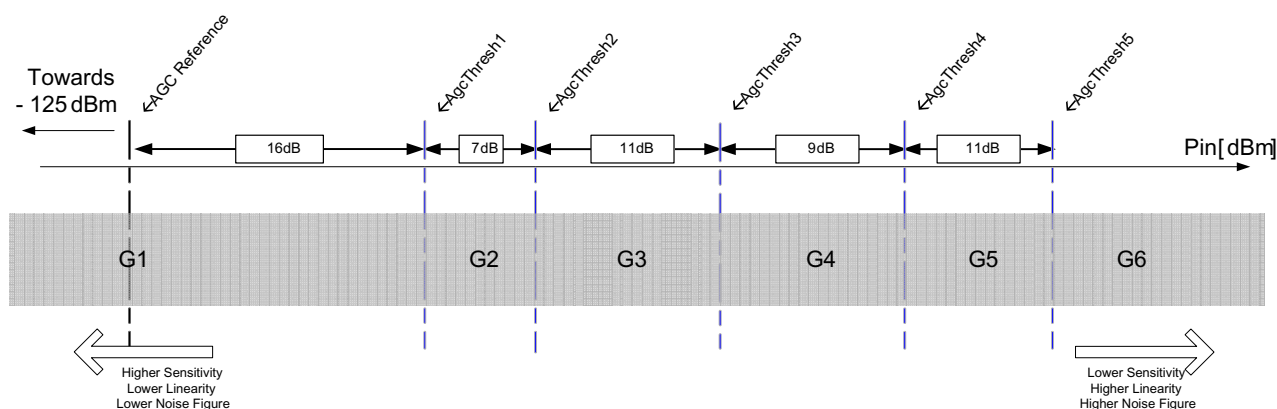


Figure 4-5. AGC Thresholds Settings

The following table summarizes the typical performance of the complete receiver:

Table 4-4. Receiver Performance Summary

Input Power Pin	Gain Setting	Receiver Performance (typ)			
		P _{-1dB} [dBm]	NF [dB]	IIP3 [dBm]	IIP2 [dBm]
Pin < AgcThresh1	G1	-37	7	-18	+35
AgcThresh1 < Pin < AgcThresh2	G2	-31	13	-15	+40
AgcThresh2 < Pin < AgcThresh3	G3	-26	18	-8	+48
AgcThresh3 < Pin < AgcThresh4	G4	-14	27	-1	+62

Table 4-4. Receiver Performance Summary

AgcThresh4 < Pin < AgcThresh5	G5	>-6	36	+13	+68
AgcThresh5 < Pin	G6	>0	44	+20	+75

4.7.2.1 RssiThreshold Setting

For correct operation of the AGC, *RssiThreshold* in *RegRssiThresh* must be set to the sensitivity of the receiver. The receiver will remain in WAIT mode until *RssiThreshold* is exceeded.

NOTE

When AFC is enabled and performed automatically at the receiver startup, the channel filter is used by the receiver during the AFC and the AGC is *RxBwAfc* instead of the standard *RxBw* setting. This may impact the sensitivity of the receiver, and the setting of *RssiThreshold* accordingly.

4.7.2.2 AGC Reference

The AGC reference level is automatically computed in the transceiver according to:

$$\text{AGC Reference [dBm]} = -174 + \text{NF} + \text{DemodSnr} + 10 \cdot \log(2 \cdot \text{RxBw}) + \text{FadingMargin} \quad [\text{dBm}]$$

With:

- $\text{NF} = 7\text{dB}$: LNA's Noise Figure at maximum gain
- $\text{DemodSnr} = 8\text{ dB}$: SNR needed by the demodulator
- RxBw : Single sideband channel filter bandwidth
- $\text{FadingMargin} = 5\text{ dB}$: Fading margin

4.7.3 Continuous-Time DAGC

In addition to the automatic gain control described in [Section 4.7.2, “Automatic Gain Control”](#), the transceiver is capable of continuously adjusting its gain in the digital domain, after the analog to digital conversion has occurred. This feature, named DAGC, is fully transparent to the end user. The digital gain adjustment is repeated every 2 bits, and has the following benefits:

- Fully transparent to the end user
- Improves the fading margin of the receiver during the reception of a packet, even if the gain of the LNA is frozen
- Improves the receiver robustness in fast fading signal conditions, by quickly adjusting the receiver gain (every 2 bits)
- Works in Continuous, Packet, and unlimited length Packet modes

The DAGC is enabled by setting *RegTestDagc* to 0x10 for low modulation index systems (i.e. when *AfcLowBetaOn=1*) and 0x30 for other systems. It is recommended to always enable the DAGC.

4.7.4 Quadrature Mixer - ADCs - Decimators

The mixer is inserted between output of the RF buffer stage and the input of the analog to digital converter (ADC) of the receiver section. This block is designed to translate the spectrum of the input RF signal to baseband, and offer both high IIP2 and IIP3 responses.

In the lower bands of operation (290 to 510 MHz), the multi-phase mixing architecture with weighted phases improves the rejection of the LO harmonics in receiver mode, hence increasing the receiver immunity to out-of-band interferers.

The I and Q digitalization is made by two 5th order continuous-time Sigma-Delta Analog to Digital Converters (ADC). Their gain is not constant over temperature, but the whole receiver is calibrated before reception, so that this inaccuracy has no impact on the RSSI precision. The ADC output is one bit per channel. It needs to be decimated and filtered afterwards. This ADC can also be used for temperature measurement; please refer to [Section 4.7.16, “Temperature Sensor”](#) for more details.

The decimators decrease the sample rate of the incoming signal in order to optimize the area and power consumption of the following receiver blocks.

4.7.5 Channel Filter

The role of the channel filter is to filter out the noise and interferers outside of the channel. Channel filtering on the transceiver is implemented with a 16-tap Finite Impulse Response (FIR) filter, providing an outstanding Adjacent Channel Rejection performance, even for narrowband applications.

NOTE

To respect oversampling rules in the decimation chain of the receiver, the Bit Rate cannot be set to a value higher than 2 times the single-side receiver bandwidth ($\text{BitRate} < 2 \times \text{RxBw}$)

The single-side channel filter bandwidth RxBw is controlled by the parameters RxBwMant and RxBwExp in RegRxBw:

- When FSK modulation is enabled:

$$\text{RxBw} = \frac{\text{FXOSC}}{\text{RxBwMant} \times 2^{\text{RxBwExp} + 2}}$$

- When OOK modulation is enabled:

$$\text{RxBw} = \frac{\text{FXOSC}}{\text{RxBwMant} \times 2^{\text{RxBwExp} + 3}}$$

NOTE

The following channel filter bandwidths are accessible (oscillator is mandated at 32 MHz):

Table 4-5. Available RxBw Settings

RxBwMant (binary/value)	RxBwExp (decimal)	RxBw (kHz)	
		FSK ModulationType=00	OOK ModulationType=01
10b / 24	7	2.6	1.3
01b / 20	7	3.1	1.6
00b / 16	7	3.9	2.0
10b / 24	6	5.2	2.6
01b / 20	6	6.3	3.1
00b / 16	6	7.8	3.9
10b / 24	5	10.4	5.2
01b / 20	5	12.5	6.3
00b / 16	5	15.6	7.8
10b / 24	4	20.8	10.4
01b / 20	4	25.0	12.5
00b / 16	4	31.3	15.6
10b / 24	3	41.7	20.8
01b / 20	3	50.0	25.0
00b / 16	3	62.5	31.3
10b / 24	2	83.3	41.7
01b / 20	2	100.0	50.0
00b / 16	2	125.0	62.5
10b / 24	1	166.7	83.3
01b / 20	1	200.0	100.0
00b / 16	1	250.0	125.0
10b / 24	0	333.3	166.7
01b / 20	0	400.0	200.0
00b / 16	0	500.0	250.0

4.7.6 DC Cancellation

DC cancellation is required in zero-IF architecture transceivers to remove any DC offset generated through self-reception. It is built into the device and its adjustable cutoff frequency f_c is controlled in $RegRxBw$:

$$f_c = \frac{4 \times RxBw}{2\pi \times 2^{DccFreq + 2}}$$

The default value of *DccFreq* cutoff frequency is typically 4% of the *RxBw* (channel filter BW). The cutoff frequency of the DCC can however be increased to slightly improve the sensitivity under wider modulation conditions. It is advised to adjust the DCC setting while monitoring the receiver sensitivity.

4.7.7 Complex Filter - OOK

In OOK mode the receiver is modified to a low-IF architecture. The IF frequency is automatically set to half the single side bandwidth of the channel filter ($F_{IF} = 0.5 \times RxBw$). The Local Oscillator is automatically offset by the IF in the OOK receiver. A complex filter is implemented on the chip to attenuate the resulting image frequency by typically 30 dB.

NOTE

This filter is automatically bypassed when receiving FSK signals (ModulationType = 00 in RegDataModul).

4.7.8 RSSI

The RSSI block evaluates the amount of energy available within the receiver channel bandwidth. Its resolution is 0.5 dB, and it has a wide dynamic range to accommodate both small and large signal levels that may be present. Its acquisition time is very short, taking only 2 bit periods. The RSSI sampling must occur during the reception of preamble in FSK, and constant “1” reception in OOK.

NOTE

- The receiver is capable of automatic gain calibration in order to improve the precision of its RSSI measurements. This function injects a known RF signal at the LNA input, and calibrates the receiver gain accordingly. This calibration is automatically performed during the PLL start-up, making it a transparent process to the end-user
- *RssiValue* can only be read when it exceeds *RssiThreshold*

4.7.9 Cordic

The Cordic task is to extract the phase and the amplitude of the modulation vector (I+j.Q). This information, still in the digital domain is used as follows:

- Phase output - used by the FSK demodulator and the AFC blocks.
- Amplitude output - used by the RSSI block, for FSK demodulation, AGC and automatic gain calibration purposes.

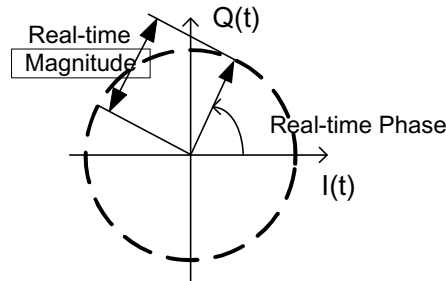


Figure 4-6. Cordic Extraction

4.7.10 FSK Demodulator

The FSK demodulator of the receiver is designed to demodulate FSK, GFSK, MSK and GMSK modulated signals. It is most efficient when the modulation index of the signal is greater than 0.5 and below 10:

$$0.5 \leq \beta = \frac{2 \times F_{DEV}}{BR} \leq 10$$

The output of the FSK demodulator can be fed to the Bit Synchronizer (described in Section 4.7.12), to provide the companion processor with a synchronous data stream in Continuous mode.

4.7.11 OOK Demodulator

The OOK demodulator performs a comparison of the RSSI output and a threshold value. Three different threshold modes are available, configured through bits *OokThreshType* in *RegOokPeak*.

The recommended mode of operation is the "Peak" threshold mode, illustrated in [Figure 4-7](#):

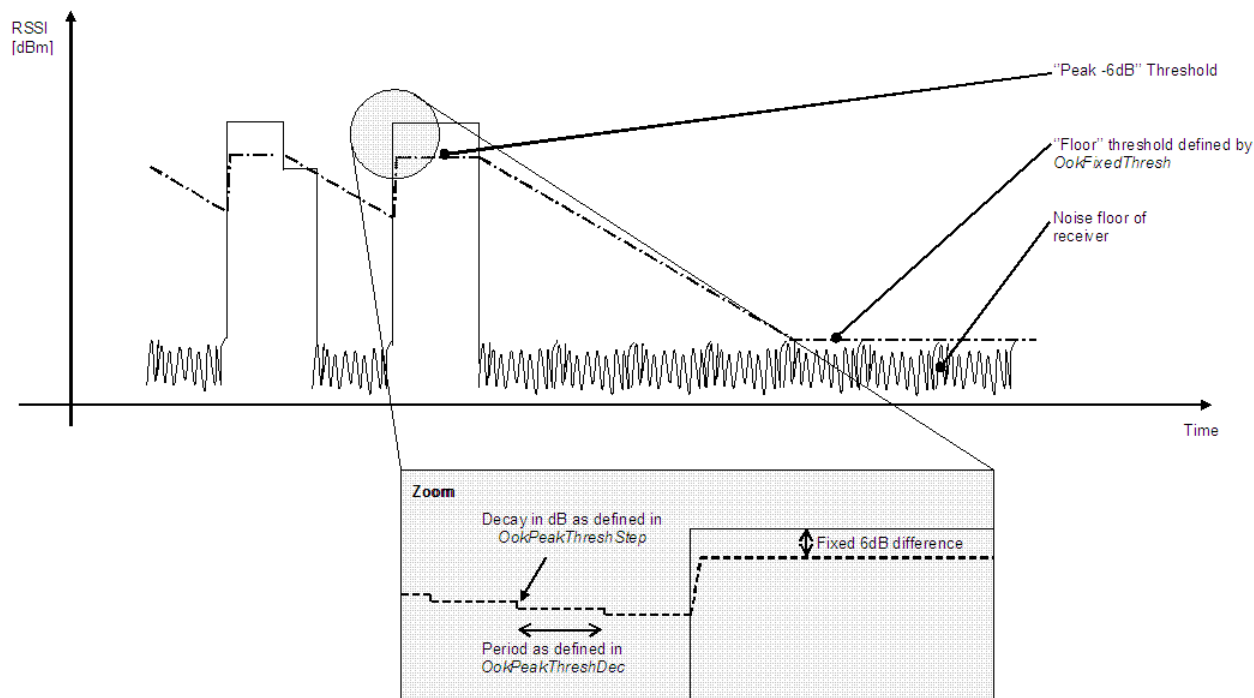


Figure 4-7. OOK Peak Demodulator Description

In peak threshold mode the comparison threshold level is the peak value of the RSSI, reduced by 6dB. In the absence of an input signal, or during the reception of a logical "0", the acquired peak value is decremented by one *OokPeakThreshStep* every *OokPeakThreshDec* period.

When the RSSI output is null for a long time (for instance after a long string of "0" received, or if no transmitter is present), the peak threshold level will continue falling until it reaches the "Floor Threshold", programmed in *OokFixedThresh*.

The default settings of the OOK demodulator lead to the performance stated in the electrical specification. However, in applications in which sudden signal drops are awaited during a reception, the three parameters should be optimized accordingly.

4.7.11.1 Optimizing the Floor Threshold

OokFixedThresh determines the sensitivity of the OOK receiver, as it sets the comparison threshold for weak input signals (i.e., those close to the noise floor). Significant sensitivity improvements can be generated if configured correctly.

Note that the noise floor of the receiver at the demodulator input depends on:

- The noise figure of the receiver.
- The gain of the receive chain from antenna to base band.
- The matching - including SAW filter if any.
- The bandwidth of the channel filters.

It is therefore important to note that the setting of *OokFixedThresh* will be application dependant. The following procedure is recommended to optimize *OokFixedThresh*.

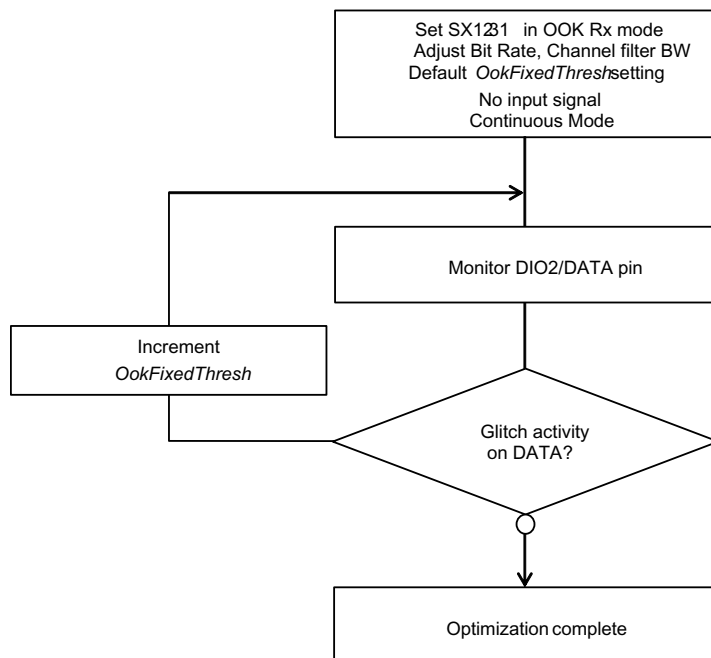


Figure 4-8. Floor Threshold Optimization

The new floor threshold value found during this test should be used for OOK reception with those receiver settings.

4.7.11.2 Optimizing OOK Demodulator for Fast Fading Signals

A sudden drop in signal strength can cause the bit error rate to increase. For applications where the expected signal drop can be estimated, the following OOK demodulator parameters *OokPeakThreshStep* and *OokPeakThreshDec* can be optimized as described below for a given number of threshold decrements per bit. Refer to *RegOokPeak* to access those settings.

4.7.11.3 Alternative OOK Demodulator Threshold Modes

In addition to the Peak OOK threshold mode, the user can alternatively select two other types of threshold detectors:

- Fixed Threshold: - The value is selected through *OokFixedThresh*
- Average Threshold - Data supplied by the RSSI block is averaged, and this operation mode should only be used with DC-free encoded data.

4.7.12 Bit Synchronizer

The Bit Synchronizer is a block that provides a clean and synchronized digital output, free of glitches. Its output is made available on pin DIO1/DCLK in Continuous mode and can be disabled through register

settings. However, for optimum receiver performance its use when running Continuous mode is strongly advised.

The Bit Synchronizer is automatically activated in Packet mode. Its bit rate is controlled by *BitRateMsb* and *BitRateLsb* in *RegBitrate*.

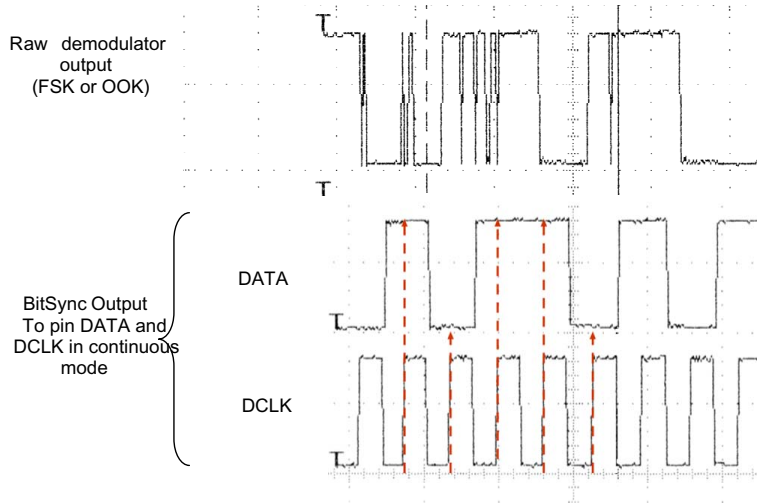


Figure 4-9. Bit Synchronizer Description

To ensure correct operation of the Bit Synchronizer, the following conditions have to be satisfied:

- A preamble (0x55 or 0xAA) of 12 bits is required for synchronization (from the *RxReady* interrupt)
- The subsequent payload bit stream must have at least one transition from '0' to '1' or '1' to '0' every 16 bits during data transmission
- The bit rate matching between the transmitter and the receiver must be better than 6.5 %.

NOTE

- If the Bit Rates of transmitter and receiver are known to be the same, the receiver will be able to receive an infinite unbalanced sequence (all “0s” or all ”1s”) with no restriction.
- If there is a difference in Bit Rate between Tx and Rx, the amount of adjacent bits at the same level that the BitSync can withstand can be estimated as follows:

$$NumberOfBits = \frac{1}{2} * \frac{BR}{\Delta BR}$$

- This implies approximately 6 consecutive unbalanced bytes when the Bit Rate precision is 1%, which is easily achievable (crystal tolerance is in the range of 50 to 100 ppm).

4.7.13 Frequency Error Indicator (FEI)

This function provides information about the frequency error of the local oscillator (LO) compared with the carrier frequency of a modulated signal at the input of the receiver. When the FEI block is launched, the frequency error is measured and the signed result is loaded in *FeiValue* in *RegFei*, in 2's complement format. The time required for an FEI evaluation is 4 times the bit period.

To ensure a proper behavior of the FEI:

- The operation must be done during the reception of preamble
- The sum of the frequency offset and the 20 dB signal bandwidth must be lower than the base band filter bandwidth

The 20 dB bandwidth of the signal can be evaluated as follows (double-side bandwidth):

$$BW_{20dB} = 2 \times \left(F_{DEV} + \frac{BR}{2} \right)$$

The frequency error, in Hz, can be calculated with the following formula:

$$FEI = F_{STEP} \times FeiValue$$

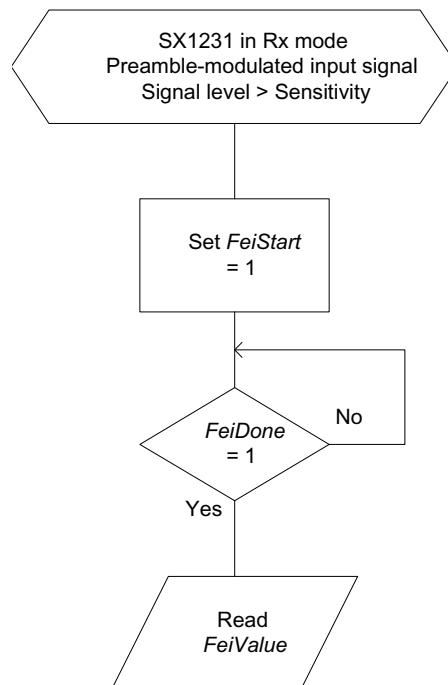


Figure 4-10. FEI Process

4.7.14 Automatic Frequency Correction (AFC)

The AFC is based on the FEI block, and therefore the same input signal and receiver setting conditions apply. When the AFC procedure is done, *AfcValue* is directly subtracted to the register that defines the frequency of operation of the chip, F_{RF} . The AFC can be launched:

- Each time the receiver is enabled, if *AfcAutoOn* = 1
- Upon user request, by setting bit *AfcStart* in *RegAfcFei*, if *AfcAutoOn* = 0

When the AFC is automatically triggered (*AfcAutoOn* = 1), the user has the option to:

- Clear the former AFC correction value, if *AfcAutoClearOn* = 1
- Start the AFC evaluation from the previously corrected frequency. This may be useful in systems in which the LO keeps on drifting in the “same direction”. Aging compensation is a good example.

The receiver offers an alternate receiver bandwidth setting during the AFC phase, to accommodate large LO drifts. If the user considers that the received signal may be out of the receiver bandwidth, a higher channel filter bandwidth can be programmed in *RegAfcBw*, at the expense of the receiver noise floor, which will impact sensitivity.

4.7.15 Optimized Setup for Low Modulation Index Systems

The following apply for optimizing low modulation index systems:

- For wide band systems, where AFC is usually not required (XTAL inaccuracies do not typically impact the sensitivity), it is recommended to offset the LO frequency of the receiver to avoid desensitization. This can be simply done by modifying *Frf* in *RegFrfLsb*. A good generalization is to offset the receiver’s LO by 10% of the expected transmitter frequency deviation.
- For narrow band systems, it is recommended to perform AFC. The receiver has a dedicated AFC, enabled when *AfcLowBetaOn* in *RegAfcCtrl* is set to 1. A frequency offset, programmable through *LowBetaAfcOffset* in *RegTestAfc*, is added and is calculated as follows:

$$\text{Offset} = \text{LowBetaAfcOffset} \times 488 \text{ Hz}$$

The user should ensure that the programmed offset exceeds the DC canceller’s cutoff frequency, set through *DccFreqAfc* in *RegAfcBw*.

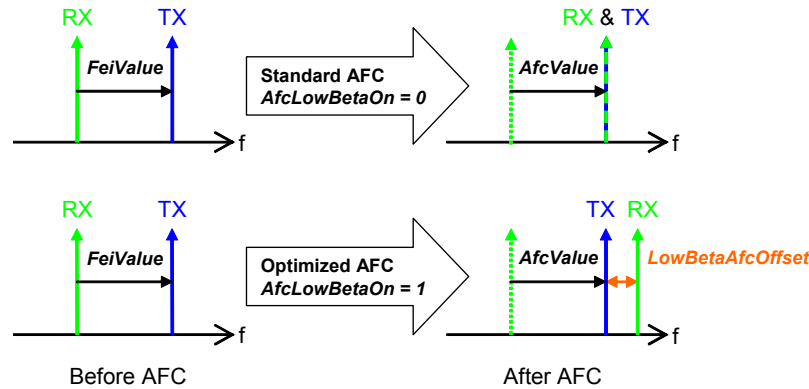


Figure 4-11. Optimized AFC (AfcLowBetaOn=1)

As shown on [Figure 4-11](#), a standard AFC sequence uses the result of the FEI to correct the LO frequency and align both local oscillators. When the optimized AFC is enabled ($AfcLowBetaOn=1$), the receiver’s LO is corrected by “ $FeiValue + LowBetaAfcOffset$ ”.

When the optimized AFC routine is enabled, the receiver startup time can be computed as follows:

$$TS_RE_AGC\&AFC \text{ (optimized AFC)} = Tana + 4.Tcf + 4.Tdcc + 3.Trssi + 2.Tafc + 2.Tpllafc$$

4.7.16 Temperature Sensor

When temperature is measured, the receiver ADC is used to digitize the sensor response. Most receiver blocks are disabled, and temperature measurement can only be triggered in Standby or Frequency Synthesizer modes.

The response of the temperature sensor is $-1^{\circ}\text{C} / \text{Lsb}$. A CMOS temperature sensor is not accurate by nature, therefore it should be calibrated at ambient temperature for precise temperature readings.

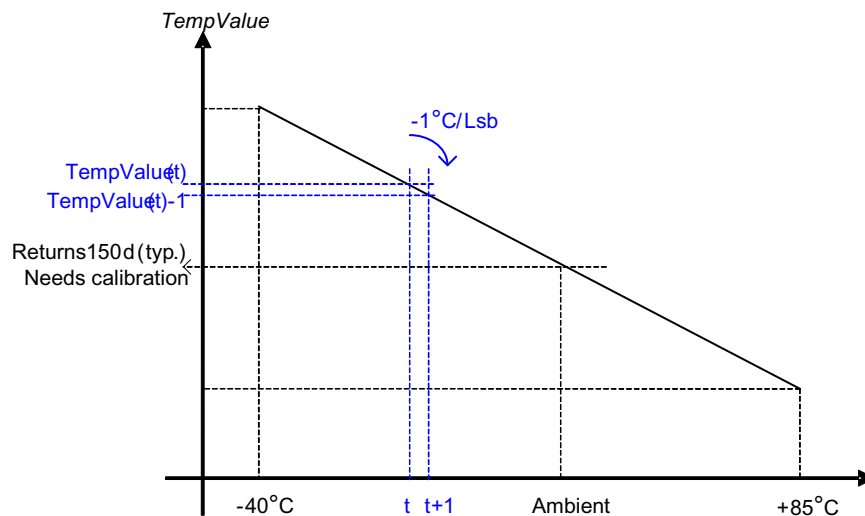


Figure 4-12. Temperature Sensor Response

It takes less than 100 microseconds for the transceiver to evaluate the temperature (from setting *TempMeasStart* to 1 to *TempMeasRunning* reset).

4.7.17 Timeout Function

The MC12311 includes a Timeout function, which allows it to automatically shut-down the receiver after a receive sequence and therefore save energy.

- Timeout interrupt is generated $TimeoutRxStart \times 8 \times Tbit$ after switching to RX mode if *RssiThreshold* flag does not raise within this time frame
- Timeout interrupt is generated $TimeoutRssiThresh \times 8 \times Tbit$ after *RssiThreshold* flag has been raised.

This timeout interrupt can be used to warn the MCU to shut down the receiver and return to a lower power mode.

Chapter 5

Transceiver Operating Modes

This chapter describes the operating modes of the MC12311 transceiver.

5.1 Basic Modes

The transceiver can be programmed to 5 different basic modes which are described in [Table 5-1](#).

By default, when switching from a mode to another one, the sub-blocks are woken up according to a pre-defined and optimized sequence. Alternatively, these operating modes can be selected directly by disabling the automatic sequencer (*SequencerOff* in *RegOpMode* = 1).

Table 5-1. Basic Transceiver Modes

ListenOn in <i>RegOpMode</i>	Mode in <i>RegOpMode</i>	Selected mode	Enabled blocks
0	0 0 0	Sleep Mode	None
0	0 0 1	Stand-by Mode	Top regulator and crystal oscillator
0	0 1 0	FS Mode	Frequency synthesizer
0	0 1 1	Transmit Mode	Frequency synthesizer and transmitter
0	1 0 0	Receive Mode	Frequency synthesizer and receiver
1	x	Listen Mode	See Listen Mode, section Section 5.3, "Listen Mode"

5.2 Automatic Sequencer and Wake-Up Times

By default, when switching from one operating mode to another, the circuit takes care of the sequence of events in such a way that the transition timing is optimized. For example, when switching from Sleep mode to Transmit mode, the device goes first to Standby mode (XO started), then to frequency synthesizer mode, and finally, when the PLL has locked, to transmit mode. Entering transmit mode is also made according to a predefined sequence starting with the wake-up of the PA regulator before applying a ramp-up on the PA and generating the DCLK clock.

- The crystal oscillator wake-up time, TS_OSC , is directly related to the time for the crystal oscillator to reach its steady state. It depends notably on the crystal characteristics.
- The frequency synthesizer wake-up time, TS_FS , is directly related to the time needed by the PLL to reach its steady state. The signal PLL_LOCK , provided on an external pin, gives an indication of the lock status. It goes high when the PLL reaches its locking range.

Four specific cases can be highlighted:

- Transmitter Wake Up time from Sleep mode = $TS_OSC + TS_FS + TS_TR$

- Receiver Wake Up time from Sleep mode = $TS_OSC + TS_FS + TS_RE$
- Receiver Wake Up time from Sleep mode, AGC enabled = $TS_OSC + TS_FS + TS_RE_AGC$
- Receiver Wake Up time from Sleep mode, AGC and AFC enabled = $TS_OSC + TS_FS + TS_RE_AGC\&AFC$

In applications where the target average power consumption, or the target startup time, do not require setting the transceiver in the lowest power modes (Sleep or Standby), the respective timings TS_OSC and TS_FS in the former equations can be omitted.

5.2.1 Transmitter Startup Time

The transmitter wake-up time, TS_TR , is given by the sequence controlled by the digital part. It is a pure digital delay which depends on the bit rate and the ramp-up time. In FSK mode, this time can be derived from the following equation.

$$TS_TR = 5\mu s + 1.25 \times PaRamp + \frac{1}{2} \times Tbit$$

where $PaRamp$ is the ramp-up time programmed in $RegPaRamp$ and $Tbit$ is the bit time.

In OOK mode, this equation can be simplified to the following:

$$TS_TR = 5\mu s + \frac{1}{2} \times Tbit$$

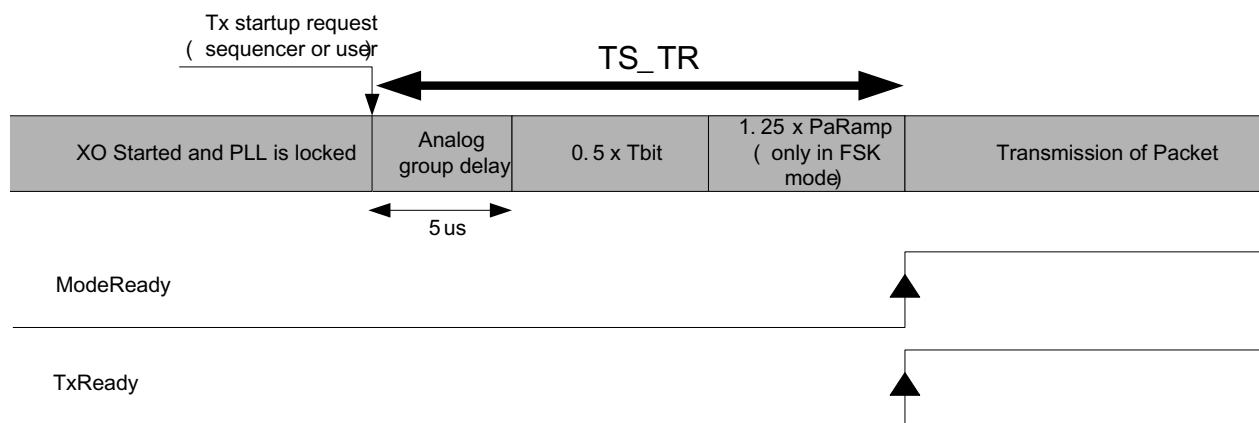


Figure 5-1. Tx Startup, FSK and OOK

5.2.2 Tx Start Procedure

As described in the former section, *ModeReady* and *TxReady* interrupts warn the MCU that the transmitter is ready to transmit data:

- In Continuous mode - the preamble bits preceding the payload can be applied on the DIO2/DATA pin immediately after any of these interrupts have fired. The DCLK signal, activated on pin DIO1/DCLK can also be used to start toggling the DATA pin, as described on [Figure 5-2](#).

- In Packet mode - the transmitter will automatically modulate the RF signal with preamble bytes as soon as *TxReady* or *ModeReady* happen. The actual packet transmission (starting with the number of preambles specified in *PreambleSize*) will start when the *TxStartCondition* is fulfilled.

5.2.3 Receiver Startup Time

It is highly recommended to use the built-in sequencer of the transceiver to optimize the delays when setting the chip in receive mode. It guarantees the shortest startup times, hence the lowest possible energy usage, for battery operated systems.

The startup times of the receiver can be calculated from the following:

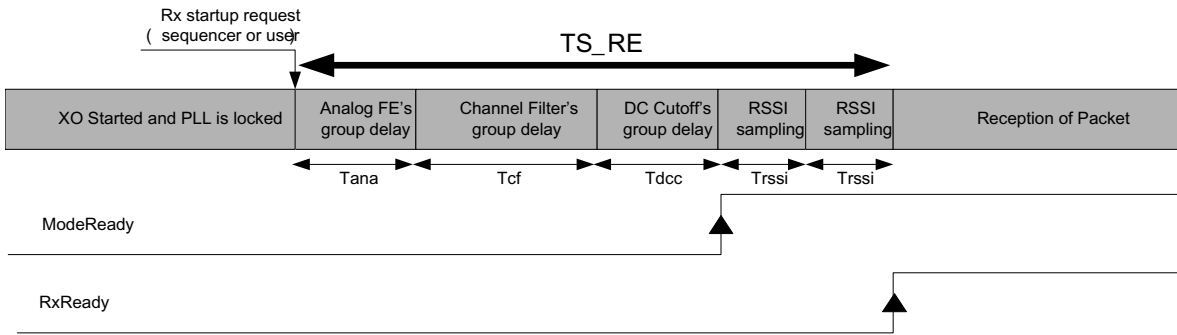


Figure 5-2. Rx Startup - No AGC, no AFC

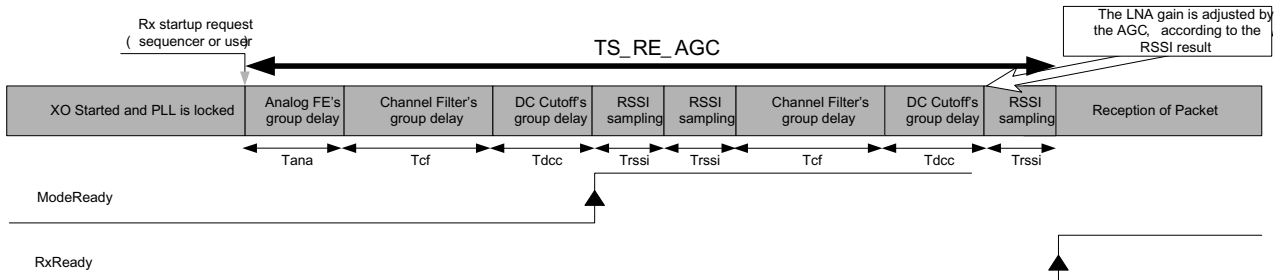


Figure 5-3. Rx Startup - AGC, no AFC

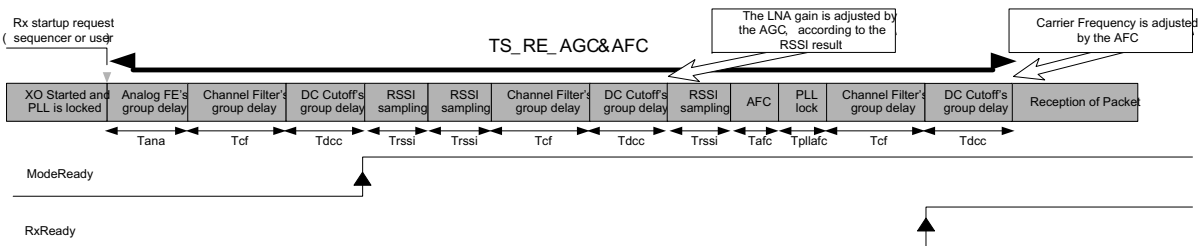


Figure 5-4. Rx Startup - AGC and AFC

The different timings shown above are as follows:

- Group delay of the analog front end - $Tana = 20 \text{ us}$
- Channel filter's group delay in FSK mode - $Tcf = 21 / (4.RxBw)$
- Channel filter's group delay in OOK mode - $Tcf = 34 / (4.RxBw)$

- DC Cutoff's group delay - $T_{dcc} = \max(8, 2^{\text{round}(\log_2(8 \cdot RxBw \cdot Tbit) + 1)}) / (4 \cdot RxBw)$
- PLL lock time after AFC adjustment - $T_{pllafc} = 5 / PLLBW$ ($PLLBW = 300 \text{ kHz}$)
- AFC sample time - $T_{afc} = 4 \times Tbit$ (also denoted TS_AFC in the general specification)
- RSSI sample time - $T_{rssi} = 2 \times \text{int}(4 \cdot RxBw \cdot Tbit) / (4 \cdot RxBw)$ (aka TS_RSSI)

NOTE

The above timings represent maximum settling times, and shorter settling times may be observed in real cases

5.2.4 Rx Start Procedure

As described in the former sections, the *RxReady* interrupt warns the MCU that the receiver is ready.

- In Continuous mode with Bit Synchronizer, the receiver will start locking its Bit Synchronizer on a minimum of 12 bits of received preamble, before the reception of correct Data, or Sync Word (if enabled) can occur.
- In Continuous mode without Bit Synchronizer, valid data will be available on DIO2/DATA right after the *RxReady* interrupt.
- In Packet mode, the receiver will start locking its Bit Synchronizer on a minimum of 12 bits of received preamble, before the reception of correct Data, or Sync Word (if enabled) can occur.

5.2.5 Optimized Frequency Hopping Sequences

In a frequency hopping-like application, it is required to turn off the transmitter when hopping from one channel to another, to avoid spectral splatter and obtain the best spectral purity.

- Transmitter hop from Ch A to Ch B - it is advised to step through the Rx mode:
 1. Transceiver is in Tx mode in Ch A
 2. Program the MC12311 in Rx mode
 3. Change the carrier frequency in the *RegFrf* registers
 4. Turn the transceiver back to Tx mode
 5. Respect the Tx start procedure
- Receiver hop from Ch A to Ch B -
 1. Transceiver is in Rx mode in Ch A
 2. Change the carrier frequency in the *RegFrf* registers Program the transceiver in FS mode
 3. Program the MC12311 in FS mode
 4. Turn the transceiver back to Rx mode
 5. Respect the Rx start procedure

NOTE

All sequences described above are assuming that the sequencer is turned on (SequencerOff=0 in RegOpMode).

5.3 Listen Mode

The circuit can be set to Listen mode, by setting *ListenOn* in *RegOpMode* to 1 while in Standby mode. In this mode, transceiver spends most of the time in Idle mode, during which only the RC oscillator runs. Periodically the receiver is woken up and listens for an RF signal. If a wanted signal is detected, the receiver is kept on and the data is demodulated.

Otherwise, if a wanted signal hasn't been detected after a pre-defined period of time, the receiver is disabled until the next time period.

This periodical Rx wake-up requirement is very common in low power applications. On the transceiver it is handled locally by the Listen mode block without using the MCU resources.

The simplified timing diagram of this procedure is illustrated in [Figure 5-5](#).

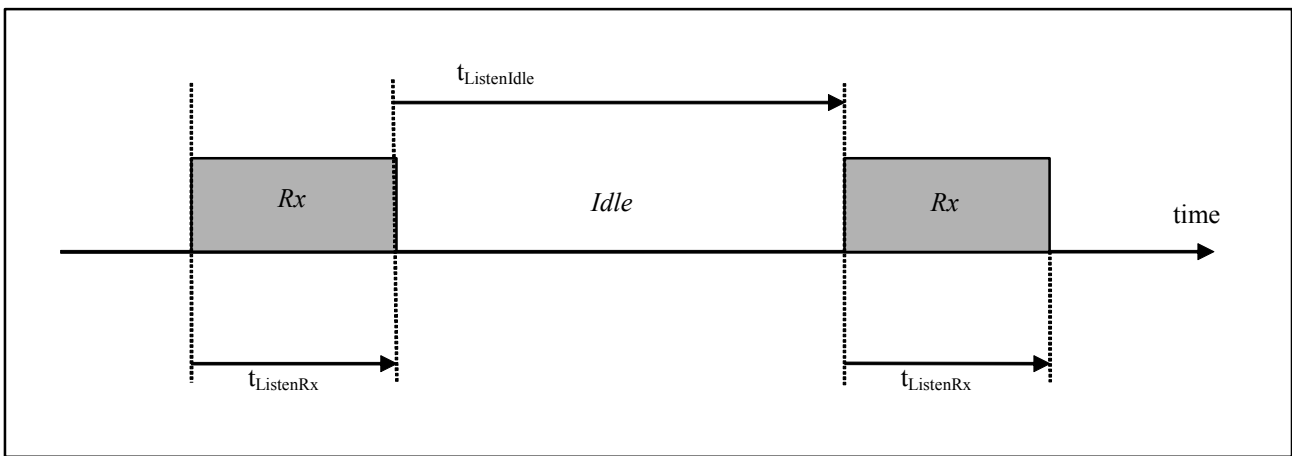


Figure 5-5. Listen Mode Sequence (no wanted signal is received)

5.3.1 Timing

The duration of the Idle phase is given by $t_{ListenIdle}$. The time during which the receiver is on and waits for a signal is given by $t_{ListenRx}$. $t_{ListenRx}$ includes the wake-up time of the receiver, described in [section 5.2.3](#). This duration can be programmed in the configuration registers via the SPI.

Both time periods $t_{ListenRx}$ and $t_{ListenIdle}$ (denoted $t_{ListenX}$ in the following text) are fixed by two parameters from the configuration register and are calculated as follows

:

$$t_{ListenX} = ListenCoefX \cdot ListenResolX$$

where *ListenResolX* is the Rx or Idle resolution and is independently programmable on three values (64us, 4.1ms or 262ms), whereas *ListenCoefX* is an integer between 1 and 255. All parameters are located in *RegListen* registers.

The timing ranges are tabulated in [Table 5-2](#) below.

Table 5-2. Range of Durations in Listen Mode

ListenResolX	Min duration (<i>ListenCoef</i> = 1)	Max duration (<i>ListenCoef</i> = 255)
01	64 us	16 ms
10	4.1 ms	1.04 s
11	0.26 s	67 s

NOTE

- The accuracy of the typical timings given in [Table 5-2](#) will depend in the RC oscillator calibration
- RC oscillator calibration is required, and must be performed at power up. See [Section 5.3.4, “RC Timer Accuracy”](#) for details

5.3.2 Criteria

The criteria taken for detecting a wanted signal and hence deciding to maintain the receiver on is defined by *ListenCriteria* in *RegListen1*.

Table 5-3. Signal Acceptance Criteria in Listen Mode

ListenCriteria	Input Signal Power >= <i>RssiThreshold</i>	SyncAddressMatch
0	Required	Not Required
1	Required	Required

5.3.3 End of Cycle Actions

The action taken after detection of a packet, is defined by *ListenEnd* in *RegListen3*, as described in the table below.

Table 5-4. End of Listen Cycle Actions

ListenEnd	Description
00	Chip stays in Rx mode. Listen mode stops and must be disabled.
01	Chip stays in Rx mode until <i>PayloadReady</i> or <i>Timeout</i> interrupt occurs. It then goes to the mode defined by <i>Mode</i> . Listen mode stops and must be disabled.
10	Chip stays in Rx mode until <i>PayloadReady</i> or <i>Timeout</i> interrupt occurs. Listen mode then resumes in Idle state. FIFO content is lost at next Rx wakeup.

Upon detection of a valid packet, the sequencing is altered, as shown below:

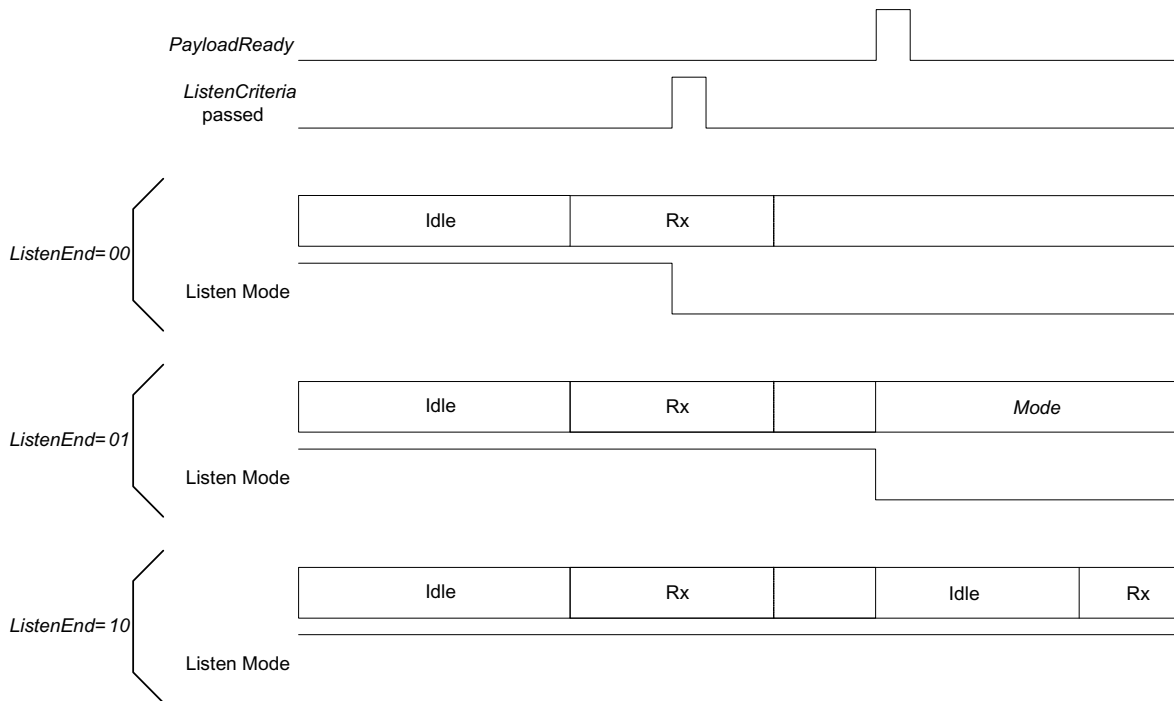


Figure 5-6. Listen Mode Sequence (wanted signal is received)

Listen mode can be disabled by writing *ListenOn* to 0.

5.3.4 RC Timer Accuracy

All timings of the Listen Mode rely on the accuracy of the internal low-power RC oscillator. This oscillator is automatically calibrated at the device power-up, and it is a user-transparent process.

For applications enduring large temperature variations, and for which the power supply is never removed, RC calibration can be performed upon user request. *RcCalStart* in *RegOsc1* can be used to trigger this calibration, and the flag *RcCalDone* will be set automatically when the calibration is over.

5.4 AutoModes

Automatic modes of packet handler can be enabled by configuring the related parameters in *RegAutoModes*. The intermediate mode of the chip is called *IntermediateMode* and the enter and exit conditions to/from this intermediate mode can be configured through the parameters *EnterCondition* & *ExitCondition*. The enter and exit conditions cannot be used independently of each other i.e. both should be enabled at the same time.

The initial and the final state is the one configured in *Mode* in *RegOpMode*. The initial & final states can be different by configuring the modes register while the chip is in intermediate mode. The pictorial description of the auto modes is shown below.

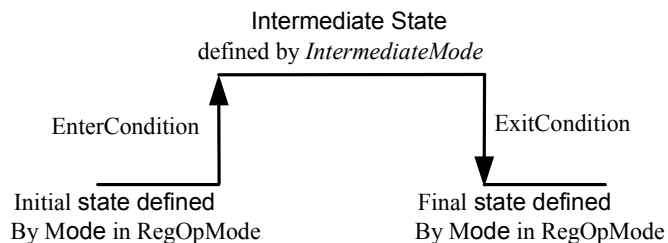


Figure 5-7. Auto Modes of Packet Handler

Some typical examples of AutoModes usage are described below:

- Automatic transmission (AutoTx) : *Mode = Sleep, IntermediateMode = Tx, EnterCondition = FifoLevel, ExitCondition = PacketSent*
- Automatic reception (AutoRx) : *Mode = Rx, IntermediateMode = Sleep, EnterCondition = CrcOk, ExitCondition = falling edge of FifoNotEmpty*
- Automatic reception of acknowledge (AutoRxAck): *Mode = Tx, IntermediateMode = Rx, EnterCondition = PacketSent, ExitCondition = CrcOk*

Chapter 6

Transceiver Digital Control and Communications

6.1 Overview

The following figure shows the MC12311 data processing circuit. Its role is to interface the data to/from the modulator/demodulator and the uC access points (SPI and DIO pins). It also controls all the configuration registers.

The circuit contains several control blocks which are described in the following paragraphs.

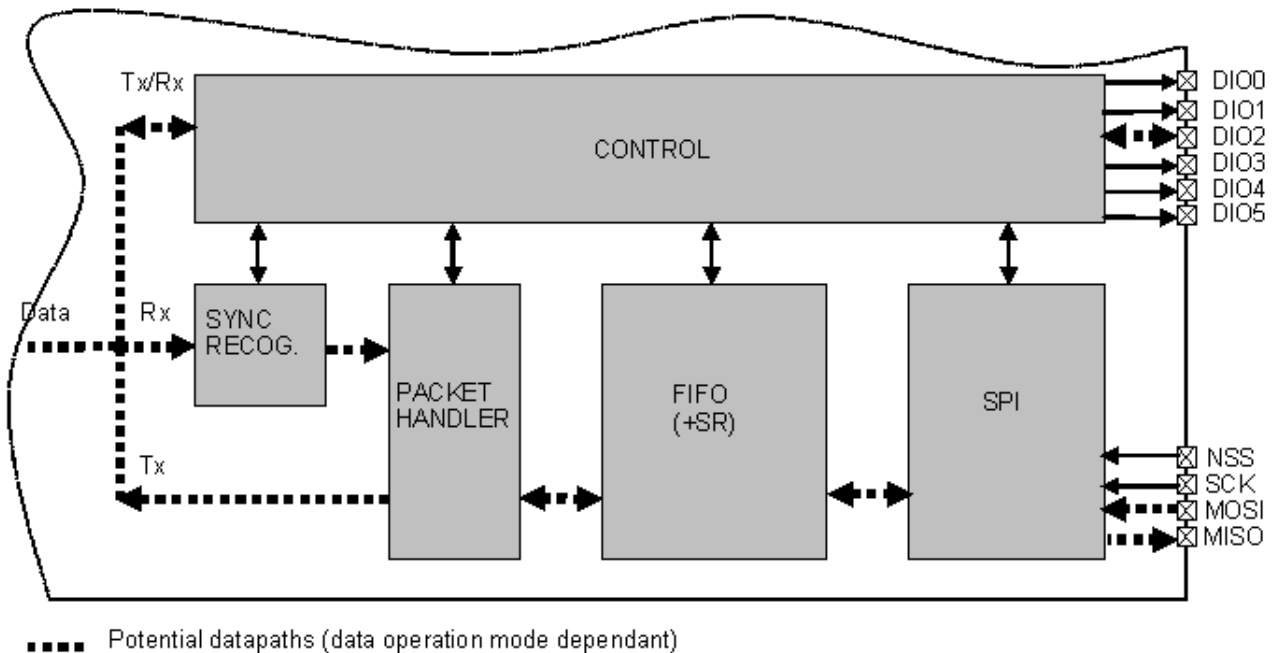


Figure 6-1. MC12311 Data Processing Conceptual View

The MC12311 implements several data operation modes, each with their own data path through the data processing section. Depending on the data operation mode selected, some control blocks are active whilst others remain disabled.

6.1.1 Data Operation Modes

The MC12311 has two different data operation modes selectable by the user:

- Continuous mode: each bit transmitted or received is accessed in real time at the DIO2/DATA pin. This mode may be used if adequate external signal processing is available.

- **Packet mode (recommended):** user only provides/retrieves payload bytes to/from the FIFO. The packet is automatically built with preamble, Sync word, and optional AES, CRC, and DC-free encoding schemes. The reverse operation is performed in reception. The uC processing overhead is hence significantly reduced compared to Continuous mode. Depending on the optional features activated (CRC, AES, etc) the maximum payload length is limited to FIFO size, 255 bytes or unlimited.

Each of these data operation modes is described fully in the following sections.

6.2 Control Block Description

6.2.1 SPI Interface

The SPI interface gives access to the configuration register via a synchronous full-duplex protocol corresponding to CPOL = 0 and CPHA = 0 in Motorola/Freescale nomenclature. Only the slave side is implemented.

Three access modes to the registers are provided:

- **SINGLE access:** an address byte followed by a data byte is sent for a write access whereas an address byte is sent and a read byte is received for the read access. The NSS pin goes low at the begin of the frame and goes high after the data byte.
- **BURST access:** the address byte is followed by several data bytes. The address is automatically incremented internally between each data byte. This mode is available for both read and write accesses. The NSS pin goes low at the beginning of the frame and stay low between each byte. It goes high only after the last byte transfer.
- **FIFO access:** if the address byte corresponds to the address of the FIFO, then succeeding data byte will address the FIFO. The address is not automatically incremented but is memorized and does not need to be sent between each data byte. The NSS pin goes low at the beginning of the frame and stay low between each byte. It goes high only after the last byte transfer.

Figure below shows a typical SPI single access to a register.

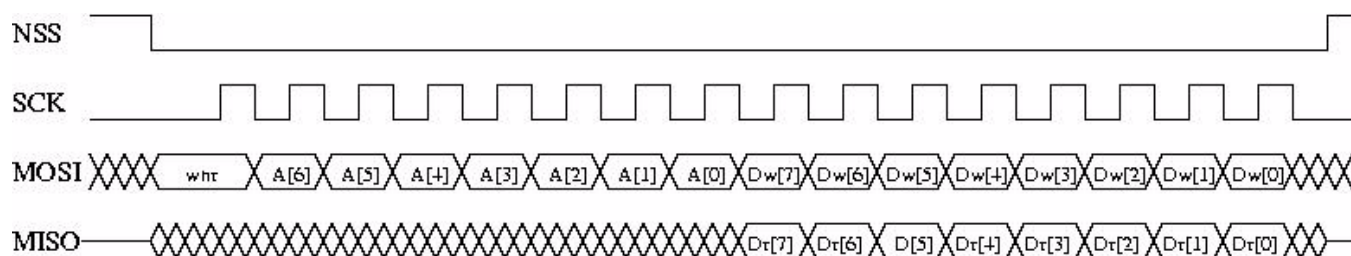


Figure 6-2. SPI Timing Diagram (single access)

MOSI is generated by the master on the falling edge of SCK and is sampled by the slave (i.e. this SPI interface) on the rising edge of SCK. MISO is generated by the slave on the falling edge of SCK.

A transfer always starts by the NSS pin going low. MISO is high impedance when NSS is high.

The first byte is the address byte. It is made of:

- wnr bit, which is 1 for write access and 0 for read access
- 7 bits of address, MSB first

The second byte is a data byte, either sent on MOSI by the master in case of a write access, or received by the master on MISO in case of read access. The data byte is transmitted MSB first.

Proceeding bytes may be sent on MOSI (for write access) or received on MISO (for read access) without rising NSS and re-sending the address. In FIFO mode, if the address was the FIFO address then the bytes will be written / read at the FIFO address. In Burst mode, if the address was not the FIFO address, then it is automatically incremented at each new byte received.

The frame ends when NSS goes high. The next frame must start with an address byte. The SINGLE access mode is actually a special case of FIFO / BURST mode with only 1 data byte transferred.

During the write access, the byte transferred from the slave to the master on the MISO line is the value of the written register before the write operation.

6.2.2 FIFO

6.2.2.1 Overview and Shift Register (SR)

In packet mode of operation, both data to be transmitted and that has been received are stored in a configurable FIFO (First In First Out) device. It is accessed via the SPI interface and provides several interrupts for transfer management.

The FIFO is 1 byte wide hence it only performs byte (parallel) operations, whereas the demodulator functions serially. A shift register is therefore employed to interface the two devices. In transmit mode it takes bytes from the FIFO and outputs them serially (MSB first) at the programmed bit rate to the modulator. Similarly, in Rx the shift register gets bit by bit data from the demodulator and writes them byte by byte to the FIFO. This is illustrated in figure below.

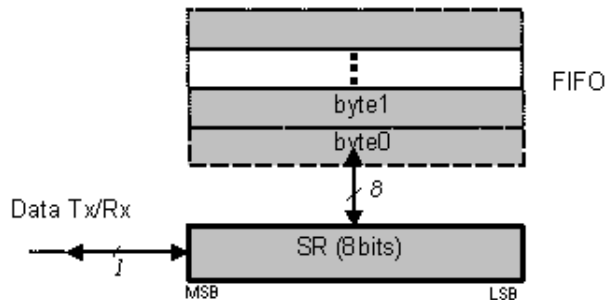


Figure 6-3. FIFO and Shift Register (SR)

NOTE

When switching to Sleep mode, the FIFO can only be used once the ModeReady flag is set (quasi immediate from all modes except from Tx)

6.2.2.2 Size

The FIFO size is fixed to 66 bytes.

6.2.2.3 Interrupt Sources and Flags

- *FifoNotEmpty*: *FifoNotEmpty* interrupt source is low when byte 0, i.e. whole FIFO, is empty. Otherwise it is high. Note that when retrieving data from the FIFO, *FifoNotEmpty* is updated on NSS falling edge, i.e. when *FifoNotEmpty* is updated to low state the currently started read operation must be completed. In other words, *FifoNotEmpty* state must be checked after each read operation for a decision on the next one (*FifoNotEmpty* = 1: more byte(s) to read; *FifoNotEmpty* = 0: no more byte to read).
- *FifoFull*: *FifoFull* interrupt source is high when the last FIFO byte, i.e. the whole FIFO, is full. Otherwise it is low.
- *FifoOverrunFlag*: *FifoOverrunFlag* is set when a new byte is written by the user (in Tx or Standby modes) or the SR (in Rx mode) while the FIFO is already full. Data is lost and the flag should be cleared by writing a 1, note that the FIFO will also be cleared.
- *PacketSent*: *PacketSent* interrupt source goes high when the SR's last bit has been sent.
- *FifoLevel*: Threshold can be programmed by *FifoThreshold* in *RegFifoThresh*. Its behavior is illustrated in figure below.

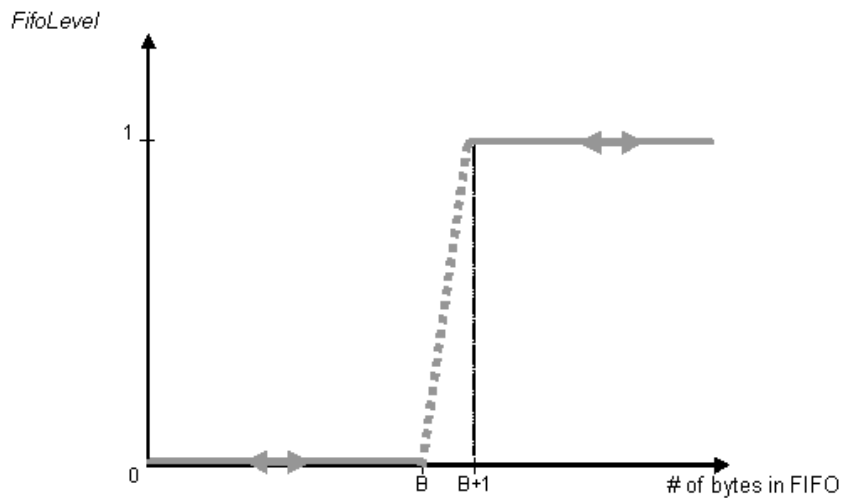


Figure 6-4. FifoLevel IRQ Source Behavior

NOTE

FifoLevel interrupt is updated only after a read or write operation on the FIFO. Thus the interrupt cannot be dynamically updated by only changing the *FifoThreshold* parameter.

FifoLevel interrupt is valid as long as *FifoFull* does not occur. An empty FIFO will restore its normal operation

6.2.2.4 FIFO Clearing

Table below summarizes the status of the FIFO when switching between different modes.

Table 6-1. Status of FIFO when Switching Between Different Modes of the Chip

From	To	FIFO status	Comments
Stdby	Sleep	Not cleared	
Sleep	Stdby	Not cleared	
Stdby/Sleep	Tx	Not cleared	To allow the user to write the FIFO in Stdby/Sleep before Tx
Stdby/Sleep	Rx	Cleared	
Rx	Tx	Cleared	
Rx	Stdby/Sleep	Not cleared	To allow the user to read FIFO in Stdby/Sleep mode after Rx
Tx	Any	Cleared	

6.2.3 Sync Word Recognition

6.2.3.1 Overview

Sync word recognition (also called Pattern recognition) is activated by setting *SyncOn* in *RegSyncConfig*. The bit synchronizer must also be activated in continuous mode (automatically done in Packet mode).

The block behaves like a shift register; it continuously compares the incoming data with its internally programmed Sync word and sets *SyncAddressMatch* when a match is detected. This is illustrated in Figure 6-5 below.

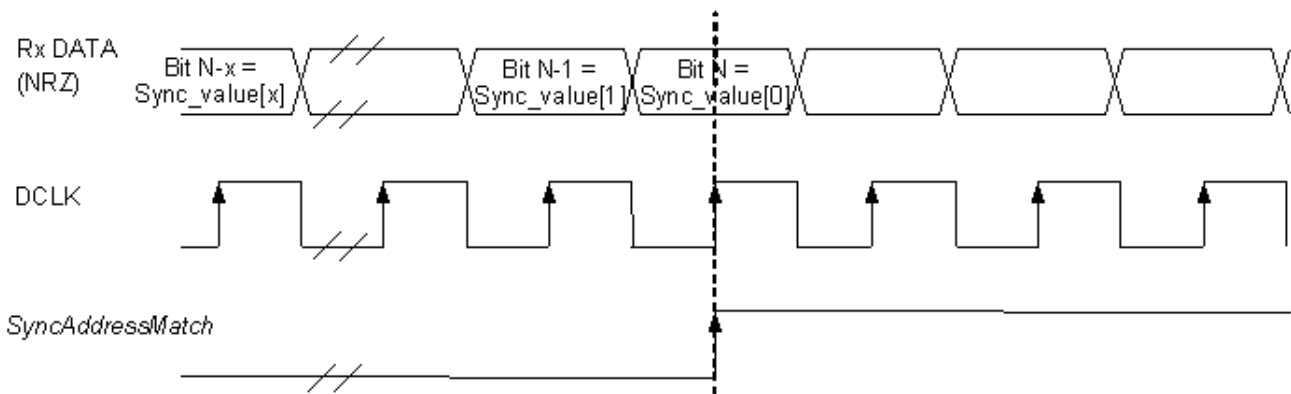


Figure 6-5. Sync Word Recognition

During the comparison of the demodulated data, the first bit received is compared with bit 7 (MSB) of *RegSyncValue1* and the last bit received is compared with bit 0 (LSB) of the last byte whose address is determined by the length of the Sync word.

When the programmed Sync word is detected the user can assume that this incoming packet is for the node and can be processed accordingly.

SyncAddressMatch is cleared when leaving Rx or FIFO is emptied.

6.2.3.2 Configuration

- Size: Sync word size can be set from 1 to 8 bytes (i.e. 8 to 64 bits) via *SyncSize* in *RegSyncConfig*. In Packet mode this field is also used for Sync word generation in Tx mode.
- Error tolerance: The number of errors tolerated in the Sync word recognition can be set from 0 to 7 bits to via *SyncTol*.
- Value: The Sync word value is configured in *SyncValue(63:0)*. In Packet mode this field is also used for Sync word generation in Tx mode.

NOTE

SyncValue choices containing 0x00 bytes are not allowed.

6.2.4 Packet Handler

The packet handler is the block used in Packet mode. Its functionality is fully described in section 6.5.

6.2.5 Control

The control block configures and controls the full chip's behavior according to the settings programmed in the configuration registers.

6.3 Digital IO Pins Mapping

Six general purpose IO pins are available on the MC12311, and their configuration in Continuous or Packet mode is controlled through *RegDioMapping1* and *RegDioMapping2*.

6.3.1 DIO Pins Mapping in Continuous Mode

Table 6-2. DIO Mapping, Continuous Mode

Mode	Diox Mapping	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
Sleep	00	-	-	-	-	-	-
	01	-	-	-	-	-	-
	10	LowBat	LowBat	AutoMode	-	LowBat	LowBat
	11	ModeReady	-	-	-	-	ModeReady
Stdby	00	ClkOut	-	-	-	-	-
	01	-	-	-	-	-	-
	10	LowBat	LowBat	AutoMode	-	LowBat	LowBat
	11	ModeReady	-	-	-	-	ModeReady
FS	00	ClkOut	-	-	-	-	PIILock
	01	-	-	-	-	-	-
	10	LowBat	LowBat	AutoMode	-	LowBat	LowBat
	11	ModeReady	PIILock	-	-	PIILock	ModeReady
Rx	00	ClkOut	Timeout	Rssi	Data	Dclk	SyncAddress
	01	Rssi	RxReady	RxReady	Data	RxReady	Timeout
	10	LowBat	SyncAddress	AutoMode	Data	LowBat	Rssi
	11	ModeReady	PIILock	Timeout	Data	SyncAddress	ModeReady
Tx	00	ClkOut	TxReady	TxReady	Data	Dclk	PIILock
	01	ClkOut	TxReady	TxReady	Data	TxReady	TxReady
	10	LowBat	LowBat	AutoMode	Data	LowBat	LowBat
	11	ModeReady	PIILock	TxReady	Data	PIILock	ModeReady

6.3.2 DIO Pins Mapping in Packet Mode

Table 6-3. DIO Mapping, Packet Mode

Mode	Diox Mapping	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
Sleep	00	-	-	FifoFull	FifoNotEmpty	FifoLevel	-
	01	-	-	-	-	FifoFull	-
	10	LowBat	LowBat	LowBat	LowBat	FifoNotEmpty	LowBat
	11	ModeReady	-	-	AutoMode	-	-
Stdby	00	ClkOut	-	FifoFull	FifoNotEmpty	FifoLevel	-
	01	-	-	-	-	FifoFull	-
	10	LowBat	LowBat	LowBat	LowBat	FifoNotEmpty	LowBat
	11	ModeReady	-	-	AutoMode	-	-
FS	00	ClkOut	-	FifoFull	FifoNotEmpty	FifoLevel	-
	01	-	-	-	-	FifoFull	-
	10	LowBat	LowBat	LowBat	LowBat	FifoNotEmpty	LowBat
	11	ModeReady	PIILock	PIILock	AutoMode	PIILock	PIILock
Rx	00	ClkOut	Timeout	FifoFull	FifoNotEmpty	FifoLevel	CrcOk
	01	Data	Rssi	Rssi	Data	FifoFull	PayloadReady
	10	LowBat	RxReady	SyncAddress	LowBat	FifoNotEmpty	SyncAddress
	11	ModeReady	PIILock	PIILock	AutoMode	Timeout	Rssi
Tx	00	ClkOut	ModeReady	FifoFull	FifoNotEmpty	FifoLevel	PacketSent
	01	Data	TxReady	TxReady	Data	FifoFull	TxReady
	10	LowBat	LowBat	LowBat	LowBat	FifoNotEmpty	LowBat
	11	ModeReady	PIILock	PIILock	AutoMode	PIILock	PIILock

NOTE

Received Data is only shown on the Data signal between RxReady and PayloadReady's rising edges

6.4 Continuous Mode

6.4.1 General Description

As illustrated in [Figure 6-6](#), in Continuous mode the NRZ data to (from) the (de)modulator is directly accessed by the uC on the bidirectional DIO2/DATA pin. The FIFO and packet handler are thus inactive.

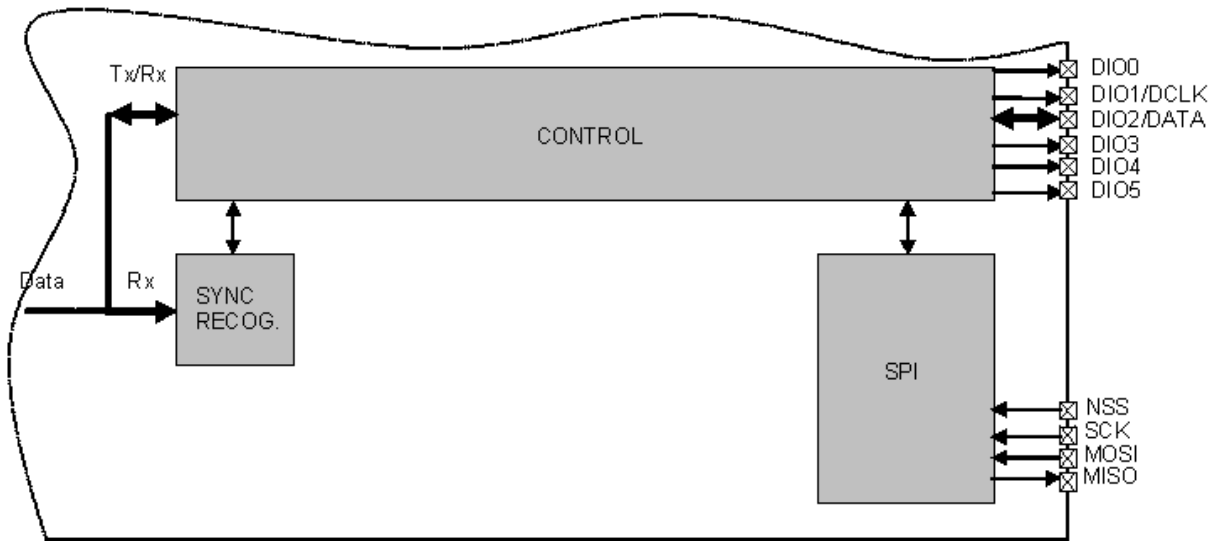


Figure 6-6. Continuous Mode Conceptual View

6.4.2 Tx Processing

In Tx mode, a synchronous data clock for an external uC is provided on DIO1/DCLK pin. Clock timing with respect to the data is illustrated in [Figure 6-7](#). DATA is internally sampled on the rising edge of DCLK so the uC can change logic state anytime outside the grayed out setup/hold zone.

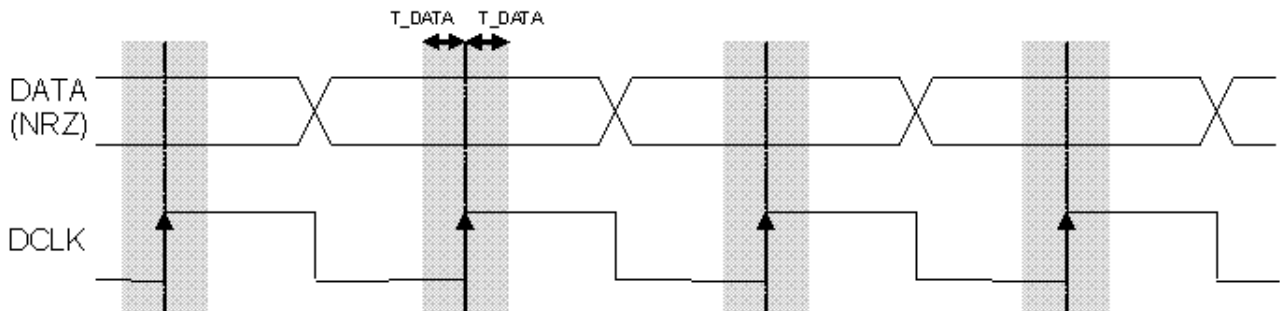


Figure 6-7. Tx Processing in Continuous Mode

NOTE

The use of DCLK is required when the modulation shaping is enabled.

6.4.3 Rx Processing

If the bit synchronizer is disabled, the raw demodulator output is made directly available on DATA pin and no DCLK signal is provided.

Conversely, if the bit synchronizer is enabled, synchronous cleaned data and clock are made available respectively on DIO2/DATA and DIO1/DCLK pins. DATA is sampled on the rising edge of DCLK and updated on the falling edge as illustrated below.

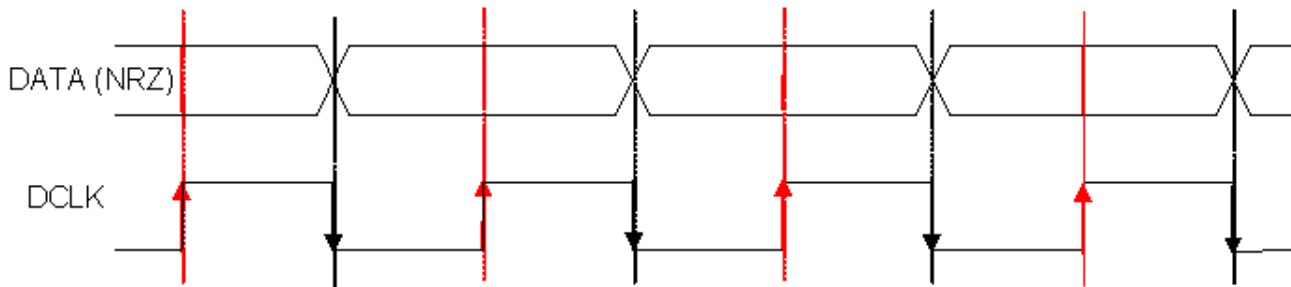


Figure 6-8. Rx Processing in Continuous Mode

NOTE

In Continuous mode it is always recommended to enable the bit synchronizer to clean the DATA signal even if the DCLK signal is not used by the uC (bit synchronizer is automatically enabled in Packet mode).

6.5 Packet Mode

6.5.1 General Description

In Packet mode the NRZ data to (from) the (de)modulator is not directly accessed by the uC but stored in the FIFO and accessed via the SPI interface.

In addition, the MC12311 packet handler performs several packet oriented tasks such as Preamble and Sync word generation, CRC calculation/check, whitening/dewhitening of data, Manchester encoding/decoding, address filtering, AES encryption/decryption, etc. This simplifies software and reduces uC overhead by performing these repetitive tasks within the RF chip itself.

Another important feature is ability to fill and empty the FIFO in Sleep/Stdby mode, ensuring optimum power consumption and adding more flexibility for the software.

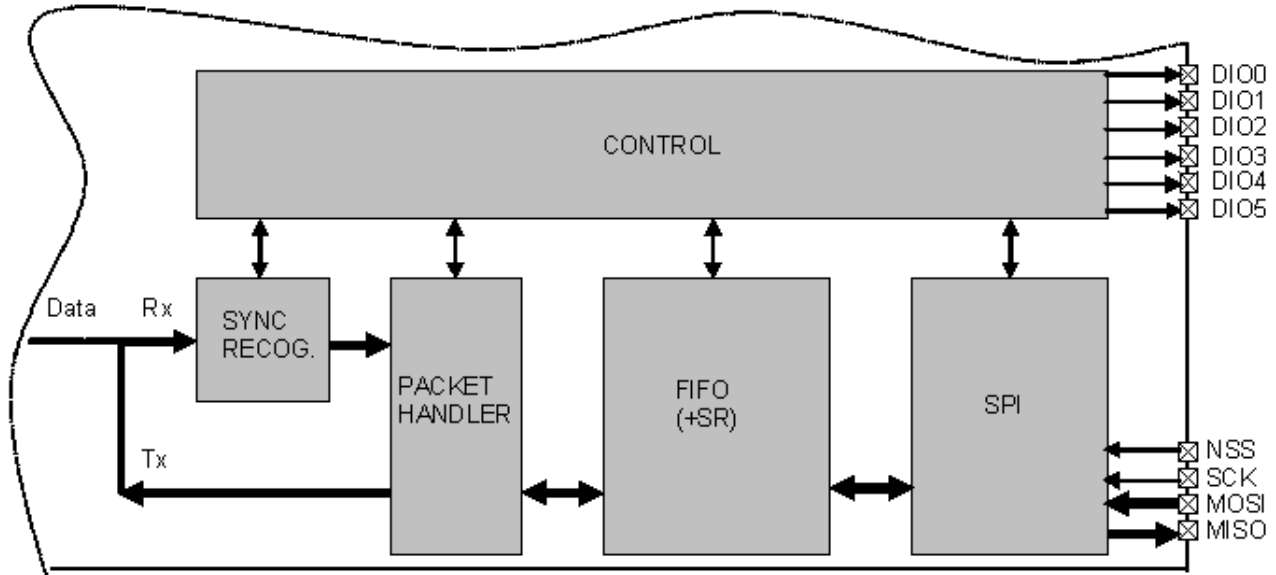


Figure 6-9. Packet Mode Conceptual View

NOTE

The Bit Synchronizer is automatically enabled in Packet mode.

6.5.2 Packet Format

6.5.2.1 Fixed Length Packet Format

Fixed length packet format is selected when bit *PacketFormat* is set to 0 and *PayloadLength* is set to any value greater than 0.

In applications where the packet length is fixed in advance, this mode of operation may be of interest to minimize RF overhead (no length byte field is required). All nodes, whether Tx only, Rx only, or Tx/Rx should be programmed with the same packet length value.

The length of the payload is limited to 255 bytes if AES is not enabled else the message is limited to 64 bytes (i.e. max 65 bytes payload if Address byte is enabled).

The length programmed in *PayloadLength* relates only to the payload which includes the message and the optional address byte. In this mode, the payload must contain at least one byte, i.e. address or message byte.

An illustration of a fixed length packet is shown below. It contains the following fields:

- Preamble (1010...)
- Sync word (Network ID)
- Optional Address byte (Node ID)
- Message data
- Optional 2-bytes CRC checksum

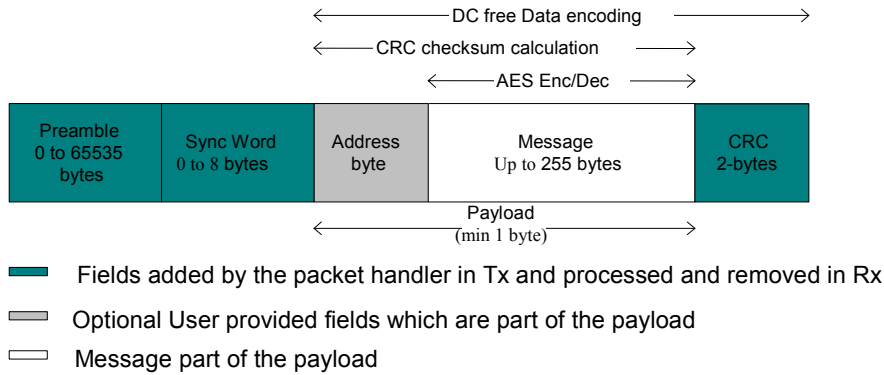


Figure 6-10. Fixed Length Packet Format

6.5.2.2 Variable Length Packet Format

Variable length packet format is selected when bit *PacketFormat* is set to 1.

This mode is useful in applications where the length of the packet is not known in advance and can vary over time. It is then necessary for the transmitter to send the length information together with each packet in order for the receiver to operate properly.

In this mode the length of the payload, indicated by the length byte, is given by the first byte of the FIFO and is limited to 255 bytes if AES is not enabled else the message is limited to 64 bytes (i.e. max 66 bytes payload if Address byte is enabled). Note that the length byte itself is not included in its calculation. In this mode, the payload must contain at least 2 bytes, i.e. length + address or message byte.

An illustration of a variable length packet is shown below. It contains the following fields:

- Preamble (1010...)
- Sync word (Network ID)
- Length byte
- Optional Address byte (Node ID)
- Message data
- Optional 2-bytes CRC checksum

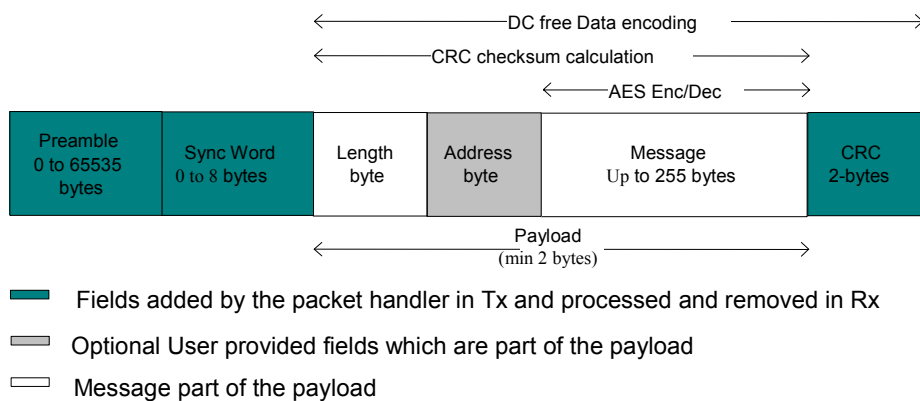


Figure 6-11. Variable Length Packet Format

6.5.2.3 Unlimited Length Packet Format

Unlimited length packet format is selected when bit *PacketFormat* is set to 0 and *PayloadLength* is set to 0.

The user can then transmit and receive packet of arbitrary length and *PayloadLength* register is not used in Tx/Rx modes for counting the length of the bytes transmitted/received. This mode is a replacement for the legacy buffered mode in SX1211/SX1212 transceivers.

In Tx the data is transmitted depending on the *TxStartCondition* bit. On the Rx side the data processing features like Address filtering, Manchester encoding and data whitening are not available if the sync pattern length is set to zero (*SyncOn* = 0). The filling of the FIFO in this case can be controlled by the bit *FifoFillCondition*. The CRC detection in Rx is also not supported in this mode of the packet handler, however CRC generation in Tx is operational. The interrupts like *CrcOk* & *PayloadReady* are not available either.

An unlimited length packet is made up of the following fields:

- Preamble (1010...).
- Sync word (Network ID).
- Optional Address byte (Node ID).
- Message data
- Optional 2-bytes CRC checksum (Tx only)

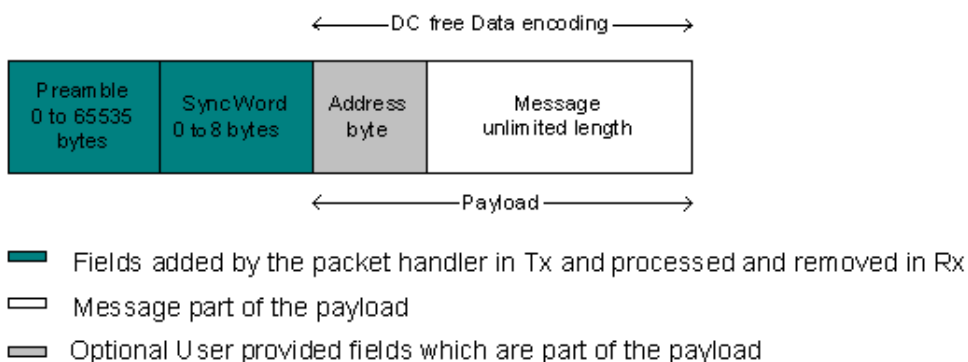


Figure 6-12. Unlimited Length Packet Format

6.5.3 Tx Processing (without AES)

In Tx mode the packet handler dynamically builds the packet by performing the following operations on the payload available in the FIFO:

- Add a programmable number of preamble bytes
- Add a programmable Sync word
- Optionally calculating CRC over complete payload field (optional length byte + optional address byte + message) and appending the 2 bytes checksum.
- Optional DC-free encoding of the data (Manchester or whitening)

Only the payload (including optional address and length fields) is required to be provided by the user in the FIFO.

The transmission of packet data is initiated by the Packet Handler only if the chip is in Tx mode and the transmission condition defined by *TxStartCondition* is fulfilled. If transmission condition is not fulfilled then the packet handler transmits a preamble sequence until the condition is met. This happens only if the preamble length $\neq 0$, otherwise it transmits a zero or one until the condition is met to transmit the packet data.

The transmission condition itself is defined as:

- if *TxStartCondition* = 1, the packet handler waits until the first byte is written into the FIFO, then it starts sending the preamble followed by the sync word and user payload
- If *TxStartCondition* = 0, the packet handler waits until the number of bytes written in the FIFO is equal to the number defined in *RegFifoThresh* + 1
- If the condition for transmission was already fulfilled i.e. the FIFO was filled in Sleep/Standby then the transmission of packet starts immediately on enabling Tx

6.5.4 Rx Processing (without AES)

In Rx mode the packet handler extracts the user payload to the FIFO by performing the following operations:

- Receiving the preamble and stripping it off
- Detecting the Sync word and stripping it off
- Optional DC-free decoding of data
- Optionally checking the address byte
- Optionally checking CRC and reflecting the result on *CrcOk*.

Only the payload (including optional address and length fields) is made available in the FIFO.

When the Rx mode is enabled the demodulator receives the preamble followed by the detection of sync word. If fixed length packet format is enabled then the number of bytes received as the payload is given by the *PayloadLength* parameter.

In variable length mode the first byte received after the sync word is interpreted as the length of the received packet. The internal length counter is initialized to this received length. The *PayloadLength* register is set to a value which is greater than the maximum expected length of the received packet. If the received length is greater than the maximum length stored in *PayloadLength* register the packet is discarded otherwise the complete packet is received.

If the address check is enabled then the second byte received in case of variable length and first byte in case of fixed length is the address byte. If the address matches to the one in the *NodeAddress* field, reception of the data continues otherwise it's stopped. The CRC check is performed if *CrcOn* = 1 and the result is available in *CrcOk* indicating that the CRC was successful. An interrupt (*PayloadReady*) is also generated on DIO0 as soon as the payload is available in the FIFO. The payload available in the FIFO can also be read in Sleep/Standby mode.

If the CRC fails the *PayloadReady* interrupt is not generated and the FIFO is cleared. This function can be overridden by setting *CrcAutoClearOff* = 1, forcing the availability of *PayloadReady* interrupt and the payload in the FIFO even if the CRC fails.

6.5.5 AES

AES is the symmetric-key block cipher that provides the cryptographic capabilities to the transceiver. The system proposed can work with 128-bit long fixed keys. The fixed key is stored in a 16-byte write only user configuration register, which retains its value in Sleep mode.

As shown in [Figure 6-10](#) and [Figure 6-11](#) above the message part of the Packet can be encrypted and decrypted with the cipher 128- cipher key stored in the configuration registers.

6.5.5.1 Tx Processing

1. User enters the data to be transmitted in FIFO in Stdby/Sleep mode and gives the transmit command.
2. On Tx command the Packet handler state machine takes over the control and If encryption is enabled then the message inside the FIFO is read in blocks of 16 bytes (padded with 0s if needed), encrypted and stored back to FIFO. All this processing is done in Tx mode before enabling the packet handling state machine. Only the Message part of the packet is encrypted and preamble, sync word, length byte, address byte and CRC are not encrypted.
3. Once the encryption is done the Packet handling state machine is enabled to transmit the data.

6.5.5.2 Rx Processing

1. The data received is stored in the FIFO, The address, CRC interrupts are generated as usual because these parameters were not encrypted.
2. Once the complete packet has been received. The data is read from the FIFO, decrypted and written back to FIFO. The *PayloadReady* interrupt is issued once the decrypted data is ready in the FIFO for reading via the SPI interface.

The AES encryption/decryption cannot be used on the fly i.e. while transmitting and receiving data. Thus when AES encryption/decryption is enabled, the FIFO acts as a simple buffer. This buffer is filled before initiating any transmission. The data in the buffer is then encrypted before the transmission can begin. On the receive side the decryption is initiated only once the complete packet has been received in the buffer.

The encryption/decryption process takes approximately 7.0 us per 16-byte block. Thus for a maximum of 4 blocks (i.e. 64 bytes) it can take up to 28 us for completing the cryptographic operations.

The receive side sees the AES decryption time as a sequential delay before the *PayloadReady* interrupt is available.

The Tx side sees the AES encryption time as a sequential delay in the startup of the Tx chain, thus the startup time of the Tx will increase according to the length of data.

In Fixed length mode the Message part of the payload that can be encrypted/decrypted can be 64 bytes long. If the address filtering is enabled, the length of the payload should be at max 65 bytes in this case.

In Variable length mode the Max message size that can be encrypted/decrypted is also 64 bytes when address filtering is disabled, else it is 48 bytes. Thus, including length byte, the length of the payload is max 65 or 50 bytes (the latter when address filtering is enabled).

If the address filtering is expected then *AddressFiltering* must be enabled on the transmitter side as well to prevent address byte to be encrypted.

Crc check being performed on encrypted data, *CrcOk* interrupt will occur "decryption time" before *PayloadReady* interrupt.

6.5.6 Handling Large Packets

When Payload length exceeds FIFO size (66 bytes) whether in fixed, variable or unlimited length packet format, in addition to *PacketSent* in Tx and *PayloadReady* or *CrcOk* in Rx, the FIFO interrupts/flags can be used as described below:

- For Tx:
 - FIFO can be prefilled in Sleep/Standby but must be refilled "on-the-fly" during Tx with the rest of the payload.
- 1. Prefill FIFO (in Sleep/Standby first or directly in Tx mode) until *FifoThreshold* or *FifoFull* is set
- 2. In Tx, wait for *FifoThreshold* or *FifoNotEmpty* to be cleared (i.e. FIFO is nearly empty)
- 3. Write bytes into the FIFO until *FifoThreshold* or *FifoFull* is set.
- 4. Continue to step 2 until the entire message has been written to the FIFO (*PacketSent* will fire when the last bit of the packet has been sent).
- For Rx:
 - FIFO must be unfilled "on-the-fly" during Rx to prevent FIFO overrun.
- 1. Start reading bytes from the FIFO when *FifoNotEmpty* or *FifoThreshold* becomes set.
- 2. Suspend reading from the FIFO if *FifoNotEmpty* clears before all bytes of the message have been read
- 3. Continue to step 1 until *PayloadReady* or *CrcOk* fires
- 4. Read all remaining bytes from the FIFO either in Rx or Sleep/Standby mode

NOTE

AES encryption is not feasible on large packets, since all Payload bytes need to be in the FIFO at the same time to perform encryption.

6.5.7 Packet Filtering

MC12311's packet handler offers several mechanisms for packet filtering, ensuring that only useful packets are made available to the uC, reducing significantly system power consumption and software complexity.

6.5.7.1 Sync Word Based

Sync word filtering/recognition is used for identifying the start of the payload and also for network identification. As previously described, the Sync word recognition block is configured (size, error tolerance, value) in *RegSyncValue* registers. This information is used, both for appending Sync word in Tx, and filtering packets in Rx.

Every received packet which does not start with this locally configured Sync word is automatically discarded and no interrupt is generated.

When the Sync word is detected, payload reception automatically starts and *SyncAddressMatch* is asserted.

NOTE

Sync Word values containing 0x00 byte(s) are forbidden.

6.5.7.2 Address Based

Address filtering can be enabled via the *AddressFiltering* bits. It adds another level of filtering, above Sync word (i.e. Sync must match first), typically useful in a multi-node networks where a network ID is shared between all nodes (Sync word) and each node has its own ID (address).

Two address based filtering options are available:

- *AddressFiltering* = 01: Received address field is compared with internal register *NodeAddress*. If they match then the packet is accepted and processed, otherwise it is discarded.
- *AddressFiltering* = 10: Received address field is compared with internal registers *NodeAddress* and *BroadcastAddress*. If either is a match, the received packet is accepted and processed, otherwise it is discarded. This additional check with a constant is useful for implementing broadcast in a multi-node networks

Please note that the received address byte, as part of the payload, is not stripped off the packet and is made available in the FIFO. In addition, *NodeAddress* and *AddressFiltering* only apply to Rx. On Tx side, if address filtering is expected, the address byte should simply be put into the FIFO like any other byte of the payload.

As address filtering requires a Sync word match, both features share the same interrupt flag *SyncAddressMatch*.

6.5.7.3 Length Based

In variable length Packet mode, *PayloadLength* must be programmed with the maximum payload length permitted. If received length byte is smaller than this maximum then the packet is accepted and processed, otherwise it is discarded.

Please note that the received length byte, as part of the payload, is not stripped off the packet and is made available in the FIFO.

To disable this function the user should set the value of the *PayloadLength* to 255.

6.5.7.4 CRC Based

The CRC check is enabled by setting bit *CrcOn* in *RegPacketConfig1*. It is used for checking the integrity of the message.

- On Tx side a two byte CRC checksum is calculated on the payload part of the packet and appended to the end of the message

- On Rx side the checksum is calculated on the received payload and compared with the two checksum bytes received. The result of the comparison is stored in bit *CrcOk*.

By default, if the CRC check fails then the FIFO is automatically cleared and no interrupt is generated. This filtering function can be disabled via *CrcAutoClearOff* bit and in this case, even if CRC fails, the FIFO is not cleared and only *PayloadReady* interrupt goes high. Please note that in both cases, the two CRC checksum bytes are stripped off by the packet handler and only the payload is made available in the FIFO.

The CRC is based on the CCITT polynomial as shown below. This implementation also detects errors due to leading and trailing zeros.

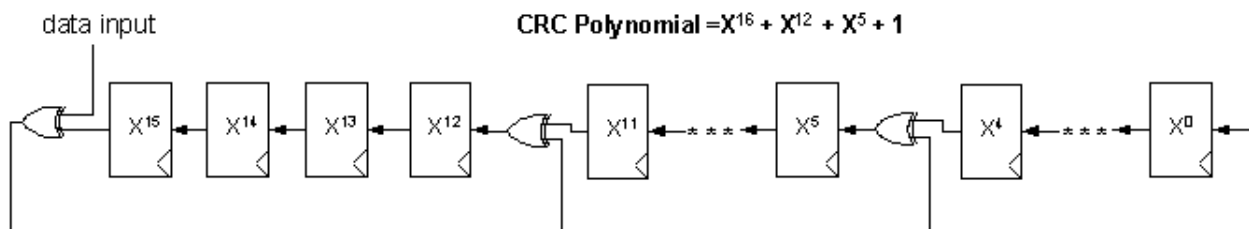


Figure 6-13. CRC Implementation

6.5.8 DC-Free Data Mechanisms

The payload to be transmitted may contain long sequences of 1's and 0's, which introduces a DC bias in the transmitted signal. The radio signal thus produced has a non uniform power distribution over the occupied channel bandwidth. It also introduces data dependencies in the normal operation of the demodulator. Thus it is useful if the transmitted data is random and DC free.

For such purposes, two techniques are made available in the packet handler: Manchester encoding and data whitening.

NOTE

Only one of the two methods should be enabled at a time.

6.5.8.1 Manchester Encoding

Manchester encoding/decoding is enabled if *DcFree* = 01 and can only be used in Packet mode.

The NRZ data is converted to Manchester code by coding '1' as "10" and '0' as "01".

In this case, the maximum chip rate is the maximum bit rate given in the specifications section and the actual bit rate is half the chip rate.

Manchester encoding and decoding is only applied to the payload and CRC checksum while preamble and Sync word are kept NRZ. However, the chip rate from preamble to CRC is the same and defined by *BitRate* in *RegBitRate* (Chip Rate = Bit Rate NRZ = 2 x Bit Rate Manchester).

Manchester encoding/decoding is thus made transparent for the user, who still provides/retrieves NRZ data to/from the FIFO.

	1/BR ... Sync								1/BR Payload...										
RF chips @ BR	...	1	1	1	0	1	0	0	1	0	0	1	0	1	1	0	1	0	...
User/NRZ bits Manchester OFF	...	1	1	1	0	1	0	0	1	0	0	1	0	1	1	0	1	0	...
User/NRZ bits Manchester ON	...	1	1	1	0	1	0	0	1	0	0	1	0	1	1	0	1	0	...

Figure 6-14. Manchester Encoding/Decoding

6.5.8.2 Data Whitening

Another technique called whitening or scrambling is widely used for randomizing the user data before radio transmission. The data is whitened using a random sequence on the Tx side and de-whitened on the Rx side using the same sequence. Comparing to Manchester technique it has the advantage of keeping NRZ data rate i.e. actual bit rate is not halved.

The whitening/de-whitening process is enabled if $DcFree = 10$. A 9-bit LFSR is used to generate a random sequence. The payload and 2-byte CRC checksum is then XORed with this random sequence as shown below. The data is de-whitened on the receiver side by XORing with the same random sequence.

Payload whitening/de-whitening is thus made transparent for the user, who still provides/retrieves NRZ data to/from the FIFO.

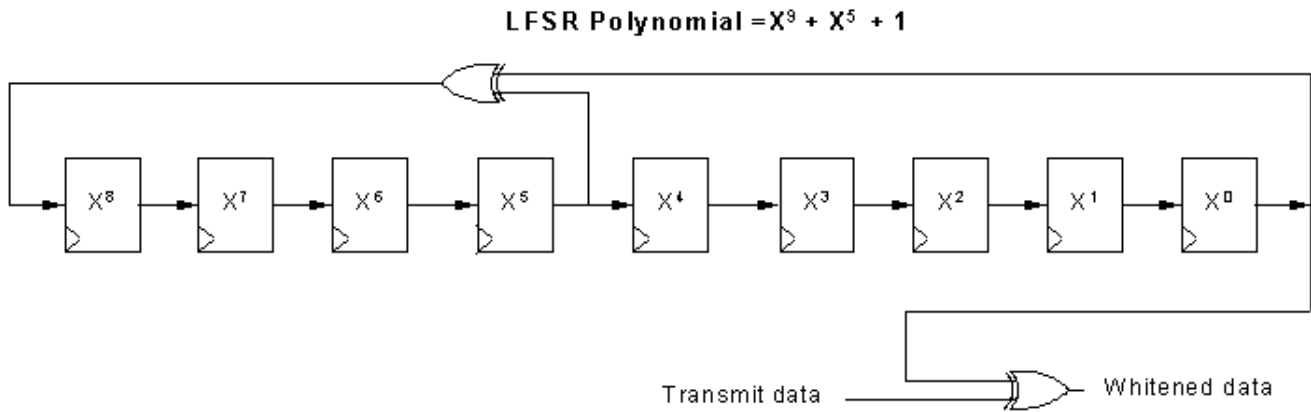


Figure 6-15. Data Whitening

6.6 Register Summary

Table 6-4. Registers Summary

Address	Register Name	Reset (built-in)	Default (recommended)	Description
0x00	RegFifo	0x00		FIFO read/write access
0x01	RegOpMode	0x04		Operating modes of the transceiver
0x02	RegDataModul	0x00		Data operation mode and Modulation settings
0x03	RegBitrateMsb	0x1A		Bit Rate setting, Most Significant Bits
0x04	RegBitrateLsb	0x0B		Bit Rate setting, Least Significant Bits

Table 6-4. Registers Summary

Address	Register Name	Reset (built-in)	Default (recommended)	Description
0x05	RegFdevMsb	0x00		Frequency Deviation setting, Most Significant Bits
0x06	RegFdevLsb	0x52		Frequency Deviation setting, Least Significant Bits
0x07	RegFrMsb	0xE4		RF Carrier Frequency, Most Significant Bits
0x08	RegFrMid	0xC0		RF Carrier Frequency, Intermediate Bits
0x09	RegFrLsb	0x00		RF Carrier Frequency, Least Significant Bits
0x0A	RegOsc1	0x41		RC Oscillators Settings
0x0B	RegAfcCtrl	0x00		AFC control in low modulation index situations
0x0C	RegLowBat	0x02		Low Battery Indicator Settings
0x0D	RegListen1	0x92		Listen Mode settings
0x0E	RegListen2	0xF5		Listen Mode Idle duration
0x0F	RegListen3	0x20		Listen Mode Rx duration
0x10	RegVersion	0x22		Semtech ID relating the silicon revision
0x11	RegPaLevel	0x9F		PA selection and Output Power control
0x12	RegPaRamp	0x09		Control of the PA ramp time in FSK mode
0x13	RegOcp	0x1A		Over Current Protection control
0x14	Reserved14	0x40		-
0x15	Reserved15	0xB0		-
0x16	Reserved16	0x7B		-
0x17	Reserved17	0x9B		-
0x18	RegLna	0x08	0x88	LNA settings
0x19	RegRxBw	0x86	0x55	Channel Filter BW Control
0x1A	RegAfcBw	0x8A	0x8B	Channel Filter BW control during the AFC routine
0x1B	RegOokPeak	0x40		OOK demodulator selection and control in peak mode
0x1C	RegOokAvg	0x80		Average threshold control of the OOK demodulator
0x1D	RegOokFix	0x06		Fixed threshold control of the OOK demodulator
0x1E	RegAfcFei	0x10		AFC and FEI control and status
0x1F	RegAfcMsb	0x00		MSB of the frequency correction of the AFC
0x20	RegAfcLsb	0x00		LSB of the frequency correction of the AFC
0x21	RegFeiMsb	0x00		MSB of the calculated frequency error
0x22	RegFeiLsb	0x00		LSB of the calculated frequency error
0x23	RegRssiConfig	0x02		RSSI-related settings

Table 6-4. Registers Summary

Address	Register Name	Reset (built-in)	Default (recommended)	Description
0x24	RegRssiValue	0xFF		RSSI value in dBm
0x25	RegDioMapping1	0x00		Mapping of pins DIO0 to DIO3
0x26	RegDioMapping2	0x05	0x07	Mapping of pins DIO4 and DIO5, ClkOut frequency
0x27	RegIrqFlags1	0x80		Status register: PLL Lock state, Timeout, RSSI > Threshold...
0x28	RegIrqFlags2	0x00		Status register: FIFO handling flags, Low Battery detection...
0x29	RegRssiThresh	0xFF	0xE4	RSSI Threshold control
0x2A	RegRxTimeout1	0x00		Timeout duration between Rx request and RSSI detection
0x2B	RegRxTimeout2	0x00		Timeout duration between RSSI detection and <i>PayloadReady</i>
0x2C	RegPreambleMsb	0x00		Preamble length, MSB
0x2D	RegPreambleLsb	0x03		Preamble length, LSB
0x2E	RegSyncConfig	0x98		Sync Word Recognition control
0x2F-0x36	RegSyncValue1-8	0x00	0x01	Sync Word bytes, 1 through 8
0x37	RegPacketConfig1	0x10		Packet mode settings
0x38	RegPayloadLength	0x40		Payload length setting
0x39	RegNodeAdrs	0x00		Node address
0x3A	RegBroadcastAdrs	0x00		Broadcast address
0x3B	RegAutoModes	0x00		Auto modes settings
0x3C	RegFifoThresh	0x0F	0x8F	Fifo threshold, Tx start condition
0x3D	RegPacketConfig2	0x02		Packet mode settings
0x3E-0x4D	RegAesKey1-16	0x00		16 bytes of the cypher key
0x4E	RegTemp1	0x01		Temperature Sensor control
0x4F	RegTemp2	0x00		Temperature readout
0x58	RegTestLna	0x1B		Sensitivity boost
0x71	RegTestAfc	0x00		AFC offset for low modulation index AFC
0x50 +	RegTest	-		Internal test registers

NOTE

Reset values are automatically refreshed in the chip at Power On Reset.

Default values are the recommended register values, optimizing the device operation.

Registers for which the Default value differs from the Reset value are denoted by a * in the appropriate tables.

6.7 Common Configuration Registers

Table 6-5. Common Configuration Registers

Name (Address)	Bits	Variable Name	Mode	Default Value	Description
RegFifo (0x00)	7-0	Fifo	rw	0x00	FIFO data input/output
RegOpMode (0x01)	7	SequencerOff	rw	0	Controls the automatic Sequencer: 0 → Operating mode as selected with Mode bits in RegOpMode is automatically reached with the Sequencer 1 → Mode is forced by the user
	6	ListenOn	rw	0	Enables Listen mode: 0 → Off 1 → On
	5	ListenAbort	w	0	Aborts Listen mode when set together with ListenOn=0 and new Mode selection in 1 SPI access Always reads 0.
	4-2	Mode	rw	001	Transceiver's operating modes: 000 → Sleep mode (SLEEP) 001 → Standby mode (STDBY) 010 → Frequency Synthesizer mode (FS) 011 → Transmitter mode (TX) 100 → Receiver mode (RX) others → reserved Reads the value corresponding to the current chip mode
	1-0	-	r	00	unused

Table 6-5. Common Configuration Registers

RegDataModul (0x02)	7	-	r	0	unused
	6-5	DataMode	rw	00	Data processing mode: 00 → Packet mode 01 → reserved 10 → Continuous mode with bit synchronizer 11 → Continuous mode without bit synchronizer
	4-3	ModulationType	rw	00	Modulation scheme: 00 → FSK 01 → OOK 10 - 11 → reserved
	2	-	r	0	unused
	1-0	ModulationShaping	rw	00	Data shaping: in FSK: 00 → no shaping 01 → Gaussian filter, BT = 1.0 10 → Gaussian filter, BT = 0.5 11 → Gaussian filter, BT = 0.3 in OOK: 00 → no shaping 01 → filtering with $f_{cutoff} = BR$ 10 → filtering with $f_{cutoff} = 2*BR$ 11 → reserved
RegBitrateMsb (0x03)	7-0	BitRate(15:8)	rw	0x1a	MSB of Bit Rate (Chip Rate when Manchester encoding is enabled)
RegBitrateLsb (0x04)	7-0	BitRate(7:0)	rw	0x0b	LSB of Bit Rate (Chip Rate if Manchester encoding is enabled) $\text{BitRate} = \frac{\text{FXOSC}}{\text{BitRate}(15,0)}$ Default value: 4.8 kb/s
RegFdevMsb (0x05)	7-6	-	r	00	unused
	5-0	Fdev(13:8)	rw	000000	MSB of the frequency deviation
RegFdevLsb (0x06)	7-0	Fdev(7:0)	rw	0x52	LSB of the frequency deviation $Fdev = Fstep \times Fdev(15,0)$ Default value: 5 kHz
RegFrfMsb (0x07)	7-0	Frf(23:16)	rw	0xe4	MSB of the RF carrier frequency
RegFrfMid (0x08)	7-0	Frf(15:8)	rw	0xc0	Middle byte of the RF carrier frequency
RegFrfLsb (0x09)	7-0	Frf(7:0)	rw	0x00	LSB of the RF carrier frequency $Frf = Fstep \times Frf(23;0)$ Default value: Frf = 915 MHz (32 MHz XO)

Table 6-5. Common Configuration Registers

RegOsc1 (0x0A)	7	RcCalStart	w	0	Triggers the calibration of the RC oscillator when set. Always reads 0. RC calibration must be triggered in Standby mode.
	6	RcCalDone	r	1	0 → RC calibration in progress 1 → RC calibration is over
	5-0	-	r	000001	unused
RegAfcCtrl (0x0B)	7-6	-	r	00	unused
	5	AfcLowBetaOn	rw	0	Improved AFC routine for signals with modulation index lower than 2. 0 → Standard AFC routine 1 → Improved AFC routine
	4-0	-	r	00000	unused
RegLowBat (0x0C)	7-5	-	r	000	unused
	4	LowBatMonitor	rw	-	Real-time (not latched) output of the Low Battery detector, when enabled.
	3	LowBatOn	rw	0	Low Battery detector enable signal 0 → LowBat off 1 → LowBat on
	2-0	LowBatTrim	rw	010	Trimming of the LowBat threshold: 000 → 1.695 V 001 → 1.764 V 010 → 1.835 V 011 → 1.905 V 100 → 1.976 V 101 → 2.045 V 110 → 2.116 V 111 → 2.185 V

Table 6-5. Common Configuration Registers

RegListen1 (0x0D)	7-6	ListenResolIdle	rw	10	Resolution of Listen mode Idle time (calibrated RC osc): 00 → reserved 01 → 64 us 10 → 4.1 ms 11 → 262 ms
	5-4	ListenResolRx	rw	01	Resolution of Listen mode Rx time (calibrated RC osc): 00 → reserved 01 → 64 us 10 → 4.1 ms 11 → 262 ms
	3	ListenCriteria	rw	0	Criteria for packet acceptance in Listen mode: 0 → signal strength is above <i>RssiThreshold</i> 1 → signal strength is above <i>RssiThreshold</i> and <i>SyncAddress</i> matched
	2-1	ListenEnd	rw	01	Action taken after acceptance of a packet in Listen mode: 00 → chip stays in Rx mode. Listen mode stops and must be disabled. 01 → chip stays in Rx mode until <i>PayloadReady</i> or <i>Timeout</i> interrupt occurs. It then goes to the mode defined by <i>Mode</i> . Listen mode stops and must be disabled. 10 → chip stays in Rx mode until <i>PayloadReady</i> or <i>Timeout</i> interrupt occurs. Listen mode then resumes in Idle state. FIFO content is lost at next Rx wakeup. 11 → Reserved
	0	-	r	0	unused
RegListen2 (0x0E)	7-0	ListenCoefIdle	rw	0xf5	Duration of the Idle phase in Listen mode.
RegListen3 (0x0F)	7-0	ListenCoefRx	rw	0x20	Duration of the Rx phase in Listen mode (startup time included)
RegVersion (0x10)	7-0	Version	r	0x23	Version code of the chip. Bits 7-4 give the full revision number; bits 3-0 give the metal mask revision number.

6.8 Transmitter Registers

Table 6-6. Transmitter Registers

Name (Address)	Bits	Variable Name	Mode	Default Value	Description
RegPaLevel (0x11)	7	Pa0On *	rw	1	Enables PA0, connected to RFIO and LNA
	6	Pa1On *	rw	0	Enables PA1, on PA_BOOST pin
	5	Pa2On *	rw	0	Enables PA2, on PA_BOOST pin
	4-0	OutputPower	rw	11111	Output power setting, with 1 dB steps Pout = -18 + <i>OutputPower</i> [dBm] , with PA0 or PA1 Pout = -14 + <i>OutputPower</i> [dBm] , with PA1 and PA2
RegPaRamp (0x12)	7-4	-	r	0000	unused
	3-0	PaRamp	rw	1001	Rise/Fall time of ramp up/down in FSK 0000 → 3.4 ms 0001 → 2 ms 0010 → 1 ms 0011 → 500 us 0100 → 250 us 0101 → 125 us 0110 → 100 us 0111 → 62 us 1000 → 50 us 1001 → 40 us 1010 → 31 us 1011 → 25 us 1100 → 20 us 1101 → 15 us 1110 → 12 us 1111 → 10 us
RegOcp (0x13)	7-5	-	r	000	unused
	4	OcpOn	rw	1	Enables overload current protection (OCP) for the PA: 0 → OCP disabled 1 → OCP enabled
	3-0	OcpTrim	rw	1010	Trimming of OCP current: $I_{max} = 45 + 5 \times OcpTrim(mA)$ 95 mA OCP by default

NOTE

Power Amplifier truth table is available in [Table 4-2](#).

6.9 Receiver Registers

Table 6-7. Receiver Registers

Name (Address)	Bits	Variable Name	Mode	Default Value	Description
Reserved14 (0x14)	7-0	-	r	0x40	unused
Reserved15 (0x15)	7-0	-	r	0xB0	unused
Reserved16 (0x16)	7-0	-	r	0x7B	unused
Reserved17 (0x17)	7-0	-	r	0x9B	unused
RegLna (0x18)	7	LnaZin	rw	1 *	LNA's input impedance 0 → 50 ohms 1 → 200 ohms
	6	-	r	0	unused
	5-3	LnaCurrentGain	r	001	Current LNA gain, set either manually, or by the AGC
	2-0	LnaGainSelect	rw	000	LNA gain setting: 000 → gain set by the internal AGC loop 001 → G1 = highest gain 010 → G2 = highest gain – 6 dB 011 → G3 = highest gain – 12 dB 100 → G4 = highest gain – 24 dB 101 → G5 = highest gain – 36 dB 110 → G6 = highest gain – 48 dB 111 → reserved
RegRxBw (0x19)	7-5	DccFreq	rw	010 *	Cut-off frequency of the DC offset canceller (DCC): $f_c = \frac{4 \times \text{RxBw}}{2\pi \times 2^{\text{DccFreq} + 2}}$ ~4% of the RxBw by default
	4-3	RxBwMant	rw	10 *	Channel filter bandwidth control: 00 → RxBwMant = 16 10 → RxBwMant = 24 01 → RxBwMant = 20 11 → reserved
	2-0	RxBwExp	rw	101 *	Channel filter bandwidth control: FSK Mode: $\text{RxBw} = \frac{\text{FXOSC}}{\text{RxBwMant} \times 2^{\text{RxBwExp} + 2}}$ OOK Mode: $\text{RxBw} = \frac{\text{FXOSC}}{\text{RxBwMant} \times 2^{\text{RxBwExp} + 3}}$

Table 6-7. Receiver Registers

RegAfcBw (0x1A)	7-5	DccFreqAfc	rw	100	DccFreq parameter used during the AFC
	4-3	RxBwMantAfc	rw	01	RxBwMant parameter used during the AFC
	2-0	RxBwExpAfc	rw	011 *	RxBwExp parameter used during the AFC
RegOokPeak (0x1B)	7-6	OokThreshType	rw	01	Selects type of threshold in the OOK data slicer: 00 → fixed 10 → average 01 → peak 11 → reserved
	5-3	OokPeakTheshStep	rw	000	Size of each decrement of the RSSI threshold in the OOK demodulator: 000 → 0.5 dB 001 → 1.0 dB 010 → 1.5 dB 011 → 2.0 dB 100 → 3.0 dB 101 → 4.0 dB 110 → 5.0 dB 111 → 6.0 dB
	2-0	OokPeakThreshDec	rw	000	Period of decrement of the RSSI threshold in the OOK demodulator: 000 → once per chip 001 → once every 2 chips 010 → once every 4 chips 011 → once every 8 chips 100 → twice in each chip 101 → 4 times in each chip 110 → 8 times in each chip 111 → 16 times in each chip
RegOokAvg (0x1C)	7-6	OokAverageThreshFilt	rw	10	Filter coefficients in average mode of the OOK demodulator: 00 → f_C ? chip rate / 32.? 01 → f_C ? chip rate / 8.? 10 → f_C ? chip rate / 4.? 11 → f_C ? chip rate / 2.?
	5-0	-	r	000000	unused
RegOokFix (0x1D)	7-0	OokFixedThresh	rw	0110 (6dB)	Fixed threshold value (in dB) in the OOK demodulator. Used when <i>OokThreshType</i> = 00

Table 6-7. Receiver Registers

RegAfcFei (0x1E)	7	-	r	0	unused
	6	FeiDone	r	0	0 → FEI is on-going 1 → FEI finished
	5	FeiStart	w	0	Triggers a FEI measurement when set. Always reads 0.
	4	AfcDone	r	1	0 → AFC is on-going 1 → AFC has finished
	3	AfcAutoclearOn	rw	0	Only valid if <i>AfcAutoOn</i> is set 0 → AFC register is not cleared before a new AFC phase 1 → AFC register is cleared before a new AFC phase
	2	AfcAutoOn	rw	0	0 → AFC is performed each time <i>AfcStart</i> is set 1 → AFC is performed each time Rx mode is entered
	1	AfcClear	w	0	Clears the <i>AfcValue</i> if set in Rx mode. Always reads 0
	0	AfcStart	w	0	Triggers an AFC when set. Always reads 0.
RegAfcMsb (0x1F)	7-0	AfcValue(15:8)	r	0x00	MSB of the <i>AfcValue</i> , 2's complement format
RegAfcLsb (0x20)	7-0	AfcValue(7:0)	r	0x00	LSB of the <i>AfcValue</i> , 2's complement format <i>Frequency correction</i> = <i>AfcValue</i> x <i>Fstep</i>
RegFeiMsb (0x21)	7-0	FeiValue(15:8)	r	-	MSB of the measured frequency offset, 2's complement
RegFeiLsb (0x22)	7-0	FeiValue(7:0)	r	-	LSB of the measured frequency offset, 2's complement <i>Frequency error</i> = <i>FeiValue</i> x <i>Fstep</i>
RegRssiConfig (0x23)	7-2	-	r	000000	unused
	1	RssiDone	r	1	0 → RSSI is on-going 1 → RSSI sampling is finished, result available
	0	RssiStart	w	0	Trigger a RSSI measurement when set. Always reads 0.
RegRssiValue (0x24)	7-0	RssiValue	r	0xFF	Absolute value of the RSSI in dBm, 0.5dB steps. $RSSI = -RssiValue/2 [dBm]$

6.10 IRQ and Pin Mapping Registers

Table 6-8. IRQ and Pin Mapping Registers

	Bits		Mode	Default Value	
RegDioMapping1 (0x25)	7-6	Dio0Mapping	rw	00	Mapping of pins DIO0 to DIO5 See Table 6-2 for mapping in Continuous mode See Table 6-3 for mapping in Packet mode
	5-4	Dio1Mapping	rw	00	
	3-2	Dio2Mapping	rw	00	
	1-0	Dio3Mapping	rw	00	
RegDioMapping2 (0x26)	7-6	Dio4Mapping	rw	00	unused
	5-4	Dio5Mapping	rw	00	
	3	-	r	0	
	2-0	ClkOut	rw	111 *	Selects CLKOUT frequency: 000 → FXOSC 001 → FXOSC / 2 010 → FXOSC / 4 011 → FXOSC / 8 100 → FXOSC / 16 101 → FXOSC / 32 110 → RC (automatically enabled) 111 → OFF

Table 6-8. IRQ and Pin Mapping Registers

RegIrqFlags1 (0x27)	7	ModeReady	r	1	Set when the operation mode requested in <i>Mode</i> , is ready - Sleep: Entering Sleep mode - Standby: XO is running - FS: PLL is locked - Rx: RSSI sampling starts - Tx: PA ramp-up completed Cleared when changing operating mode.
	6	RxReady	r	0	Set in Rx mode, after RSSI, AGC and AFC. Cleared when leaving Rx.
	5	TxReady	r	0	Set in Tx mode, after PA ramp-up. Cleared when leaving Tx.
	4	PllLock	r	0	Set (in FS, Rx or Tx) when the PLL is locked. Cleared when it is not.
	3	Rssi	rwc	0	Set in Rx when the <i>RssiValue</i> exceeds <i>RssiThreshold</i> . Cleared when leaving Rx.
	2	Timeout	r	0	Set when a timeout occurs (see <i>TimeoutRxStart</i> and <i>TimeoutRssiThresh</i>) Cleared when leaving Rx or FIFO is emptied.
	1	AutoMode	r	0	Set when entering Intermediate mode. Cleared when exiting Intermediate mode. Please note that in Sleep mode a small delay can be observed between <i>AutoMode</i> interrupt and the corresponding enter/exit condition.
	0	SyncAddressMatch	r/rwc	0	Set when Sync and Address (if enabled) are detected. Cleared when leaving Rx or FIFO is emptied. This bit is read only in Packet mode, rwc in Continuous mode

Table 6-8. IRQ and Pin Mapping Registers

RegIrqFlags2 (0x28)	7	FifoFull	r	0	Set when FIFO is full (i.e. contains 66 bytes), else cleared.
	6	FifoNotEmpty	r	0	Set when FIFO contains at least one byte, else cleared
	5	FifoLevel	r	0	Set when the number of bytes in the FIFO strictly exceeds <i>FifoThreshold</i> , else cleared.
	4	FifoOverrun	rwc	0	Set when FIFO overrun occurs. (except in Sleep mode) Flag(s) and FIFO are cleared when this bit is set. The FIFO then becomes immediately available for the next transmission / reception.
	3	PacketSent	r	0	Set in Tx when the complete packet has been sent. Cleared when exiting Tx.
	2	PayloadReady	r	0	Set in Rx when the payload is ready (i.e. last byte received and CRC, if enabled and <i>CrcAutoClearOff</i> is cleared, is Ok). Cleared when FIFO is empty.
	1	CrcOk	r	0	Set in Rx when the CRC of the payload is Ok. Cleared when FIFO is empty.
	0	LowBat	rwc	-	Set when the battery voltage drops below the Low Battery threshold. Cleared only when set by the user.
RegRssiThresh (0x29)	7-0	RssiThreshold	rw	0xE4 *	RSSI trigger level for <i>Rssi</i> interrupt : - $RssiThreshold / 2$ [dBm]
RegRxTimeout1 (0x2A)	7-0	TimeoutRxStart	rw	0x00	<i>Timeout</i> interrupt is generated $TimeoutRxStart * 16 * T_{bit}$ after switching to Rx mode if <i>Rssi</i> interrupt doesn't occur (i.e. $RssiValue > RssiThreshold$) 0x00: <i>TimeoutRxStart</i> is disabled
RegRxTimeout2 (0x2B)	7-0	TimeoutRssiThresh	rw	0x00	<i>Timeout</i> interrupt is generated $TimeoutRssiThresh * 16 * T_{bit}$ after <i>Rssi</i> interrupt if <i>PayloadReady</i> interrupt doesn't occur. 0x00: <i>TimeoutRssiThresh</i> is disabled

6.11 Packet Engine Registers

Table 6-9. Packet Engine Registers

	Bits		Mode	Default Value	
RegPreambleMsb (0x2c)	7-0	PreambleSize(15:8)	rw	0x00	Size of the preamble to be sent (from <i>TxStartCondition</i> fulfilled). (MSB byte)
RegPreambleLsb (0x2d)	7-0	PreambleSize(7:0)	rw	0x03	Size of the preamble to be sent (from <i>TxStartCondition</i> fulfilled). (LSB byte)
RegSyncConfig (0x2e)	7	SyncOn	rw	1	Enables the Sync word generation and detection: 0 → Off 1 → On
	6	FifoFillCondition	rw	0	FIFO filling condition: 0 → if <i>SyncAddress</i> interrupt occurs 1 → as long as <i>FifoFillCondition</i> is set
	5-3	SyncSize	rw	011	Size of the Sync word: (<i>SyncSize</i> + 1) bytes
	2-0	SyncTol	rw	000	Number of tolerated bit errors in Sync word
RegSyncValue1 (0x2f)	7-0	SyncValue(63:56)	rw	0x01 *	1 st byte of Sync word. (MSB byte) Used if <i>SyncOn</i> is set.
RegSyncValue2 (0x30)	7-0	SyncValue(55:48)	rw	0x01 *	2 nd byte of Sync word Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) >= 2.
RegSyncValue3 (0x31)	7-0	SyncValue(47:40)	rw	0x01 *	3 rd byte of Sync word. Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) >= 3.
RegSyncValue4 (0x32)	7-0	SyncValue(39:32)	rw	0x01 *	4 th byte of Sync word. Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) >= 4.
RegSyncValue5 (0x33)	7-0	SyncValue(31:24)	rw	0x01 *	5 th byte of Sync word. Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) >= 5.
RegSyncValue6 (0x34)	7-0	SyncValue(23:16)	rw	0x01 *	6 th byte of Sync word. Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) >= 6.
RegSyncValue7 (0x35)	7-0	SyncValue(15:8)	rw	0x01 *	7 th byte of Sync word. Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) >= 7.
RegSyncValue8 (0x36)	7-0	SyncValue(7:0)	rw	0x01 *	8 th byte of Sync word. Used if <i>SyncOn</i> is set and (<i>SyncSize</i> + 1) = 8.

Table 6-9. Packet Engine Registers

RegPacketConfig1 (0x37)	7	PacketFormat	rw	0	Defines the packet format used: 0 → Fixed length 1 → Variable length
	6-5	DcFree	rw	00	Defines DC-free encoding/decoding performed: 00 → None (Off) 01 → Manchester 10 → Whitening 11 → reserved
	4	CrcOn	rw	1	Enables CRC calculation/check (Tx/Rx): 0 → Off 1 → On
	3	CrcAutoClearOff	rw	0	Defines the behavior of the packet handler when CRC check fails: 0 → Clear FIFO and restart new packet reception. No <i>PayloadReady</i> interrupt issued. 1 → Do not clear FIFO. <i>PayloadReady</i> interrupt issued.
	2-1	AddressFiltering	rw	00	Defines address based filtering in Rx: 00 → None (Off) 01 → Address field must match <i>NodeAddress</i> 10 → Address field must match <i>NodeAddress</i> or <i>BroadcastAddress</i> 11 → reserved
	0	-	rw	0	unused
RegPayloadLength (0x38)	7-0	PayloadLength	rw	0x40	If PacketFormat = 0 (fixed), payload length. If PacketFormat = 1 (variable), max length in Rx, not used in Tx.
RegNodeAdrs (0x39)	7-0	NodeAddress	rw	0x00	Node address used in address filtering.
RegBroadcastAdrs (0x3A)	7-0	BroadcastAddress	rw	0x00	Broadcast address used in address filtering.

Table 6-9. Packet Engine Registers

RegAutoModes (0x3B)	7-5	EnterCondition	rw	000	Interrupt condition for entering the intermediate mode: 000 → None (AutoModes Off) 001 → Rising edge of <i>FifoNotEmpty</i> 010 → Rising edge of <i>FifoLevel</i> 011 → Rising edge of <i>CrcOk</i> 100 → Rising edge of <i>PayloadReady</i> 101 → Rising edge of <i>SyncAddress</i> 110 → Rising edge of <i>PacketSent</i> 111 → Falling edge of <i>FifoNotEmpty</i> (i.e. FIFO empty)
	4-2	ExitCondition	rw	000	Interrupt condition for exiting the intermediate mode: 000 → None (AutoModes Off) 001 → Falling edge of <i>FifoNotEmpty</i> (i.e. FIFO empty) 010 → Rising edge of <i>FifoLevel</i> or <i>Timeout</i> 011 → Rising edge of <i>CrcOk</i> or <i>Timeout</i> 100 → Rising edge of <i>PayloadReady</i> or <i>Timeout</i> 101 → Rising edge of <i>SyncAddress</i> or <i>Timeout</i> 110 → Rising edge of <i>PacketSent</i> 111 → Rising edge of <i>Timeout</i>
	1-0	IntermediateMode	rw	00	Intermediate mode: 00 → Sleep mode (SLEEP) 01 → Standby mode (STDBY) 10 → Receiver mode (RX) 11 → Transmitter mode (TX)
RegFifoThresh (0x3C)	7	TxStartCondition	rw	1 *	Defines the condition to start packet transmission : 0 → <i>FifoLevel</i> (i.e. the number of bytes in the FIFO exceeds <i>FifoThreshold</i>) 1 → <i>FifoNotEmpty</i> (i.e. at least one byte in the FIFO)
	6-0	FifoThreshold	rw	000111 1	Used to trigger <i>FifoLevel</i> interrupt.
RegPacketConfig2 (0x3D)	7-4	InterPacketRxDelay	rw	0000	After <i>PayloadReady</i> occurred, defines the delay between FIFO empty and the start of a new RSSI phase for next packet. Must match the transmitter's PA ramp-down time. - Tdelay = 0 if <i>InterpacketRxDelay</i> >= 12 - Tdelay = $(2^{\text{InterpacketRxDelay}}) / \text{BitRate}$ otherwise
	3	-	rw	0	unused
	2	RestartRx	w	0	Forces the Receiver in WAIT mode, in Continuous Rx mode. Always reads 0.
	1	AutoRxRestartOn	rw	1	Enables automatic Rx restart (RSSI phase) after <i>PayloadReady</i> occurred and packet has been completely read from FIFO: 0 → Off. <i>RestartRx</i> can be used. 1 → On. Rx automatically restarted after <i>InterPacketRxDelay</i> .
	0	AesOn	rw	0	Enable the AES encryption/decryption: 0 → Off 1 → On (payload limited to 66 bytes maximum)
RegAesKey1 (0x3E)	7-0	AesKey(127:120)	w	0x00	1 st byte of cipher key (MSB byte)

Table 6-9. Packet Engine Registers

RegAesKey2 (0x3F)	7-0	AesKey(119:112)	w	0x00	2 nd byte of cipher key
RegAesKey3 (0x40)	7-0	AesKey(111:104)	w	0x00	3 rd byte of cipher key
RegAesKey4 (0x41)	7-0	AesKey(103:96)	w	0x00	4 th byte of cipher key
RegAesKey5 (0x42)	7-0	AesKey(95:88)	w	0x00	5 th byte of cipher key
RegAesKey6 (0x43)	7-0	AesKey(87:80)	w	0x00	6 th byte of cipher key
RegAesKey7 (0x44)	7-0	AesKey(79:72)	w	0x00	7 th byte of cipher key
RegAesKey8 (0x45)	7-0	AesKey(71:64)	w	0x00	8 th byte of cipher key
RegAesKey9 (0x46)	7-0	AesKey(63:56)	w	0x00	9 th byte of cipher key
RegAesKey10 (0x47)	7-0	AesKey(55:48)	w	0x00	10 th byte of cipher key
RegAesKey11 (0x48)	7-0	AesKey(47:40)	w	0x00	11 th byte of cipher key
RegAesKey12 (0x49)	7-0	AesKey(39:32)	w	0x00	12 th byte of cipher key
RegAesKey13 (0x4A)	7-0	AesKey(31:24)	w	0x00	13 th byte of cipher key
RegAesKey14 (0x4B)	7-0	AesKey(23:16)	w	0x00	14 th byte of cipher key
RegAesKey15 (0x4C)	7-0	AesKey(15:8)	w	0x00	15 th byte of cipher key
RegAesKey16 (0x4D)	7-0	AesKey(7:0)	w	0x00	16 th byte of cipher key (LSB byte)

6.12 Temperature Sensor Registers

Table 6-10. Temperature Sensor Registers

Name (Address)	Bits	Variable Name	Mode	Default Value	Description
RegTemp1 (0x4E)	7-4	-	r	0000	unused
	3	TempMeasStart	w	0	Triggers the temperature measurement when set. Always reads 0.
	2	TempMeasRunning	r	0	Set to 1 while the temperature measurement is running. Toggles back to 0 when the measurement has completed. The receiver can not be used while measuring temperature
	1-0	-	r	01	unused
RegTemp2 (0x4F)	7-0	TempValue	r	-	Measured temperature -1°C per Lsb Needs calibration for accuracy

6.13 Test Registers

Table 6-11. Test Registers

Name (Address)	Bits	Variable Name	Mode	Default Value	Description
RegTestLna (0x58)	7-0	SensitivityBoost	rw	0x1B	High sensitivity or normal sensitivity mode: 0x1B → Normal mode 0x2D → High sensitivity mode
RegTestDagc (0x6F)	7-0	ContinuousDagc	rw	0x30	Fading Margin Improvement, refer to Section 4.7.3 , “Continuous-Time DAGC” 0x00 → Normal mode 0x10 → Improved margin, use if <i>AfcLowBetaAfcOn</i> = 1 0x30 → Improved margin, use if <i>AfcLowBetaAfcOn</i> = 0
RegTestAfc (0x71)	7-0	LowBetaAfcOffset	rw	0x00	AFC offset set for low modulation index systems, used if <i>AfcLowBetaOn</i> =1. <i>Offset</i> = <i>LowBetaAfcOffset</i> x 488 Hz

Chapter 7

MC12311 Transceiver - MCU SPI Interface

The MC12311 transceiver and CPU communicate primarily through the onboard SPI interface. The MCU has a single SPI module that is dedicated to the transceiver SPI interface. The transceiver is a SPI slave only, and the MCU SPI module must be programmed and used as a master only.

7.1 SiP Level SPI Pin Connections

The SiP level SPI pin connections are all internal to the device. [Figure 7-1](#) shows all the SiP interconnections with the SPI bus highlighted.

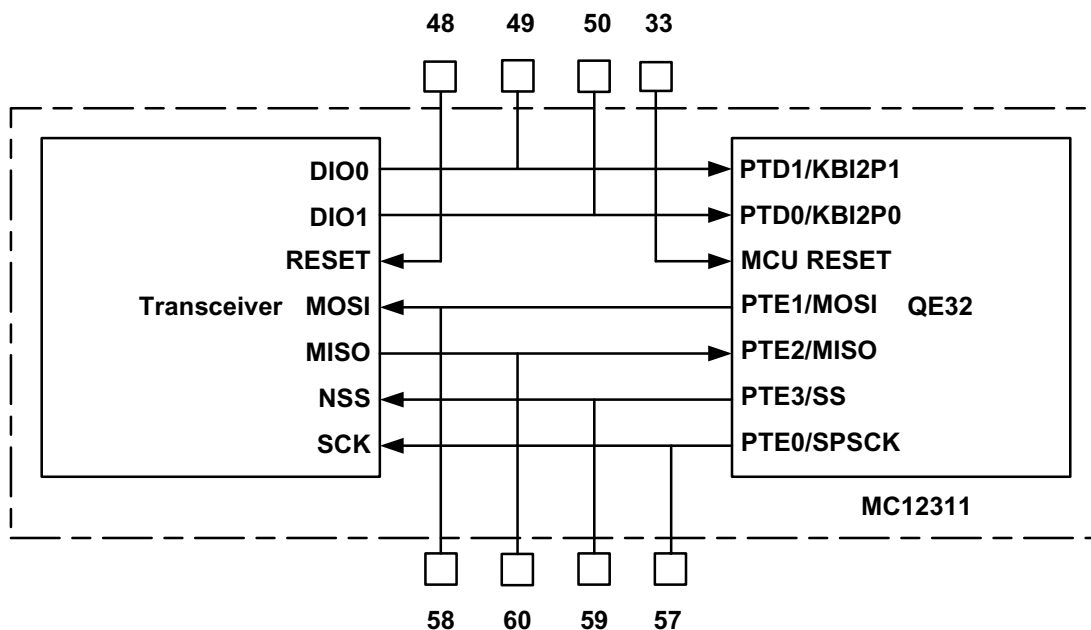


Figure 7-1. MC12311 Internal Interconnects Highlighting SPI Bus

Table 2-2 in Chapter 2 provides a complete listing of the MC12311 onboard device interconnects.

NOTE

- The MCU SPI port does not default to Port E connections, and these connections **MUST** be programmed for proper device communication. See [Section 2.3, “Internal Functional Interconnects”](#).
- The SPI external access Pins 57-60 are located on the bottom of the package.

7.2 Features

Features of the SPI bus interface:

- MCU is the SPI Bus master
- Transceiver is bus slave
- Bi-directional data transfer
- Dedicated interface; must meet transceiver protocol requirements
- Programmable SPI clock rate; maximum transfer rate is 10 MHz as determined by the transceiver
- Double-buffered transmit and receive at MCU
- Serial clock phase and polarity must meet transceiver requirements (MCU control bits CPHA = 0 and CPOL = 0)
- Slave select programmed to meet transceiver protocol
- MSB-first shifting

7.3 SPI System Block Diagram

This section shows the system level diagram for the SPI. [Figure 7-2](#) shows the SPI modules of the MCU and transceiver in the master-slave arrangement.

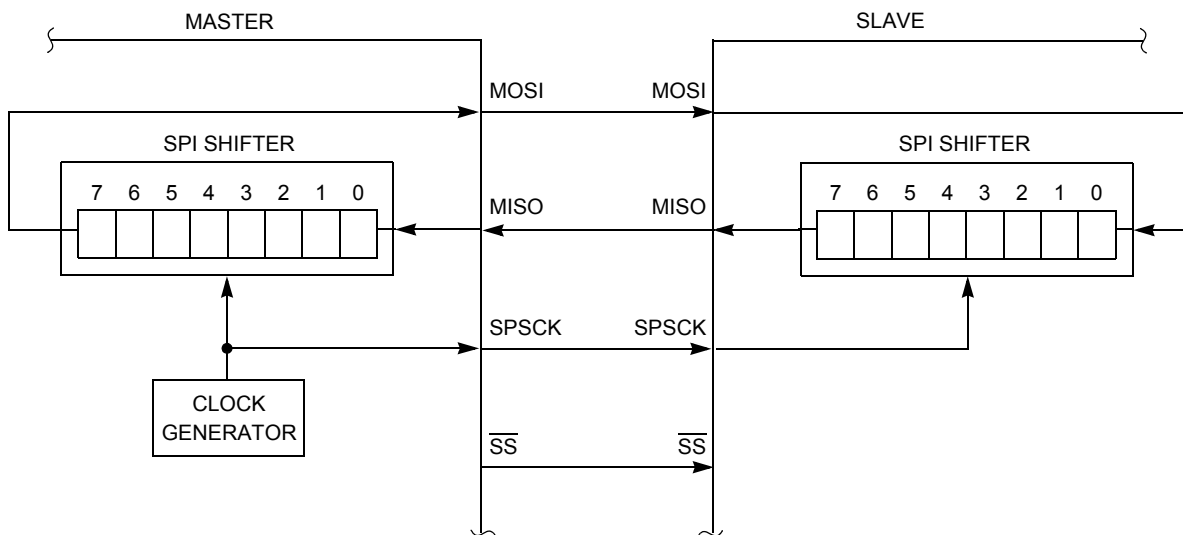


Figure 7-2. SPI System Block Diagram

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 transceiver SPI protocol only uses data exchange in one direction at a time. The SPCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input (\overline{SS} pin).

7.3.1 SPI Signal Definitions

The SPI signals of \overline{SS} , SCK, MOSI, and MISO are defined in the following paragraphs.

7.3.1.1 Slave Select (\overline{SS} or NSS)

A transaction on the SPI port is framed by the active low Slave Select (\overline{SS}) input signal which is driven by the MCU as master.

7.3.1.2 SPI Clock (SCK or SPSCK)

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

NOTE

- The SPI Bus protocol as defined by the Motorola/Freescale standard supports other clock/data timing, however, the described mode is the only one used.
- For the MC12311 microcontroller, the SPI clock format is the clock phase control bit CPHA = 0 and the clock polarity control bit CPOL = 0.

7.3.1.3 Master Out / Slave In (MOSI)

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

7.3.1.4 Master In / Slave Out (MISO)

The Master In/Slave Out (MISO) signal presents incoming data from the transceiver to the MCU (master input).

7.3.2 MC12311 SPI Transaction Protocol

Although standard SPI protocol is based on 8-bit transfers (see [Section Figure 7-6., “MCU SPI Clock Format \(CPHA = 0 and CPOL = 0\)”](#)), the transceiver imposes a higher level transaction protocol that is based on multiple 8-bit transfers per transaction. There are three SPI transaction modes defined to access the transceiver registers:

- SINGLE access - an address byte is followed by a data byte
 - For a write access, the data byte (MOSI) is written to the addressed transceiver register
 - For a read access, the data byte (MISO) is read from the addressed transceiver register
 - The NSS pin goes low at the beginning of the frame and goes high after the data byte
- BURST access - an address byte is followed by several data bytes.
 - The address byte provides the starting register address for the data burst
 - The address is automatically incremented internally between each data byte.

- This mode supports both read and write accesses.
- The NSS pin goes low at the beginning of the frame and stays low between bytes. It goes high only after the last byte transfer.
- FIFO access - special case of burst access for the FIFO
 - The address byte corresponds to the address of the FIFO
 - The address is not automatically incremented, but stays pointed at the FIFO.
 - The data bytes are sequentially written to or read from the FIFO.
 - The NSS pin goes low at the beginning of the frame and stays low between bytes. It goes high only after the last byte transfer

For the defined transaction formats:

- The falling edge of NSS always initializes the start of the frame transfer.
- All address and data sent MSB first
- The address byte (first byte) of the transaction is composed of -
 - WNR Bit (Bit 7) - which is “1” for write access and “0” for read access
 - Address (Bit 6:0) - 7-bit register address (sent MSB first)
- Following data byte(s) -
 - Write data sent on MOSI
 - Read data sent on MISO
- The rising edge of NSS signifies the end of the frame transfer (NSS must remain asserted for the entire frame).

7.3.3 MC12311 SPI Transaction Timing

As defined in Section 7.3.2, the SPI transaction protocol is composed of two or more bytes per frame. Although the transceiver is capable of a continuous bit transfer for the entire frame, the MCU SPI port only transfers data in bursts of 8 bits. There are implications in the way the MCU SPI is programmed and used:

- The MCU SS signal -
 - Cannot be programmed for SPI module master mode driven operation
 - Port signal PTE3 must be enabled and programmed as a GPIO output to provide the required SS signal timing
- The transaction bytes are sent as a bursts as allowed by the MCU 8-bit SPI module.

Because the MCU is embedded in the SiP and the transceiver only supports the 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.

Figure 7-3 illustrates a simple single read access transaction timing. The top part of the figure shows the SPI timing for a single byte transfer. For the complete frame, the SS signal goes low and stays low for both byte transfers. The byte transfers are actually accomplished as two operations for the MCI SPI peripheral block.

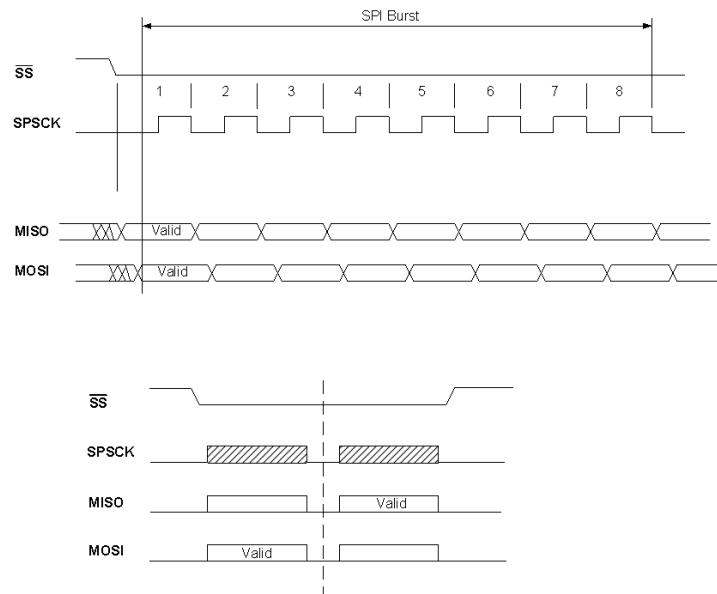


Figure 7-3. Transceiver SPI Read Single Access Timing Diagram

The SPI Bus timing is determined by the transceiver specification as given in Table 7-1.

Table 7-1. Transceiver SPI Timing Specifications

Parameter	Min	Typ	Max	Unit
SPSCCK period	100			ns
Pulse width, SPSCCK low	50			ns
Pulse width, SPSCCK high	50			ns
Setup time, \overline{SS} low to rising SPSCCK	30			ns
Hold time, from falling SPSCCK to \overline{SS} rising edge	60			ns
Setup time, MOSI valid to rising SPSCCK	30			ns
Hold time, MOSI valid from rising SPSCCK	60			ns
SS high time between accesses	20			ns

NOTE

For the onboard QE32 MCU, the bus clock to the SPI module is always $\frac{1}{2}$ the CPU clock, and in turn, the maximum SPI clock baud rate is $\frac{1}{2}$ the peripheral clock. The default bus clock rate for Freescale supplied software is 32 MHz which in turn provides an 8 Mhz SPI baud rate.

7.4 QE32 MCU SPI Module Overview

The SPI module on the QE32 is standard peripheral function. In the case of the MC12311, however, the SPI is dedicated to communication with the onboard transceiver.

- The module must be programmed for Master Mode only, MSB-first shifting, and full-duplex operation
- The SPI signals default to Port B and must be reprogrammed to Port E
- The SS signal must be programmed a Port E GPIO (output)
- SPI port input filters must be disabled

7.5 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.

7.5.1 MCU SPI Module Block Diagram

Figure 7-4 is a block diagram of the SPI module.

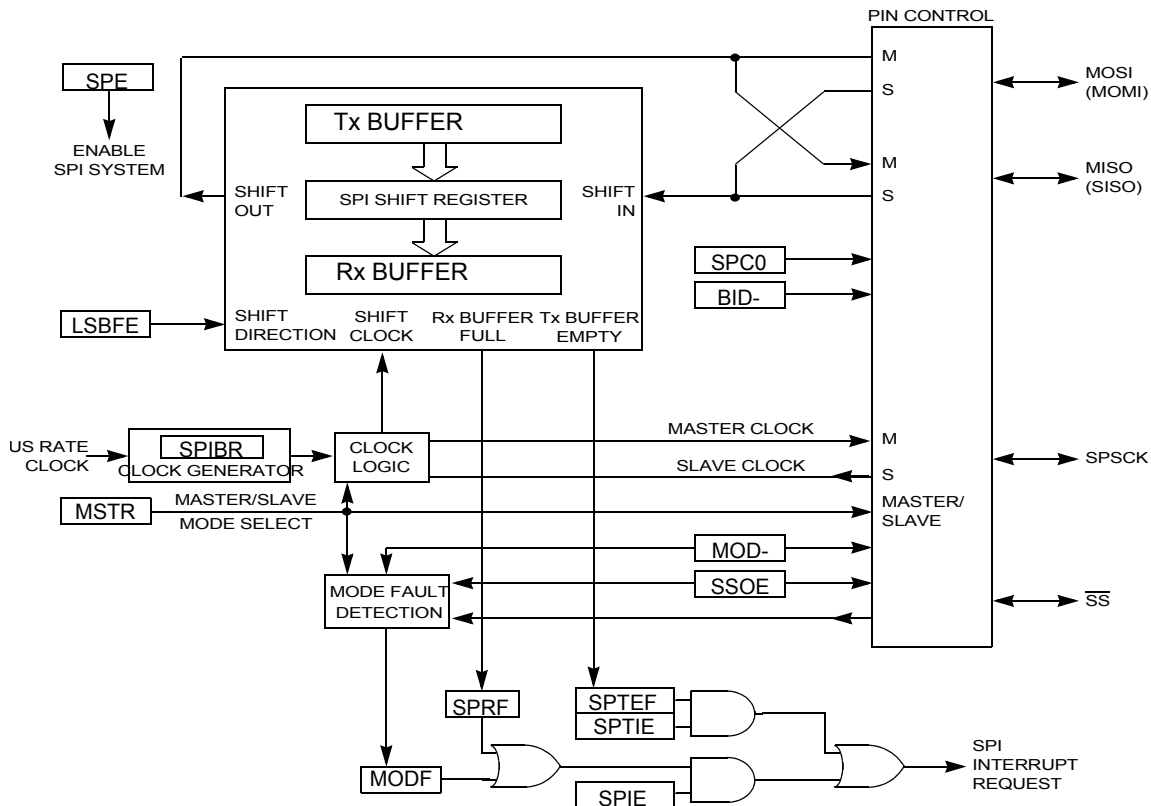


Figure 7-4. MCU SPI Module Block Diagram

The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter 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. Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

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

7.5.2 MCU SPI Baud Rate Generation

As shown in [Figure 7-5](#), 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.

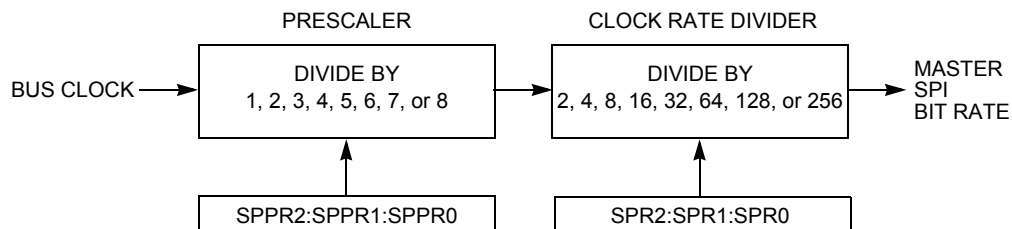


Figure 7-5. MCU SPI Baud Rate Generation

7.6 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 MISO pin at one SPSCCK edge and shifted, changing the bit value on the MOSI pin, one-half SPSCCK cycle later. After eight SPSCCK cycles, the data that was in the shift register of the master has been shifted out the MOSI pin to the slave while eight bits of data were shifted in the MISO pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading 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 MC12311.

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.

7.6.1 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.

NOTE

The MC12311 requires use of the CPOL = 0 and CPHA = 0 format

Figure 7-6 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 transceiver protocol as described in Section 7.3.2, “MC12311 SPI Transaction Protocol” requires that the \overline{SS} stay low for an entire multibyte transaction.

7.6.2 SPI Clock Gating

The bus clock to the SPI can be gated on and off using the SPI bit in SCGC2. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use.

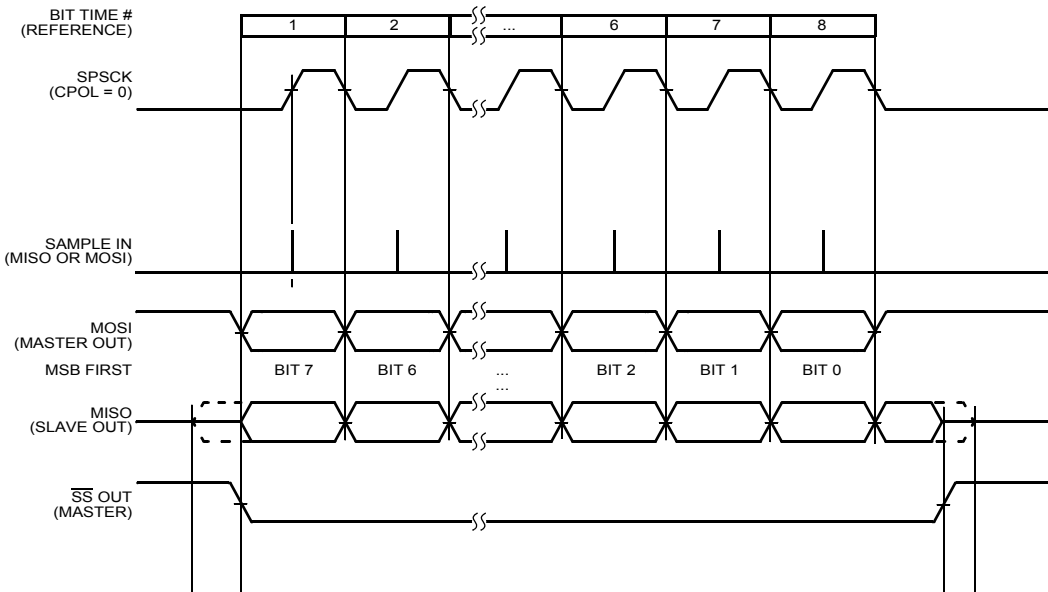


Figure 7-6. MCU SPI Clock Format (CPHA = 0 and CPOL = 0)

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.

7.6.3 MCU SPI Pin Controls

The SPI port pins are shared with GPIO port pins. They must be configured properly for use with the transceiver.

7.6.3.1 Assigning Port Pins

When the SPI is enabled, the SPI signals default to Port B pin assignments. For proper operation with the transceiver they must be over-programmed to Port E pin assignments as shown in Table 7-2. This is done by setting the SPIPS bit in the SOPT2 Register.

Table 7-2. SPI Position Options

SPIPS in SOPT2	Port Pin for SPSCLK	Port Pin for MOSI	Port Pin for MISO	Port Pin for \overline{SS}
0 (default)	PTB2	PTB3	PTB4	PTB5
1	PTE0	PTE1	PTE2	PTE3

7.6.3.2 Configuring \overline{SS} /PTE3 for Proper Operation

Due to required multi-byte transfers for each transceiver transaction, the SS signal must be controlled as a peripheral pin, i.e., PTE3. See Section 7.8.1.3, “GPIO PTE3 Control (Used as SS)” for details.

7.6.3.3 Disabling SPI input Filters

To configure the SPI for high speed operation (operating frequency of greater than 5 MHz), the input filters on the SPI port pins must be disabled by clearing the SPIxFE in SOPT2 and enabling the high output drive strength on the port pins corresponding to the SPI function pins. Doing so allows data transfers to occur at higher frequency. This is required for typical operation of 8 MHz SPI baudrate.

By default, the input filters on the SPI port pins will be enabled (SPIxFE=1), which restricts the SPI data rate to 5 MHz, but protects the SPI from noise during data transfers.

7.6.4 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).

7.6.5 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the \overline{SS} pin (provided the \overline{SS} pin is configured as the mode fault input signal). The \overline{SS} pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's \overline{SS} pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to Slave Mode. The output drivers on the SPSCCK, MOSI, and MISO (if not Bidirectional Mode) are disabled.

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

7.7 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 9, “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.

7.7.1 SPI Control Register 1 (SPI1C1)

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

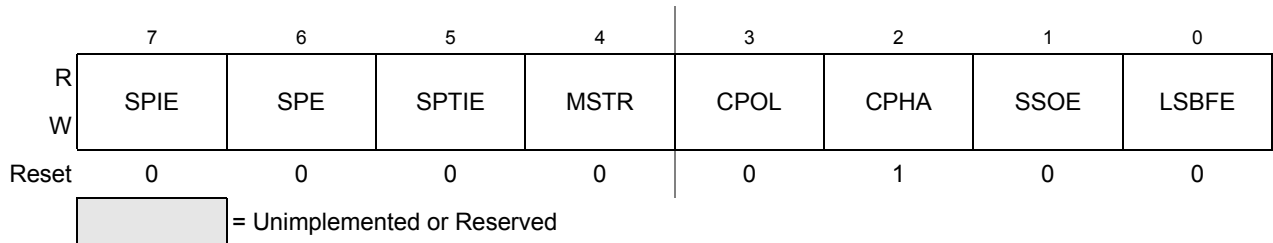


Figure 7-7. SPI Control Register 1 (SPI1C1)

Table 7-3. SPI1C1 Field Descriptions

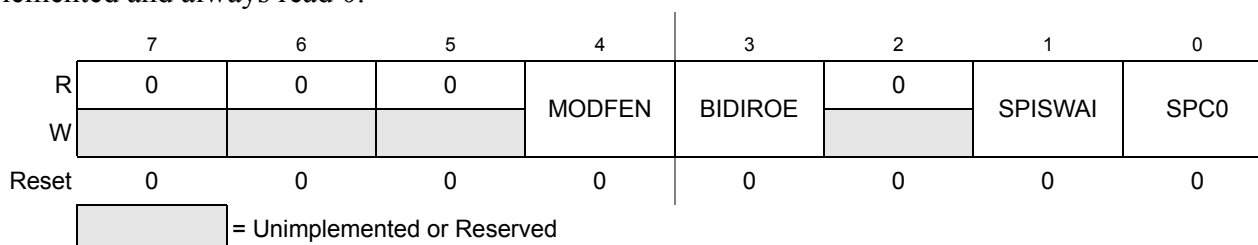
Field	Description
7 SPIE	SPI Interrupt Enable (for SPRF and MODF) — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling). 1 When SPRF or MODF is 1, request a hardware interrupt.
6 SPE	SPI System Enable — 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	SPI Transmit Interrupt Enable — 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	Master/Slave Mode Select 0 SPI module configured as a slave SPI device. 1 SPI module configured as a master SPI device.
3 CPOL	Clock Polarity — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 7.6.1, “SPI Clock Formats” , for more details. 0 Active-high SPI clock (idles low). 1 Active-low SPI clock (idles high).
2 CPHA	Clock Phase — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 7.6.1, “SPI Clock Formats” , for more details. 0 First edge on SPSCCK occurs at the middle of the first cycle of an 8-cycle data transfer. 1 First edge on SPSCCK occurs at the start of the first cycle of an 8-cycle data transfer.
1 SSOE	Slave Select Output Enable — This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the \overline{SS} pin as shown in Table 7-4 .
0 LSBFE	LSB First (Shifter Direction) 0 SPI serial data transfers start with most significant bit. 1 SPI serial data transfers start with least significant bit.

Table 7-4. \overline{SS} Pin Function

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	\overline{SS} input for mode fault	Slave select input
1	1	Automatic \overline{SS} output	Slave select input

7.7.2 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 7-8. SPI Control Register 2 (SPI1C2)
Table 7-5. SPI1C2 Field Descriptions

Field	Description
4 MODFEN	Master Mode-Fault Function Enable — When the SPI is configured for Slave Mode, this bit has no meaning or effect. (The \overline{SS} pin is the slave select input.) In Master Mode, this bit determines how the \overline{SS} pin is used (refer to Table 7-4 for more details). 0 Mode fault function disabled, master \overline{SS} pin reverts to general-purpose I/O not controlled by SPI. 1 Mode fault function enabled, master \overline{SS} pin acts as the mode fault input or the slave select output.
3 BIDIROE	Bidirectional Mode Output Enable — When Bidirectional Mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input. 1 SPI I/O pin enabled as an output.
1 SPISWAI	SPI Stop in Wait Mode 0 SPI clocks continue to operate in Wait Mode. 1 SPI clocks stop when the MCU enters Wait Mode.
0 SPC0	SPI Pin Control 0 — The SPC0 bit chooses Single-wire Bidirectional Mode. If MSTR = 0 (Slave Mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (Master Mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output. 1 SPI configured for single-wire bidirectional operation.

7.7.3 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.

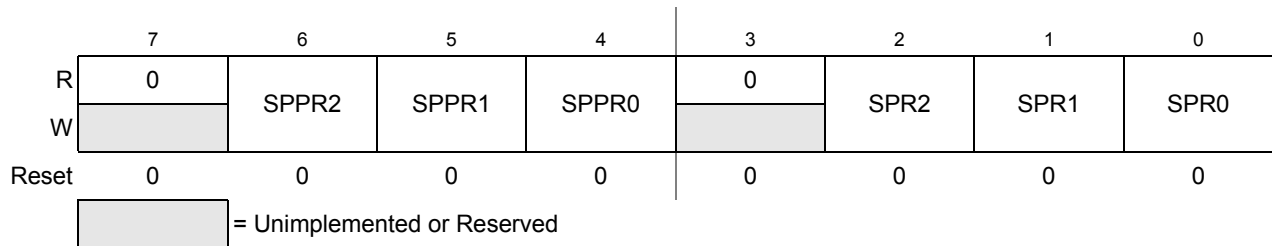


Figure 7-9. SPI Baud Rate Register (SPI1BR)

Table 7-6. SPI1BR Field Descriptions

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

Table 7-7. 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 7-8. 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

7.7.4 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.

	7	6	5	4	3	2	1	0
R	SPRF	0	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-10. SPI Status Register (SPI1S)

Table 7-9. SPI1S Field Descriptions

Field	Description
7 SPRF	<p>SPI Read Buffer Full Flag — 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>SPI Transmit Buffer Empty Flag — 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 shifter, 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>Master Mode Fault Flag — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The \overline{SS} pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPI1C1).</p> <p>0 No mode fault error. 1 Mode fault error detected.</p>

7.7.5 SPI Data Register (SPI1D)

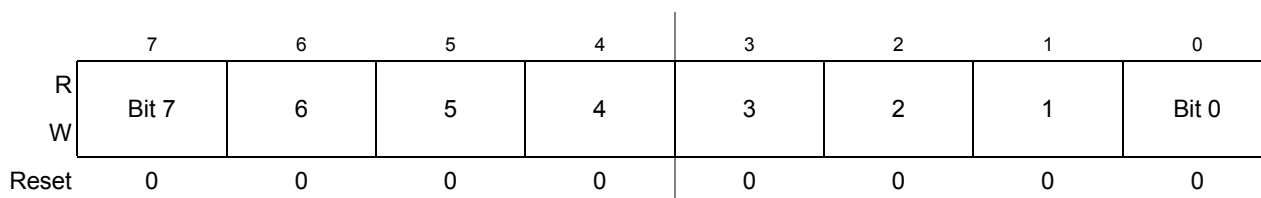


Figure 7-11. SPI Data Register (SPI1D)

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.

7.8 Configuring MCU Registers for Proper SPI Operation

The MCU SPI module must be configured for proper operation to meet the transceiver 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{SS} must be controlled as a GPIO pin, i.e., PTE2 to meet the protocol

7.8.1 Set SPI Module Mode

7.8.1.1 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{SS} 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)

7.8.1.2 SPI1C2 Register settings

The following register settings apply:

- MODFEN (Bit 4) = 0 - \overline{SS} reverts to GPIO controlled as PTE3
- 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)

7.8.1.3 GPIO PTE3 Control (Used as \overline{SS})

The PTE2/ \overline{SS} is used to control to the transceiver \overline{CE} input, however, the MCU SPI module does not support use of this signal in the manner required by the transceiver 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.

7.8.2 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 transceiver and lengthen the initialization time of the transceiver.

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 8

MCU Modes of Operation

8.1 Introduction

The operating modes of the 9S08QE32 MCU are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

8.2 Features

- Active background mode for code development
- Run mode — CPU clocks can be run at full speed and the internal supply is fully regulated.
- LPrun mode — CPU and peripheral clocks are restricted to 125 kHz maximum and the internal voltage regulator is in standby
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained
- LPwait mode — CPU shuts down to conserve power; peripheral clocks are restricted to 125 kHz maximum and the internal voltage regulator is in standby
- Stop modes — System clocks are stopped and voltage regulator is in standby
 - Stop3 — All internal circuits are powered for fast recovery
 - Stop2 — Partial power down of internal circuits, RAM content is retained, I/O states held

8.3 Run Mode

This is the normal operating mode for the 9S08QE32 series. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE–0xFFFF after reset.

8.3.1 Low Power Run Mode (LPRun)

In the low-power run mode, the on-chip voltage regulator is put into its standby state. In this state, the power consumption is reduced to a minimum that still allows CPU functionality. Power consumption is reduced the most by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC1 and SCGC2 registers.

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

- FBELP is the selected clock mode for the ICS.
- The HGO bit in the ICSC2 register is clear.
- The bus frequency is less than 125 kHz.

- The ADC, if enabled, must be configured to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements. The bandgap channel cannot be converted in low-power run mode.
- The LVDE or LVDSE bit in SPMSC1 register must be clear. LVD and LVW automatically are disabled.
- Flash programming/erasing is not allowed.
- ACMP option to compare to internal bandgap reference is not allowed in LPrun and LPwait.
- The MCU cannot be in active background mode.

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

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

8.3.1.1 Interrupts in Low Power Run Mode

Low power run mode provides the option to return to full regulation if any interrupt occurs. To do this, set the LPWUI bit in the SPMSC2 register. The ICS can then be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is clear, interrupts are serviced in low-power run mode.

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

8.3.1.2 Resets in Low Power Run Mode

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

8.3.1.3 BDM in Low Power Run Mode

Low power run mode cannot be entered when the MCU is in active background debug mode.

If a device is in low power run mode, a falling edge on an active BKGD/MS pin exits low power run mode, clears the LPR and LPRS bits and returns the device to normal run mode.

8.4 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the QE32 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 six ways:

- When the BKGD/MS pin is low during POR
- When the BKGD/MS pin is low immediately after issuing a background debug force reset

- When a BACKGROUND command is received through the BKGD/MS pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

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

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
 - Memory access commands
 - Memory-access-with-status commands
 - BDC register access commands
 - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
 - Read or write CPU registers
 - Trace one user program instruction at a time
 - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the flash program memory before the MCU is operated in run mode for the first time. When the 9S08QE32 series are 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.

8.5 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 stop or wait mode. The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

8.5.1 Low-Power Wait Mode (LPWait)

Low-power wait mode is entered by executing a wait instruction while the MCU is in low-power run mode. In the low-power wait mode, the on-chip voltage regulator remains in its standby state as in the low-power run mode. In this state, the power consumption is reduced to a minimum that still allows most modules to maintain functionality. Power consumption is reduced the most by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC register.

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

8.5.1.1 Interrupts in Low Power Wait Mode

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

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

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

8.5.1.2 Resets in Low Power Wait Mode

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

8.5.1.3 BDM in Low Power Wait Mode

If a device is in low power wait mode, a falling edge on an active BKGD/MS pin exits low power wait mode, clears the LPR and LPRS bits and returns the device to normal run mode.

8.6 Stop Modes

One of two stop modes (stop2 or stop3) is entered upon execution of a STOP instruction when the STOPE bit in the system option 1 register (SOPT1) is set. In both stop modes, the bus and CPU clocks are halted. In stop3 the regulator is in standby. In stop2 the regulator is in partial powerdown. The ICS module can be configured to leave the reference clocks running.

If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU does not enter either stop mode, and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in the system power management status and control 2 register (SPMSC2).

Table 8-1 shows all of the control bits that affect stop mode selection and the mode selected under various conditions. The selected mode is entered following the execution of a STOP instruction.

Table 8-1. Stop Mode Selection

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

¹ ENBDM is located in the BDCSCR which is accessible only through BDC commands, see the Development Support Chapter for more information.

² When in stop3 mode with BDM enabled, The S_{IDD} is near R_{IDD} levels because internal clocks are enabled.

8.6.1 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions (Table 8-1). Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM and optionally the RTC and low power oscillator. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting the wakeup pin (PTA5/IRQ/TPM1CLK/ $\overline{\text{RESET}}$) on the MCU.

NOTE

PTA5/IRQ/TPM1CLK/ $\overline{\text{RESET}}$ functions is an active low wakeup and must be configured as an input prior to executing a STOP instruction to avoid an immediate exit from stop2. PTA5/IRQ/TPM1CLK/ $\overline{\text{RESET}}$ can be disabled as a wakeup if it is configured as a high driven output. For lowest power consumption in stop 2, this pin should not be left open if configured as input (enable the internal pullup or tie an external pullup device or set pin as output).

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

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

- All module control and status registers are reset, except for SPMSC1-SPMSC3, RTCSC, RTCCNT, and RTCMOD.
- The LVD reset function is enabled and the MCU remains in the reset state if V_{DD} is below the LVD trip point
- The CPU takes the reset vector

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

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

To maintain I/O states for pins configured as general-purpose I/O before entering stop2, you must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the pins switches to their reset states when PPDACK is written.

For pins configured as peripheral I/O, you 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 is controlled by their associated port control registers when the I/O latches are opened.

8.6.1.1 Stop2 Mode Recovery Time

The stop2 recovery time is defined as the interval from the exit trigger to the first opcode fetch. There are three main components to this wake up time: the voltage regulator recovery time, the clock source start up time, and the reset processing time.

The voltage regulator recovery time (t_{VRR}) is provided in the data sheet. This time is not influenced by the clock source frequency or V_{DD} and is therefore relatively consistent.

Because exiting from stop2 causes the MCU to wake up as if a POR occurred, the standard reset processing will always occur which takes about 150 ICSOUT cycles after the clock source has started. Therefore, the equation for stop2 recovery time is

$$\text{Stop2 recovery time} = t_{VRR} + \text{clock start up time} + 150 \text{ ICSOUT cycles.} \quad \text{Eqn. 8-1}$$

Because ICSOUT defaults to FLL output running at 8.4 MHz during a reset, and the FLL takes about 1 microsecond to start outputting a clock signal (although it won't be stable initially) [Equation 8-5](#) simplifies to

$$\text{Stop2 recovery time} = t_{VRR} + 1 \mu\text{s} + 17.9 \mu\text{s.} \quad \text{Eqn. 8-2}$$

8.6.2 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions in [Table 8-1](#). The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained.

Stop3 can be exited by asserting $\overline{\text{RESET}}$, or by an interrupt from one of the following sources: the RTC, LVD, LVW, ADC, ACMPx, IRQ, SCIx or the KBIx.

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

8.6.2.1 Stop3 Mode Recovery Time

The stop3 recovery time is defined as the interval from the exit trigger to the first opcode fetch. There are three main components to this wake up time: the voltage regulator recovery time, the clock source start up time, and the reset or interrupt processing time.

When an interrupt is used as the exit trigger, the clock must restart and ICSOUT must oscillate six times before the interrupt processing begins. The interrupt processing requires 11 bus cycles (22 ICSOUT cycles) for the stacking and vector fetch. Therefore, the first opcode of the interrupt service routine (ISR) will begin after

$$\text{Stop3 recovery time} = t_{VRR} + \text{clock start up time} + 28 \text{ ICSOUT cycles.} \quad \text{Eqn. 8-3}$$

The clock source start up time is dependent on the clock mode selected when the MCU enters stop mode. When the FLL output is selected as the clock source, the FLL starts up within a microsecond at roughly the same frequency as before stop mode is entered. Typical start up time for the internal reference is given in the data sheet. Typical start up times for the crystal oscillator are also given in the data sheet.

Assuming the FLL is the selected clock source upon entering stop3 and the FLL is configured for a 20 MHz ICSOUT frequency, then [Equation 8-6](#) simplifies to

$$\text{Stop3 recovery time} = t_{VRR} + 1 \mu\text{s} + 1.4 \mu\text{s} \quad \text{Eqn. 8-4}$$

When reset is used as the exit trigger, more time is required for the reset processing, so [Equation 8-3](#) becomes

$$\text{Stop3 recovery time} = t_{VRR} + \text{clock start up time} + 162 \text{ ICSOUT cycles.} \quad \text{Eqn. 8-5}$$

Because ICSOUT defaults to FLL output running at 8.4 MHz during a reset, [Equation 8-5](#) simplifies to

$$\text{Stop3 recovery time} = t_{VRR} + 1 \mu\text{s} + 19.3 \mu\text{s.} \quad \text{Eqn. 8-6}$$

8.6.3 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. Also, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If you attempt to enter stop2 with ENBDM set, the MCU instead enters stop3.

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

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

8.6.4 LVD Enabled in Stop Mode

The LVD system can generate an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set), the voltage regulator remains active during stop mode. If you attempt to enter stop2 with the LVD enabled for stop, the MCU instead enters stop3.

8.6.5 Stop modes in Low Power Run Mode

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

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

8.7 Mode selection

Several control signals are used to determine the current operating mode of the device. [Table 8-2](#) shows the conditions for each of the device's operating modes.

Table 8-2. Power Mode Selections

Mode of Operation	BDCSCR BDM	SPMSC1 PMC		SPMSC2 PMC		CPU & Periph CLKs	Affects on Sub-System	
	ENBDM ¹	LVDE	LVDSE	LPR	PPDC		BDM Clock	Voltage Regulator
Run mode	0	x	x	0	x	on. ICS in any mode.	off	on
		1	1	1			on	
	1	x	x	x				
LP run mode	0	0	x	1	0	low freq required. ICS in FBELP mode only.	off	standby
		1	0					
Wait mode - (Assumes WAIT instruction executed.)	0	x	x	0	0	CPU clock is off; peripheral clocks on. ICS state same as run mode.	off	on
		1	1	1			on	
	1	x	x	x				
LP wait mode - (Assumes WAIT instruction executed.)	0	0	x	1	x	CPU clock is off; peripheral clocks at low speed. ICS in FBELP mode.	off	standby
		1	0					
Stop3 - (Assumes STOPE bit is set and STOP instruction executed.) Note that stop3 is used in place of stop2 if the BDM or LVD is enabled.	0	0	x	x	0	ICS in STOP. IC SERCLK and ICS IRCLK optionally on ²	off	standby
	0	1	0	x	0		off	
	0	1	1	x	x		off	
	1	x	x	x	x	on		
Stop2 - (Assumes STOPE bit is set and STOP instruction executed.) If BDM or LVD is enabled, stop3 is invoked rather than stop2.	0 ³	0	x	0	1	OSCOUT optionally on ^{2,4}	off	partial powerdown
		1	0					

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

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

³ Design Interlock: ENBDM=1 disallows STOP2. If STOP was issued from RUN mode, then STOP4 will be entered. If STOP was issued from LPRUN, then STOP3 will be entered.

⁴ In stop2, CPU, flash, ICS and all peripheral modules are powered down except for the RTC.

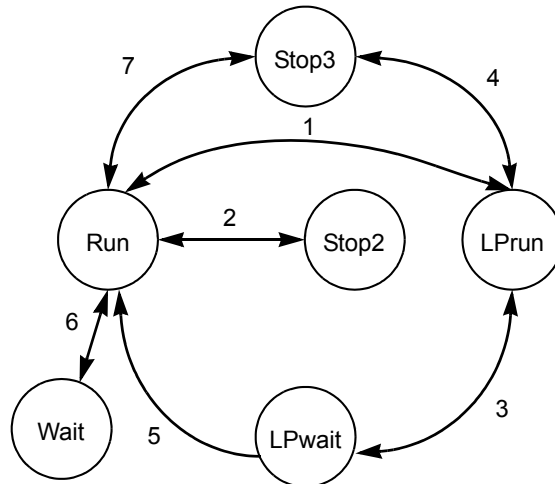


Figure 8-1. Allowable Power Mode Transitions for the 9S08QE32

Table 8-3. Regulator States

Mode	Regulator State
Run	Full on
Wait	Full on
LPrun	Standby
LPwait	Standby
Stop3	Standby
Stop2	Partial powerdown

Figure 8-1 illustrates mode state transitions allowed between the legal states (Table 8-1). PTA5/IRQ/TPM1CLK/RESET must be asserted low in order to exit stop2. Interrupts suffice for the other stop and wait modes.

Table 8-4 defines triggers for the various state transitions (Figure 8-1).

Table 8-4. Triggers for Transitions in Figure 8-1

Transition #	From	To	Trigger
1	Run	LPrun	Configure settings in Table 8-1, switch LPR=1 last
	LPrun	Run	Clear LPR Interrupt when LPWUI=1
2	Run	Stop2	Pre-configure settings in Table 8-1, issue STOP instruction
	Stop2	Run	assert zero on PTA5/IRQ/TPM1CLK/RESET¹ , reload environment from RAM

Table 8-4. Triggers for Transitions in Figure 8-1 (continued)

Transition #	From	To	Trigger
3	LPrun	LPwait	WAIT instruction
	LPwait	LPrun	Interrupt when LPWUI=0
4	LPrun	Stop3	STOP instruction
	Stop3	LPrun	Interrupt when LPWUI=0
5	LPwait	Run	Interrupt when LPWUI=1
	Run	LPwait	NOT SUPPORTED
6	Run	Wait	WAIT instruction
	Wait	Run	Interrupt or reset
7	Stop3	Run	Interrupt (if LPR = 0, or LPR = 1 and LPWUI =1) or reset
	Run	Stop3	STOP instruction

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

8.7.1 On-Chip Peripheral Modules in Stop and Low Power Modes

When the 9S08QE32 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 8.6.1, “Stop2 Mode”](#), and [Section 8.6.2, “Stop3 Mode”](#), for specific information on system behavior in stop modes.

When the MCU enters LPWait or LPRun modes, system clocks to the internal peripheral modules continue based on the settings of the clock gating control registers (SCGC1 and SCGC2).

Table 8-5. Stop and Low Power Mode Behavior

Peripheral	Mode			
	Stop2	Stop3	LPWait	LPRun
CPU	Off	Standby	Standby	On
RAM	Standby	Standby	Standby	On
FLASH	Off	Standby	Standby	On
Port I/O Registers	Off	Standby	Standby	On
ADC12	Off	Optionally On ¹	Optionally On ¹	Optionally On ¹
ACMPx	Off	Optionally On ²	Optionally On	Optionally On
BDM	Off ³	Optionally On	Off ⁴	Off ⁴
COP	Off	Off	Optionally On	Optionally On
ICS	Off	Optionally On ⁵	On ⁶	On ⁶
IIC	Off	Standby	Optionally On	Optionally On
IRQ	Wake Up	Optionally On	Optionally On	Optionally On
KBIX	Off	Optionally On	Optionally On	Optionally On
LVD/LVW	Off ⁷	Optionally On	Off ⁸	Off ⁸
RTC	Optionally On	Optionally On	Optionally On	Optionally On
SCIx	Off	Standby	Optionally On	Optionally On
SPI	Off	Standby	Optionally On	Optionally On
TPMx	Off	Standby	Optionally On	Optionally On
Voltage Regulator	Partial Powerdown	Optionally On ⁹	Standby	Standby
XOSCVP	Optionally On ¹⁰	Optionally On ¹⁰	Optionally On	Optionally On
I/O Pins	States Held	Peripheral Control	Peripheral Control	On

¹ Requires the asynchronous ADC clock. For stop3, LVD must be enabled to run in stop if converting the bandgap channel.

² LVD must be enabled to run in stop if using the bandgap as a reference.

³ If ENBDM is set when entering stop2, the MCU actually enters stop3.

⁴ If ENBDM is set when entering LPRun or LPWait, the MCU actually stays in run mode or enter wait mode, respectively.

⁵ IRCLKEN and IREFSTEN set in ICSC1, else in standby.

⁶ ICS must be configured for FBELP, bus frequency limited to 125kHz in LPRun or LPWait.

⁷ If LVDSE is set when entering stop2, the MCU actually enters stop3.

⁸ If LVDSE is set when entering LPRun or LPWait, the MCU actually enters run or wait mode, respectively.

⁹ Requires the LVD to be enabled, else in standby. See [Section 8.6.4, “LVD Enabled in Stop Mode”](#).

¹⁰ ERCLKEN and EREFSTEN set in ICSC2, else in standby.

Chapter 9 Memory

9.1 9S08QE32 Memory Map

On-chip memory in the 9S08QE32 series of MCUs consists of RAM, flash program memory for nonvolatile data storage, and I/O and control/status registers (Figure 9-1). The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x007F)
- High-page registers (0x1800 through 0x187F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

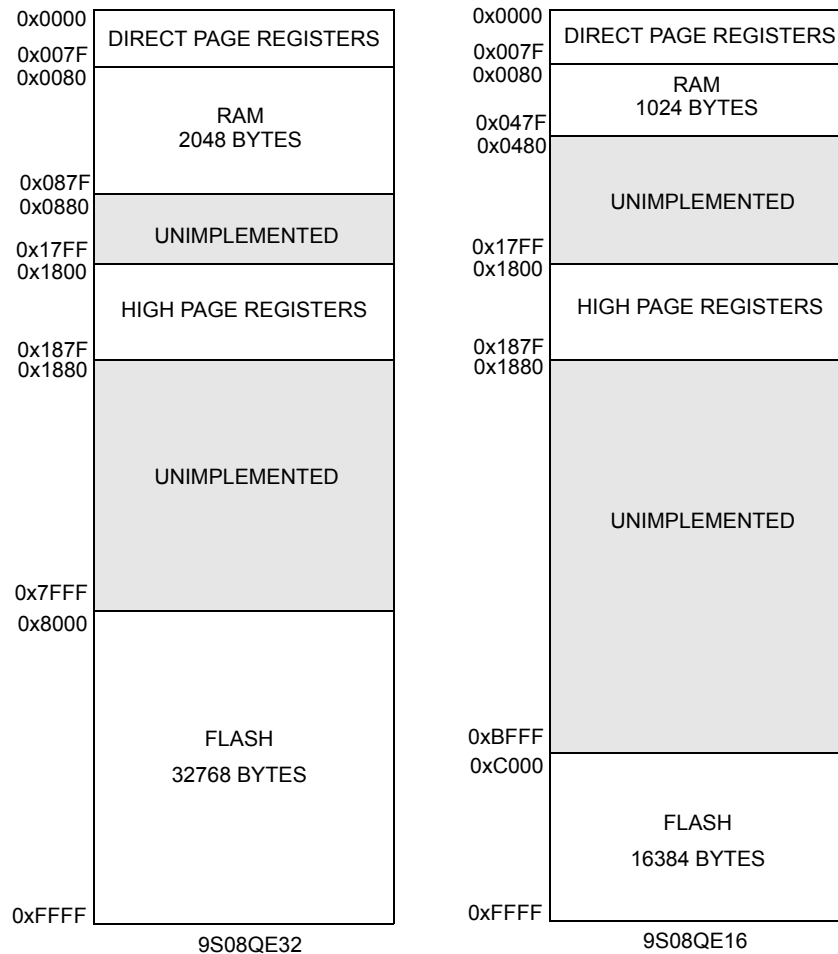


Figure 9-1. 9S08QE32 Memory Maps

9.2 Reset and Interrupt Vector Assignments

Table 9-1 shows address assignments for reset and interrupt vectors. The vector names in this table are the labels used in the Freescale Semiconductor provided equate file for the 9S08QE32 series.

Table 9-1. Reset and Interrupt Vectors

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1	TPM3 Overflow	Vtpm3ovf
0xFFC2:0xFFC3	TPM3 Channel 5	Vtpm3ch5
0xFFC4:0xFFC5	TPM3 Channel 4	Vtpm3ch4
0xFFC6:0xFFC7	TPM3 Channel 3	Vtpm3ch3
0xFFC8:0xFFC9	TPM3 Channel 2	Vtpm3ch2
0xFFCA:0xFFCB	TPM3 Channel 1	Vtpm3ch1
0xFFCC:0xFFCD	TPM3 Channel 0	Vtpm3ch0
0xFFCE:0xFFCF	RTC	Vrtc
0xFFD0:0xFFD1	SCI2 Transmit	Vsci2tx
0xFFD2:0xFFD3	SCI2 Receive	Vsci2rx
0xFFD4:0xFFD5	SCI2 Error	Vsci2err
0xFFD6:0xFFD7	ACMPx ¹	Vacmpx
0xFFD8:0xFFD9	ADC Conversion	Vadc
0xFFDA:0xFFDB	KBlx Interrupt ²	Vkeyboard
0xFFDC:0xFFDD	IIC	Viic
0xFFDE:0xFFDF	SCI1 Transmit	Vsci1tx
0xFFE0:0xFFE1	SCI1 Receive	Vsci1rx
0xFFE2:0xFFE3	SCI1 Error	Vsci1err
0xFFE4:0xFFE5	SPI	Vspi
0xFFE6:0xFFE7	Reserved	—
0xFFE8:0xFFE9	TPM2 Overflow	Vtpm2ovf
0xFFEA:0xFFEB	TPM2 Channel 2	Vtpm2ch2
0xFFEC:0xFFED	TPM2 Channel 1	Vtpm2ch1
0xFFEE:0xFFEF	TPM2 Channel 0	Vtpm2ch0
0xFFFF0:0xFFFF1	TPM1 Overflow	Vtpm1ovf
0xFFFF2:0xFFFF3	TPM1 Channel 2	Vtpm1ch2
0xFFFF4:0xFFFF5	TPM1 Channel 1	Vtpm1ch1
0xFFFF6:0xFFFF7	TPM1 Channel 0	Vtpm1ch0
0xFFFF8:0xFFFF9	Low Voltage Detect or Low Voltage Warning	Vlvd

Table 9-1. Reset and Interrupt Vectors (continued)

Address (High/Low)	Vector	Vector Name
0xFFFA:0xFFFB	IRQ	Virq
0xFFFC:0xFFFD	SWI	Vswi
0xFFFE:0xFFFF	Reset	Vreset

¹ ACMP1 and ACMP2 share this vector, if both modules are enabled, poll each flag to determine pending interrupt.

² KBI1 and KBI2 share this vector, if both modules are enabled, poll each flag to determine pending interrupt.

9.3 Register Addresses and Bit Assignments

The registers in the 9S08QE32 series are divided into these groups:

- Direct-page registers are the ones located in the first 256 bytes locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and RAM.
- The nonvolatile register area consists of a block of 16 locations in flash memory at 0xFFB0–0xFFBF. Nonvolatile register locations include:
 - NVPROT and NVOPT are loaded into working registers at reset
 - An 8-byte backdoor comparison key that optionally allows you to gain controlled access to secure memory

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

Direct-page registers can be accessed with efficient direct addressing mode instructions. Bit manipulation instructions can be used to access any bit in any direct-page register. [Table 9-2](#) is a summary of all user-accessible direct-page registers and control bits.

The direct page registers in [Table 9-2](#) can use the more efficient direct addressing mode, which requires only the lower byte of the address. Because of this, the lower byte of the address in column one is in bold text. In [Table 9-3](#) and [Table 9-4](#), the whole address in column one is in bold. In [Table 9-2](#), [Table 9-3](#), and [Table 9-4](#), the register names in column two are in bold to set them apart from the bit names to the right. Cells not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s. When writing to these bits, write a 0 unless otherwise specified.

Table 9-2. Direct-Page Register Summary (Sheet 1 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0	
0x0000	PTAD	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0	
0x0001	PTADD	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0	
0x0002	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0	
0x0003	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0	
0x0004	PTCD	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0	
0x0005	PTCDD	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0	
0x0006	PTDD	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0	
0x0007	PTDDD	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0	
0x0008	PTED	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0	
0x0009	PTEDD	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0	
0x000A		—	—	—	—	—	—	—	—	
—	Reserved	—	—	—	—	—	—	—	—	
0x000B		—	—	—	—	—	—	—	—	
0x000C	KBI1SC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD	
0x000D	KBI1PE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0	
0x000E	KBI1ES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0	
0x000F	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD	
0x0010	ADSC1	COCO	AIEN	ADCO	ADCH					
0x0011	ADSC2	ADACT	ADTRG	ACFE	ACFGT	—	—	—	—	
0x0012	ADRH	0	0	0	0	ADR11	ADR10	ADR9	ADR8	
0x0013	ADRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	
0x0014	ADCVH	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8	
0x0015	ADCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0	
0x0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK		
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0	
0x0018	APCTL2	—	—	—	—	—	—	ADPC9	ADPC8	
0x0019	Reserved	—	—	—	—	—	—	—	—	
0x001A	ACMP1SC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0	
0x001B	ACMP2SC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0	
0x001C		—	—	—	—	—	—	—	—	
—	Reserved	—	—	—	—	—	—	—	—	
0x001F		—	—	—	—	—	—	—	—	
0x0020	SCI1BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8	
0x0021	SCI1BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0	
0x0022	SCI1C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT	
0x0023	SCI1C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	
0x0024	SCI1S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	
0x0025	SCI1S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF	
0x0026	SCI1C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE	
0x0027	SCI1D	Bit 7	6	5	4	3	2	1	Bit 0	

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	SPIC1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0029	SPIC2	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x002A	SPIBR	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x002B	SPIS	SPRF	0	SPTEF	MODF	0	0	0	0
0x002C	Reserved	0	0	0	0	0	0	0	0
0x002D	SPID	Bit 7	6	5	4	3	2	1	Bit 0
0x002E	Reserved	—	—	—	—	—	—	—	—
0x002F		—	—	—	—	—	—	—	—
0x0030	IICA	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x0031	IICF	MULT			<u>TAP2 TAP1</u> ICR				
0x0032	IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x0033	IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x0034	IICD	DATA							
0x0035	IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x0036	Reserved	—	—	—	—	—	—	—	—
0x0037		—	—	—	—	—	—	—	—
0x0038	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0039	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTE N
0x003A	ICSTRM	TRIM							
0x003B	ICSSC	DRS/DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
0x003C	KBI2SC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x003D	KBI2PE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x003E	KBI2ES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x003F	Reserved	—	—	—	—	—	—	—	—
0x0040	TPM1SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0041	TPM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0042	TPM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0043	TPM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0044	TPM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0045	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0046	TPM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0047	TPM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0048	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0049	TPM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x004A	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x004B	TPM1C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0

Table 9-2. Direct-Page Register Summary (Sheet 3 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x004C	TPM1C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x004D	TPM1C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x004E	Reserved	—	—	—	—	—	—	—	—
0x004F		—	—	—	—	—	—	—	—
0x0050	TPM2SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0051	TPM2CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0052	TPM2CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0053	TPM2MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0054	TPM2MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0055	TPM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0056	TPM2C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0057	TPM2C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0058	TPM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0059	TPM2C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x005A	TPM2C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x005B	TPM2C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x005C	TPM2C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x005D	TPM2C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x005E	Reserved	—	—	—	—	—	—	—	—
0x005F		—	—	—	—	—	—	—	—
0x0060	TPM3SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0061	TPM3CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0062	TPM3CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0063	TPM3MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0064	TPM3MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0065	TPM3C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0066	TPM3C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0067	TPM3C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0068	TPM3C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0069	TPM3C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x006A	TPM3C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x006B	TPM3C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x006C	TPM3C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x006D	TPM3C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x006E	TPM3C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x006F	TPM3C3VH	Bit 15	14	13	12	11	10	9	Bit 8

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0070	TPM3C3VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0071	TPM3C4SC	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	0	0
0x0072	TPM3C4VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0073	TPM3C4VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0074	TPM3C5SC	CH5F	CH5IE	MS5B	MS5A	ELS5B	ELS5A	0	0
0x0075	TPM3C5VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0076	TPM3C5VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0077		—	—	—	—	—	—	—	—
—	Reserved	—	—	—	—	—	—	—	—
0x007F		—	—	—	—	—	—	—	—

High-page registers (Table 9-3) 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 9-3. High-Page Register Summary (Sheet 1 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBD FR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPE	COPT	STOPE	—	0	0	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	0	0	SPIFE	SPIPS	ACIC2	IICPS	ACIC1
0x1804– 0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x1809	SPMSC2	LPR	LPRS	LPWUI	0	PPDF	PPDACK	PPDE	PPDC
0x180A	Reserved	—	—	—	—	—	—	—	—
0x180B	SPMSC3	LVWF	LVWACK	LVDV	LVWV	LVWIE	—	—	—
0x180C	Reserved	—	—	—	—	—	—	—	—
0x180D	SPCF	—	—	—	—	—	—	FLSCF	RAMCF
0x180D	Reserved	—	—	—	—	—	—	—	—
0x180E	SCGC1	TPM3	TPM2	TPM1	ADC	1	IIC	SCI2	SCI1
0x180F	SCGC2	DBG	FLS	IRQ	KBI	ACMP	RTC	1	SPI
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGCCH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGCCL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8

Table 9-3. High-Page Register Summary (Sheet 2 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1817	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1818	DBGCAx	RWAEN	RWA	0	0	0	0	0	0
0x1819	DBGCBx	RWBEN	RWB	0	0	0	0	0	0
0x181A	DBGCCx	RWCEN	RWC	0	0	0	0	0	0
0x181B	Reserved	—	—	—	—	—	—	—	—
0x181C	DBGc	DBGEN	ARM	TAG	BRKEN	0	0	0	LOOP1
0x181D	DBGt	TRGSEL	BEGIN	0	0	TRG			
0x181E	DBGs	AF	BF	CF	0	0	0	0	ARMF
0x181F	DBGcNT	0	0	0	0	CNT			
0x1820	FCDIV	DIVLD	PRDIV8	DIV					
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS							FPDIS
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD							
0x1827– 0x182F	Reserved	—	—	—	—	—	—	—	—
0x1830	RTCSC	RTIF	RTCLKS		RTIE	RTCPS			
0x1831	RTCCNT	RTCCNT							
0x1832	RTCMOD	RTCMOD							
0x1833– 0x183F	Reserved	—	—	—	—	—	—	—	—
0x1840	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1845	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1846	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1849	PTCSE	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x184A	PTCDS	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	PTDPE	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x184D	PTDSE	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x184E	PTDDS	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x184F	Reserved	—	—	—	—	—	—	—	—

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1850	PTEPE	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x1851	PTESE	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x1852	PTEDS	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x1853	Reserved	—	—	—	—	—	—	—	—
0x1854– 0x186F	Reserved	—	—	—	—	—	—	—	—
0x1870	SCI2BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x1871	SCI2BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x1872	SCI2C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x1873	SCI2C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x1874	SCI2S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x1875	SCI2S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x1876	SCI2C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x1877	SCI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x1878– 0x187F	Reserved	—	—	—	—	—	—	—	—

Several reserved flash memory locations (Table 9-4) are used for storing values used by several registers. These registers include an 8-byte backdoor key, NVBACKKEY, which can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the reserved flash memory are transferred into corresponding FPROT and FOPT registers in the high-page registers area to control security and block protection options.

The factory ICS trim value is stored in the flash information row (IFR¹) and will be loaded into the ICSTRM and ICSSC registers after any reset. The internal reference trim values stored in flash, TRIM and FTRIM, can be programmed by third party programmers and must be copied into the corresponding ICS registers by user code to override the factory trim.

NOTE

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

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

Table 9-4. Reserved Flash Memory Addresses

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAE	Reserved for storage of FT-RIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Reserved for store of ICSTRM	TRIM							
0xFFB0 – 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8 – 0xFFBC	Reserved	—	—	—	—	—	—	—	—
0xFFBD	NVPROT	FPS							FPDIS
0xFFBE	Reserved	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC	

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

9.4 RAM

The 9S08QE32 series include static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided the supply voltage does not drop below the minimum value for RAM retention (V_{RAM}).

For compatibility with M68HC05 MCUs, the QE32 resets the stack pointer to 0x00FF. In the 9S08QE32 series, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor-provided equate file).

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

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See [Section 9.6, “Security”](#), for a detailed description of the security feature.

9.5 Flash

The flash memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to the *QE32 Family Reference Manual*.

9.5.1 Features

Features of the flash memory include:

- Flash size
 - 9S08QE32: 32,768 bytes (64 pages of 512 bytes each)
 - MC9S08GB32: 16,384 bytes (32 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for flash and RAM
- Auto power-down for low-frequency read accesses

9.5.2 Program and Erase Times

Before any program or erase command can be accepted, the flash clock divider register (FCDIV) must be written to set the internal clock for the flash module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz (see [Section 9.7.1, “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. You must ensure FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{\text{FCLK}}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses are used by the command processor to complete a program or erase command.

[Table 9-5](#) 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 $t_{\text{FCLK}} = 1/f_{\text{FCLK}}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{\text{FCLK}} = 5 \mu\text{s}$. Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

Table 9-5. Program and Erase Times

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 μ s
Byte program (burst)	4	20 μ s ¹
Page erase	4000	20 ms
Mass erase	20,000	100 ms

¹ Excluding start/end overhead

9.5.3 Program and Erase Command Execution

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

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

NOTE

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

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

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

A strictly monitored procedure must be obeyed or the command is not accepted. This minimizes the possibility of any unintended changes to the flash memory contents. The command complete flag (FCCF) indicates when a command is complete. The command sequence must be completed by clearing FCBEF to launch the command. [Figure 9-2](#) is a flowchart for executing all of the commands except for burst programming. The FCDIV register must be initialized before using any flash commands. This must be done only once following a reset.

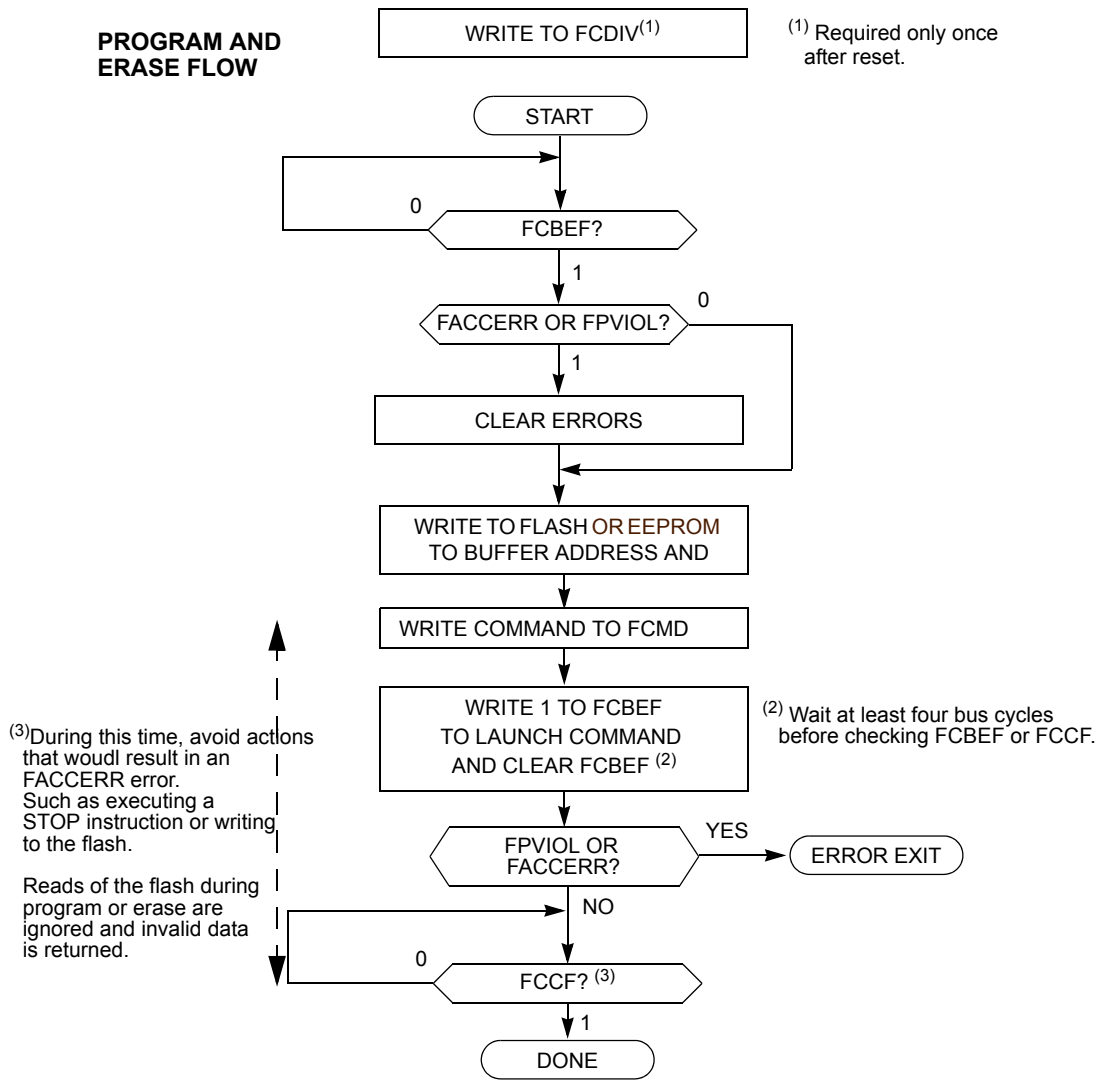


Figure 9-2. Flash Program and Erase Flowchart

9.5.4 Burst Program Execution

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

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

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

The first byte of a series of sequential bytes being programmed in burst mode takes the same amount of time to program as a byte programmed in standard mode. Subsequent bytes 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 is 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 is disabled and high voltage removed from the array.

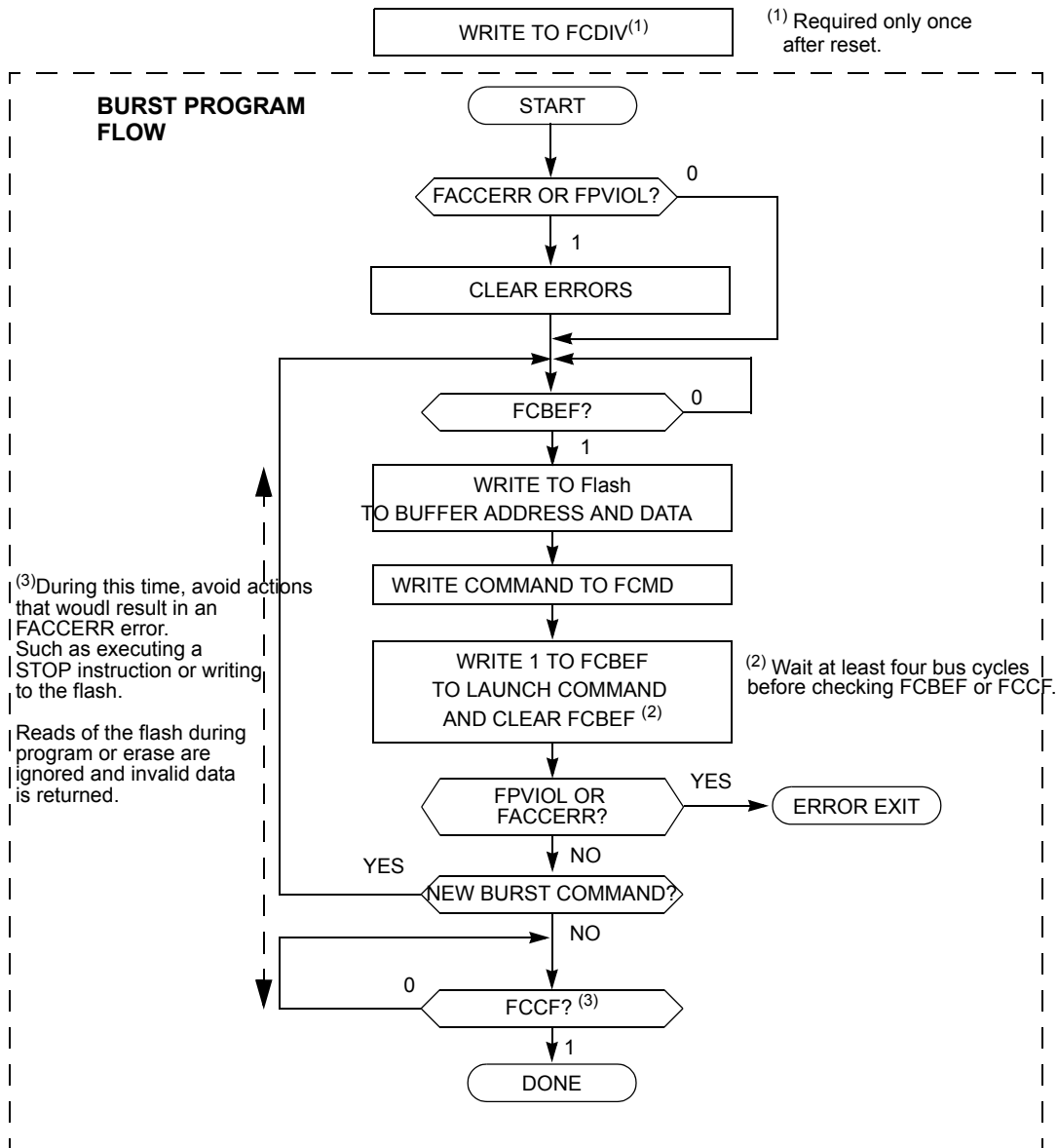


Figure 9-3. Flash Burst Program Flowchart

9.5.5 Access Errors

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

Any of the following specific actions causes 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

9.5.6 Flash Block Protection

The block protection feature prevents the protected region of flash from program or erase changes. Block protection is controlled through the flash protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of flash, 0xFFFF. (See [Section 9.7.4, “Flash Protection Register \(FPROT and NVPROT\)”](#).)

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

The block protection mechanism is illustrated in [Figure 9-4](#). The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits, as shown. For example, to protect the last 1536 bytes of memory (addresses 0xFA00 through 0xFFFF), the FPS bits must be set to 1111 100, which results in the value 0xF9FF as the last address of unprotected memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT)

must be programmed to logic 0 to enable block protection. Therefore the value 0xF8 must be programmed into NVPROT to protect addresses 0xFA00 through 0xFFFF.

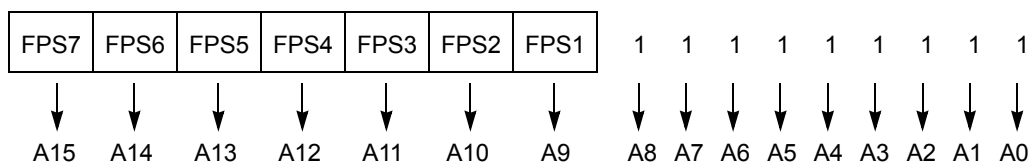


Figure 9-4. Block Protection Mechanism

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.

9.5.7 Vector Redirection

When any block protection is enabled, the reset and interrupt vectors are protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the flash memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFFD) are redirected, though the reset vector (0xFFFFE:FFFF) is not.

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 you 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.

9.6 Security

The 9S08QE32 series include circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, flash and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from flash into the working FOPT register in high-page register space. You engage security by programming the NVOPT location which can be done at the same time the flash memory is programmed. The 1:0 state disengages security and the other three combinations engage security. Notice the erased state (1:1) makes the MCU secure. During development, whenever the flash is erased, 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 of unsecured resources.

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

1. Writing 1 to KEYACC in the FCNFG register. This makes the flash module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a flash program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX must not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally retrieves 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 are disengaged until the next reset.

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

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

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

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

9.7 Flash Registers and Control Bits

The flash module has nine 8-bit registers in the high-page register space, two locations (NVOPT, NVPROT) in the nonvolatile register space in flash memory are copied into corresponding high-page control registers (FOPT, FPROT) at reset. There is also an 8-byte comparison key in flash memory. Refer to [Table 9-3](#) and [Table 9-4](#) for the absolute address assignments for all flash registers. This section refers

to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

9.7.1 Flash Clock Divider Register (FCDIV)

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

Offset

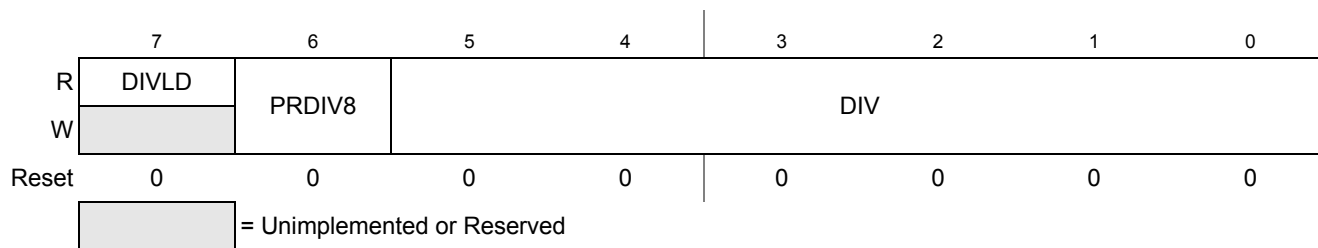


Figure 9-5. Flash Clock Divider Register (FCDIV)

Table 9-6. FCDIV Register Field Descriptions

Field	Description
7 DIVLD	<p>Divisor Loaded Status Flag — 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>Prescale (Divide) flash Clock by 8</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>
5:0 DIV	<p>Divisor for flash Clock Divider — The flash clock divider divides the bus rate clock (or the bus rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV field plus one. The resulting frequency of the internal flash clock must fall within the range of 200 kHz to 150 kHz for proper flash operations. Program/Erase timing pulses are one cycle of this internal flash clock which corresponds to a range of 5 μs to 6.7 μs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See Equation 9-1 and Equation 9-2.</p>

$$\text{if PRDIV8} = 0 \text{ — } f_{\text{CLK}} = f_{\text{Bus}} \div (\text{DIV} + 1) \tag{Eqn. 9-1}$$

$$\text{if PRDIV8} = 1 \text{ — } f_{\text{CLK}} = f_{\text{Bus}} \div (8 \times (\text{DIV} + 1)) \tag{Eqn. 9-2}$$

Table 9-7 shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

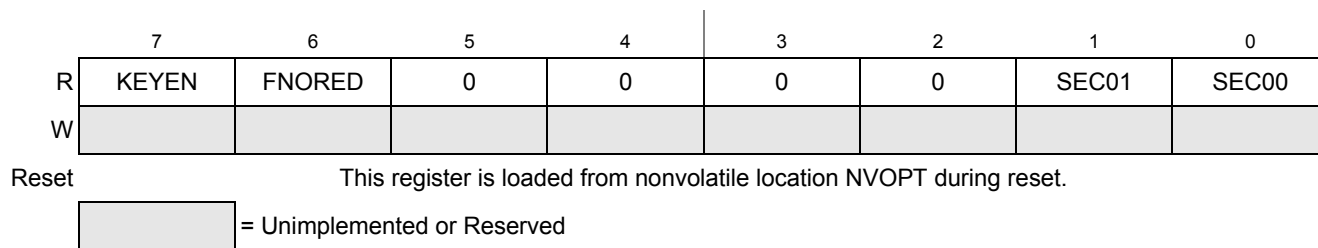
Table 9-7. Flash Clock Divider Settings

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

9.7.2 Flash Options Register (FOPT and NVOPT)

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

Offset


Figure 9-6. Flash Options Register (FOPT)
Table 9-8. FOPT Register Field Descriptions

Field	Description
7 KEYEN	<p>Backdoor Key Mechanism Enable — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from your (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 Section 9.6, “Security”.</p> <p>0 No backdoor key access allowed.</p> <p>1 If your 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>

Table 9-8. FOPT Register Field Descriptions (continued)

Field	Description
6 FNORED	Vector Redirection Disable — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	Security State Code — This 2-bit field determines the security state of the MCU (Table 9-9). When the MCU is secure, the contents of RAM and flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash. For more detailed information about security, refer to Section 9.6, “Security”.

Table 9-9. Security States¹

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

¹ SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash.

9.7.3 Flash Configuration Register (FCNFG)

Offset

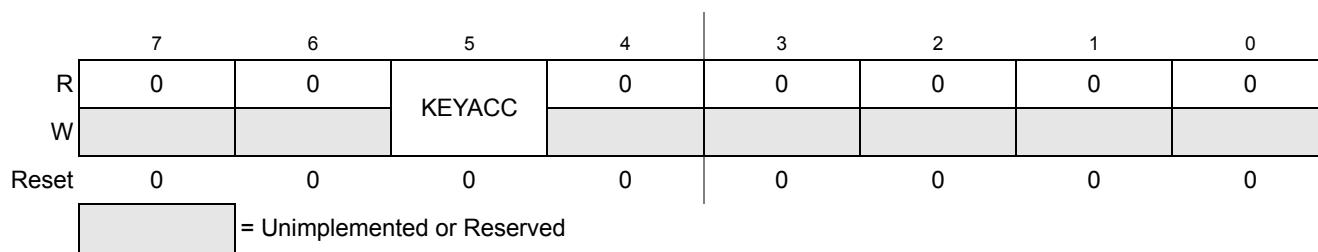


Figure 9-7. Flash Configuration Register (FCNFG)

Table 9-10. FCNFG Register Field Descriptions

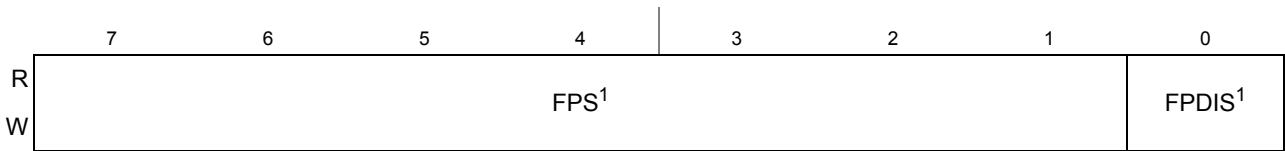
Field	Description
5 KEYACC	Enable Writing of Access Key — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to Section 9.6, “Security”. 0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a flash programming or erase command. 1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.

9.7.4 Flash Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT is copied from flash into FPROT. FPROT can be read at any time. With FPDIS set, all bits are writable, but with FPDIS clear the FPS bits are writable

as long as the size of the protected region is being increased. Any FPROT write that attempts to decrease the size of the protected region is ignored.

Offset



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

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

Figure 9-8. Flash Protection Register (FPROT)

Table 9-11. FPROT Register Field Descriptions

Field	Description
7:1 FPS	Flash Protect Select Bits — When FPDIS = 0, this 7-bit field determines the ending address of unprotected flash locations at the high address end of the flash. Protected flash locations cannot be erased or programmed.
0 FPDIS	Flash Protection Disable 0 Flash block specified by FPS7:FPS1 is block protected (program and erase not allowed). 1 No flash block is protected.

9.7.5 Flash Status Register (FSTAT)

Offset

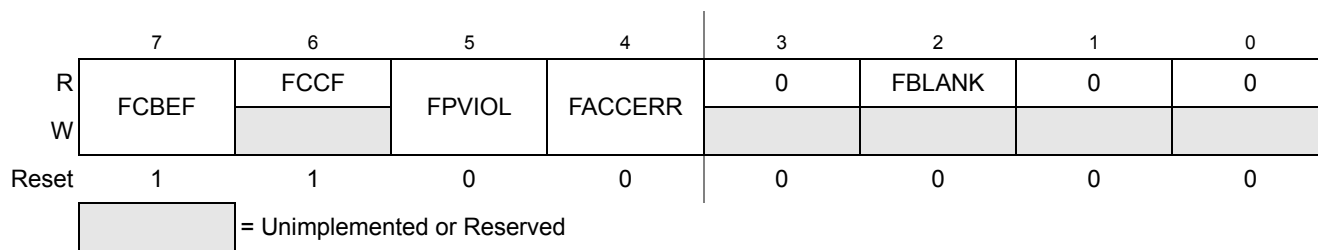


Figure 9-9. Flash Status Register (FSTAT)

Table 9-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	<p>Flash Command Buffer Empty Flag — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so 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.</p> <p>0 Command buffer is full (not ready for additional commands). 1 A new burst program command can be written to the command buffer.</p>
6 FCCF	<p>Flash Command Complete Flag — 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.</p> <p>0 Command in progress 1 All commands complete</p>
5 FPVIOL	<p>Protection Violation Flag — FPVIOL is set automatically when a command is written that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL.</p> <p>0 No protection violation. 1 An attempt was made to erase or program a protected location.</p>
4 FACCERR	<p>Access Error Flag — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the actions considered access errors, see Section 9.5.5, “Access Errors”. FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect.</p> <p>0 No access error. 1 An access error has occurred.</p>
2 FBLANK	<p>Flash Verified as All Blank (erased) Flag — 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.</p> <p>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).</p>

9.7.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes (Table 9-13). Refer to Section 9.5.3, “Program and Erase Command Execution”, for a detailed discussion of flash programming and erase operations.

Offset

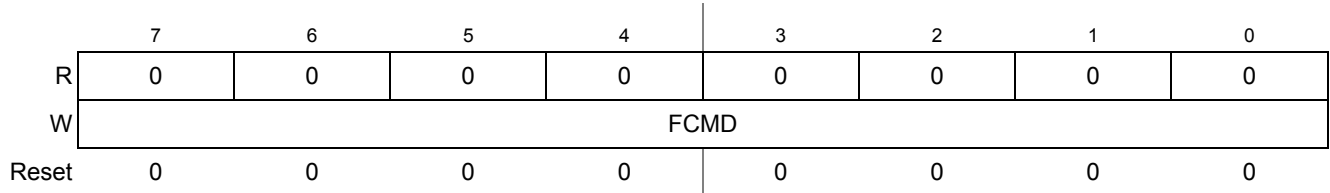


Figure 9-10. Flash Command Register (FCMD)

Table 9-13. Flash Commands

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

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

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

Chapter 10

Resets, Interrupts, and General System Control

10.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the 9S08QE32 series. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this document. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog are not part of on-chip peripheral systems with their own chapters.

10.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vector for most modules (reduces polling overhead) (see [Table 10-2](#))

10.3 MCU Reset

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

The 9S08QE32 series have the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Low-voltage detect (LVD)
- Background debug forced reset

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

10.4 Computer Operating Properly (COP) Watchdog

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

After any reset, the COPE becomes set in SOPT1 enabling the COP watchdog (see [Section 10.8.4, “System Options Register 1 \(SOPT1\)”](#),” for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The COPCLKS bit in SOPT2 (see [Section 10.8.5, “System Options Register 2 \(SOPT2\)”](#),” for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1-kHz clock source. With each clock source, there is an associated short and long time-out controlled by COPT in SOPT1. [Table 10-1](#) summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1-kHz clock source and the associated long time-out (2^8 cycles) `{sim_908aw60_isc_cop.asm}`.

Table 10-1. COP Configuration Options

Control Bits		Clock Source	COP Overflow Count
COPCLKS	COPT		
0	0	~1 kHz	2^5 cycles (32 ms) ¹
0	1	~1 kHz	2^8 cycles (256 ms) ¹
1	0	Bus	2^{13} cycles
1	1	Bus	2^{18} cycles

¹ Values in this column based on $t_{LPO} = 1$ ms. See t_{LPO} in the data sheet for the tolerance of this value.

Even if the application uses the reset default settings of COPE, COPCLKS, and COPT, you must write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost. The initial writes to SOPT1 and SOPT2 resets the COP counter.

The write to SRS that services (clears) the COP counter must not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

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

When the bus clock source is selected, the COP counter does not increment while the system is in stop mode. The COP counter resumes as soon as the MCU exits stop mode.

When the 1 kHz clock source is selected, the COP counter is re-initialized to zero upon entry to stop mode. `{sim_908aw60_isc_cop_bdm.asm}` The COP counter begins from zero after the MCU exits stop mode.

10.5 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 becomes set. The CPU will not respond unless the local interrupt enable is a 1 (enabled) and the I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which prevents all maskable interrupt sources. Your program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

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

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

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

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

NOTE

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

If more than one interrupt is pending when the I bit is cleared, the highest priority source is serviced first (see [Table 10-2](#)).

10.5.1 Interrupt Stack Frame

Figure 10-1 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.

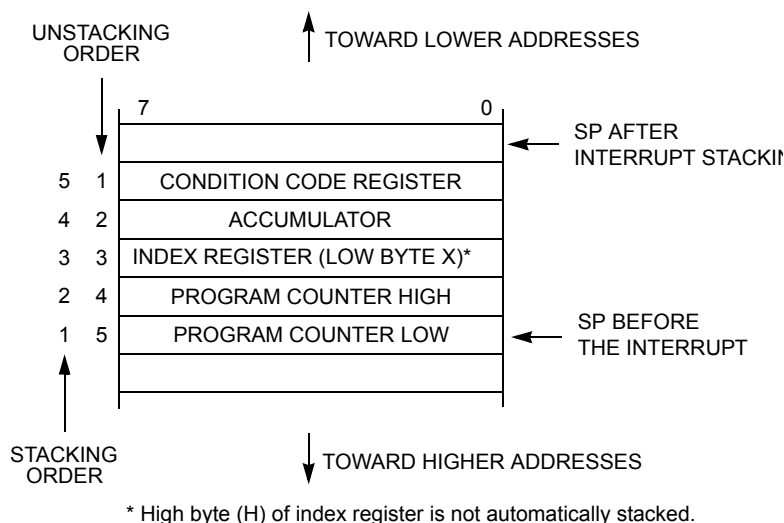


Figure 10-1. Interrupt Stack Frame

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

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

10.5.2 External Interrupt Request (IRQ) Pin

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

10.5.2.1 Pin Configuration Options

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

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

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

NOTE

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

The voltage measured on the internally pulled up PTA5/IRQ/TPM1CLK/RESET pin is not pulled to V_{DD} . The internal gates connected to this pin are pulled to V_{DD} . Do not use the PTA5/IRQ/TPM1CLK/RESET pullup to pullup components external to the MCU.

NOTE

When enabling the IRQ pin for use, the $IRQF$ will be set, and should be cleared prior to enabling the interrupt. When configuring the pin for falling edge and level sensitivity in a 3V system, it is necessary to wait at least 6 cycles between clearing the flag and enabling the interrupt.

10.5.2.2 Edge and Level Sensitivity

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

10.5.3 Interrupt Vectors, Sources, and Local Masks

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

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

Table 10-2. Vector Summary

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lowest Highest	31	0xFFC0/0xFFC1	Vtpm3ovf	TPM3	TOF	TOIE	TPM3 overflow
	30	0xFFC2/0xFFC3	Vtpm3ch5	TPM3	CH5F	CH5IE	TPM3 channel 5
	29	0xFFC4/0xFFC5	Vtpm3ch4	TPM3	CH4F	CH4IE	TPM3 channel 4
	28	0xFFC6/0xFFC7	Vtpm3ch3	TPM3	CH3F	CH3IE	TPM3 channel 3
	27	0xFFC8/0xFFC9	Vtpm3ch2	TPM3	CH2F	CH2IE	TPM3 channel 2
	26	0xFFCA/0xFFCB	Vtpm3ch1	TPM3	CH1F	CH1IE	TPM3 channel 1
	25	0xFFCC/0xFFCD	Vtpm3ch0	TPM3	CH0F	CH0IE	TPM3 channel 0
	24	0xFFCE/0xFFCF	Vrtc	RTC	RTIF	RTIE	Real-time interrupt
	23	0xFFD0/0xFFD1	Vsci2tx	SCI2	TDRE, TC	TIE, TCIE	SCI2 transmit
	22	0xFFD2/0xFFD3	Vsci2rx	SCI2	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI2 receive
	21	0xFFD4/0xFFD5	Vsci2err	SCI2	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI2 error
	20	0xFFD6/0xFFD7	Vacmpx	ACMPx ¹	ACF	ACIE	Analog comparator x
	19	0xFFD8/0xFFD9	Vadc	ADC	COCO	AIEN	ADC
	18	0xFFDA/0xFFDB	Vkeyboard	KBIX ²	KBF	KBIE	Keyboard x pins
	17	0xFFDC/0xFFDD	Viic	IIC	IICIS	IICIE	IIC control
	16	0xFFDE/0xFFDF	Vsci1tx	SCI1	TDRE, TC	TIE, TCIE	SCI1 transmit
	15	0xFFE0/0xFFE1	Vsci1rx	SCI1	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI1 receive
	14	0xFFE2/0xFFE3	Vsci1err	SCI1	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI1 error
	13	0xFFE4/0xFFE5	Vspi	SPI	SPIF, MODF, SPTEF	SPIE, SPIE, SPTIE	SPI
	12	0xFFE6/0xFFE7	—	—	—	—	—
	11	0xFFE8/0xFFE9	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
	10	0xFFEA/0xFFEB	Vtpm2ch2	TPM2	CH2F	CH2IE	TPM2 channel 2
	9	0xFFEC/0xFFED	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
	8	0xFFEE/0xFFEF	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
	7	0xFFFF0/0xFFFF1	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
	6	0xFFFF2/0xFFFF3	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2
	5	0xFFFF4/0xFFFF5	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
	4	0xFFFF6/0xFFFF7	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
	3	0xFFFF8/0xFFFF9	Vlvd	System control	LVDF, LVWF	LVDIE, LVWIE	Low-voltage detect, Low-voltage warning
	2	0xFFFFA/0xFFFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	0xFFFFC/0xFFFFD	Vswi	Core	SWI Instruction	—	Software interrupt
	0	0xFFFFE/0xFFFFF	Vreset	System control	COP LVD RESET pin Illegal opcode Illegal address	COPE LVDRE — — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address

¹ ACMP1 and ACMP2 share this vector, if both modules are enabled, poll each flag to determine pending interrupt.

² KBI1 and KBI2 share this vector, if both modules are enabled, poll each flag to determine pending interrupt.

10.6 Low-Voltage Detect (LVD) System

The 9S08QE32 series include a system to protect against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit. The LVD circuit is enabled when LVDE in SPMSC1. The LVD is disabled upon entering either of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU enters stop3 instead of stop2, and the current consumption in stop3 with the LVD enabled is greater.

10.6.1 Power-On Reset Operation

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

10.6.2 Low-Voltage Detection (LVD) Reset Operation

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

10.6.3 Low-Voltage Detection (LVD) Interrupt Operation

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

10.6.4 Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low voltage warning flag (LVWF) to indicate that the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an interrupt associated with it, enabled by setting the LVWIE bit in the SPMSC3 register. If enabled, an LVW interrupt request occurs when the LVWF is set. LVWF is cleared by writing a 1 to the LVWACK bit in SPMSC3.

10.7 Peripheral Clock Gating

The 9S08QE32 series include a clock gating system to manage the bus clock sources to the individual peripherals. Using this system, you can enable or disable the bus clock to each of the peripherals at the clock source, eliminating unnecessary clocks to peripherals which are not in use and thereby reducing the overall run and wait mode currents.

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

NOTE

Your software must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by your software.

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

10.8 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.

This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation.

10.8.1 Interrupt Pin Request Status and Control Register (IRQSC)

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

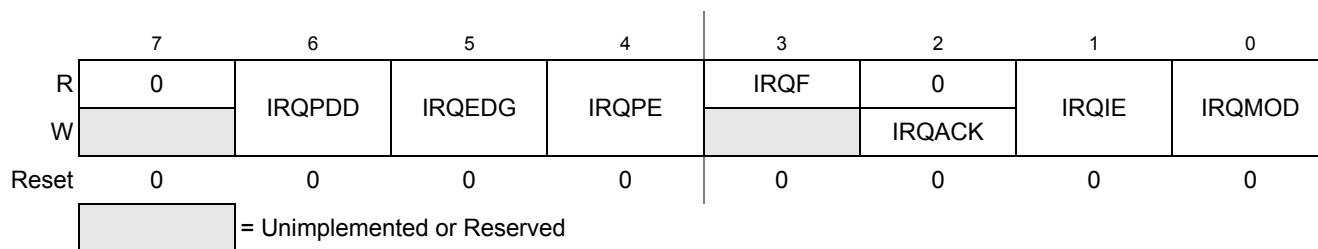


Figure 10-2. Interrupt Request Status and Control Register (IRQSC)

Table 10-3. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	Interrupt Request (IRQ) Pull Device Disable — This read/write control bit is used to disable the internal pullup/pulldown device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	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 causes IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When IRQEDG = 1 and the internal pull device is enabled, the pullup device is reconfigured as an optional pulldown device. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	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. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	IRQ Flag — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 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.
1 IRQIE	IRQ Interrupt Enable — This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever IRQF = 1.
0 IRQMOD	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 detected as interrupt request events. See Section 10.5.2.2, “Edge and Level Sensitivity” for more details. 0 IRQ event on falling edges or rising edges only. 1 IRQ event on falling edges and low levels or on rising edges and high levels.

10.8.2 System Reset Status Register (SRS)

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

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u ¹	0	0	0	0	0	1	0
Any other reset:	0	Note ²	Note ²	Note ²	Note ²	0	0	0

¹ u = unaffected

² Any of these reset sources that are active at the time of reset entry causes the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry are cleared.

Figure 10-3. System Reset Status (SRS)

Table 10-4. SRS Register Field Descriptions

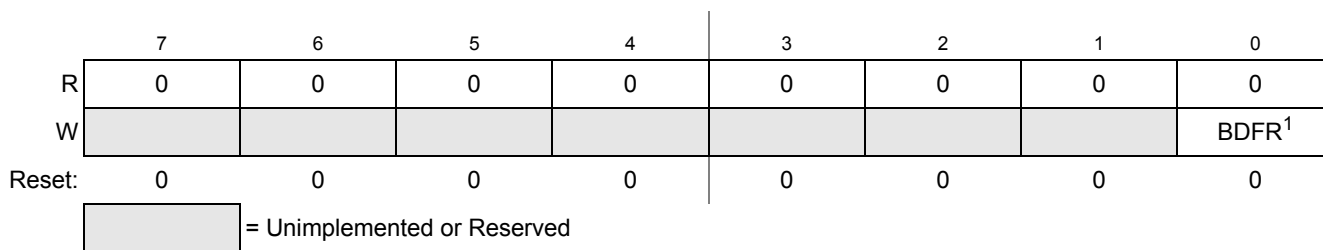
Field	Description
7 POR	Power-On Reset — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	Computer Operating Properly (COP) Watchdog — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by COPE = 0. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	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. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.

Table 10-4. SRS Register Field Descriptions (continued)

Field	Description
3 ILAD	Illegal Address — Reset was caused by an attempt to access either data or an instruction at an unimplemented memory address. 0 Reset not caused by an illegal address 1 Reset caused by an illegal address
1 LVD	Low Voltage Detect — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

10.8.3 System Background Debug Force Reset Register (SBDFR)

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



¹ BDFR is writable only through serial background debug commands, not from your programs.

Figure 10-4. System Background Debug Force Reset Register (SBDFR)
Table 10-5. SBDFR Register Field Descriptions

Field	Description
0 BDFR	Background Debug Force Reset — A serial background command such as WRITE_BYTE can be used to allow an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from your program. To enter user mode, PTA4/ACMPO/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTA4/ACMPO/BKGD/MS must be low immediately after issuing WRITE_BYTE command. See the data sheet for more information.

10.8.4 System Options Register 1 (SOPT1)

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

Offset

	7	6	5	4	3	2	1	0
R	COPE	COPT	STOPE	0	0	0	BKGDPE	RSTPE
W								
Reset:	1	1	0	0	0	0	1	u ¹
POR:	1	1	0	0	0	0	1	0
LVR:	1	1	0	0	0	0	1	0

= Unimplemented or Reserved
 u = unaffected

Figure 10-5. System Integration Module Options Register 1 (SOPT1)

Table 10-6. SOPT1 Register Field Descriptions

Field	Description
7 COPE	COP Watchdog Enable — This write-once bit selects whether the COP watchdog is enabled. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	COP Watchdog Timeout — This write-once bit selects the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. 0 Short timeout period selected. 1 Long timeout period selected.
5 STOPE	Stop Mode Enable — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. STOPT — Stop Recovery Time Select This write-once bit is not used and ignored in the 9S08QE32 because this MCU always defaults to the self-clocked VCO clock source on recovery from stop mode. Because there is little startup time for the self-clocked clock source, stop recovery is always fast in the 9S08QE32. 1 = Long stop recovery to allow crystal oscillator startup (does not apply in 9S08QE32). Short stop recovery (assumes oscillator is running as the MCU recovers from stop). 1 Stop mode enabled.

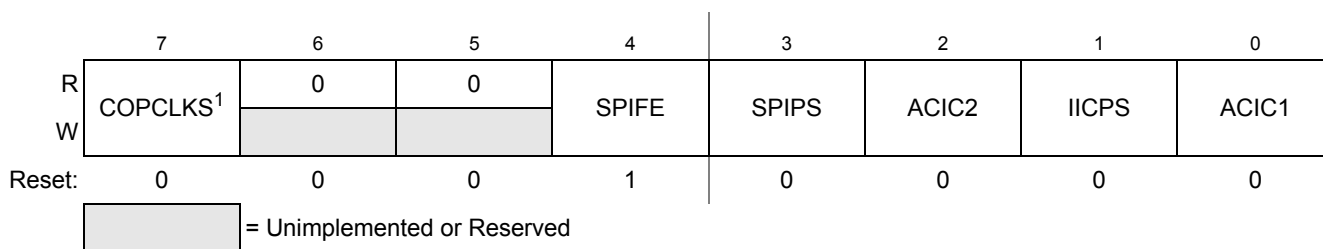
Table 10-6. SOPT1 Register Field Descriptions (continued)

Field	Description
1 BKGDPPE	<p>Background Debug Mode Pin Enable — This write-once bit when set enables the PTA4/ACMP10/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset.</p> <p>0 PTA4/ACMP10/BKGD/MS pin functions as PTA4 or ACMP10STOP — Stop Recovery Time Select This write-once bit is not used and ignored in the 9S08QE32 because this MCU always defaults to the self-clocked VCO clock source on recovery from stop mode. Because there is little startup time for the self-clocked clock source, stop recovery is always fast in the 9S08QE32. 1 = Long stop recovery to allow crystal oscillator startup (does not apply in 9S08QE32). Short stop recovery (assumes oscillator is running as the MCU recovers from stop). 1 PTA4/ACMP10/BKGD/MS pin functions as BKGD/MS.</p>
0 RSTPE	<p>RESET Pin Enable — This write-once bit when set enables the PTA5/IRQ/TPM1CLK/RESET pin to function as RESET. When clear, the pin functions as one of its input only alternative functions. This pin defaults to its PTA5 function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on RESET.</p> <p>0 PTA5/IRQ/TPM1CLK/RESET pin functions as PTA5, IRQ or TPM1CLK.STOPT — Stop Recovery Time Select This write-once bit is not used and ignored in the 9S08QE32 because this MCU always defaults to the self-clocked VCO clock source on recovery from stop mode. Because there is little startup time for the self-clocked clock source, stop recovery is always fast in the 9S08QE32. 1 = Long stop recovery to allow crystal oscillator startup (does not apply in 9S08QE32). Short stop recovery (assumes oscillator is running as the MCU recovers from stop). 1 PTA5/IRQ/TPM1CLK/RESET pin functions as RESET.</p>

10.8.5 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the 9S08QE32 series devices.

Offset



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

Figure 10-6. System Integration Module Options Register 2 (SOPT2)

Table 10-7. SOPT2 Register Field Descriptions

Field	Description
7 COPCLKS	COP Watchdog Clock Select — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz clock is source to COP. 1 Bus clock is source to COP.
4 SPIFE	SPI Ports Input Filter Enable 0 Disable input filter on SPI port pins to allow for higher maximum SPI baud rate. 1 Enable input filter on SPI port pins to eliminate noise and restrict maximum SPI baud rate.
3 SPIPS	SPI Pin Select — This bit selects the location of the MOSI, MISO, SPSCLK, and \overline{SS} pins of the SPI module. 0 SPSCLK on PTB2, MOSI on PTB3, MISO on PTB4, and \overline{SS} on PTB5. 1 SPSCLK on PTE0, MOSI on PTE1, MISO on PTE2, and \overline{SS} on PTE3.
2 ACIC2	Analog Comparator 2 to Input Capture Enable — This bit connects the output of ACMP2 to TPM2 input channel 0. 0 ACMP2 output not connected to TPM2 input channel 0. 1 ACMP2 output connected to TPM2 input channel 0.
1 IICPS	IIC Pin Select — This bit selects the location of the SDA and SCL pins of the IIC module. 0 SDA on PTA2, SCL on PTA3. 1 SDA on PTB6, SCL on PTB7.
0 ACIC1	Analog Comparator 1 to Input Capture Enable — Connects the output of ACMP1 to TPM1 input channel 0. 0 ACMP output not connected to TPM1 input channel 0. 1 ACMP output connected to TPM1 input channel 0.

10.8.6 System Device Identification Register (SDIDH, SDIDL)

These high page read-only registers are included so host development systems can identify the QE32 derivative. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

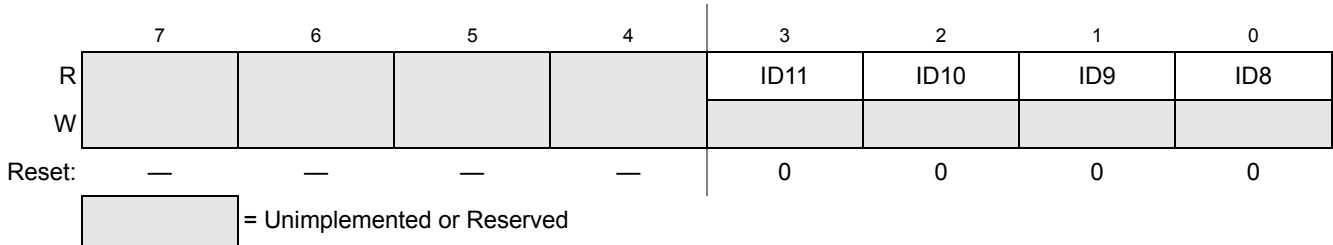


Figure 10-7. System Device Identification Register - High (SDIDH)

Table 10-8. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	Bits 7:4 are reserved. Reading these bits result in an indeterminate value; writes have no effect.
3:0 ID[11:8]	Part Identification Number — Each derivative in the QE32 family has a unique identification number. The 9S08QE32 is hard coded to the value 0x01F. See also ID bits in Table 10-9 .

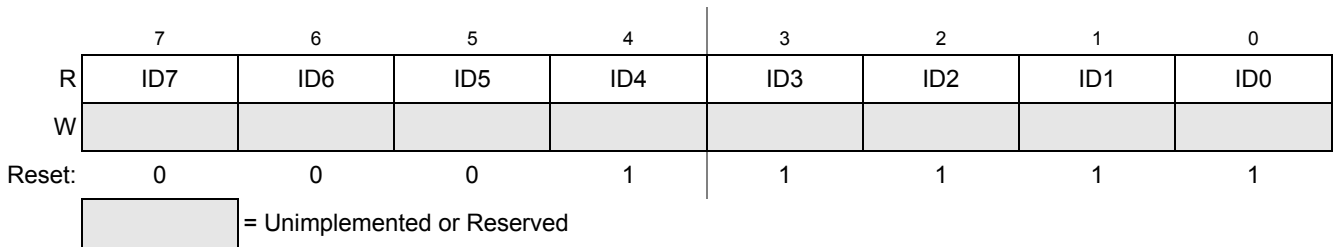


Figure 10-8. System Device Identification Register — Low (SDIDL)

Table 10-9. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	Part Identification Number — Each derivative in the QE32 family has a unique identification number. The 9S08QE32 is hard coded to the value 0x01F. See also ID bits in Table 10-8 .

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

This high page register contains status and control bits to support the low voltage detect function, and to enable the bandgap voltage reference for use by the ADC or ACMP modules. To configure the low voltage detect trip voltage, see [Table 10-12](#) for the LVDV bit description in SPMSC3.

Offset

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

= Unimplemented or Reserved
 u = Unaffected by reset

¹ Bit 1 is a reserved bit that must always be written to 0.

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

Figure 10-9. System Power Management Status and Control 1 Register (SPMSC1)

Table 10-10. SPMSC1 Register Field Descriptions

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

Table 10-10. SPMSC1 Register Field Descriptions (continued)

Field	Description
2 LVDE	Low-Voltage Detect Enable — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	Bandgap Buffer Enable — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels or as a voltage reference for ACMP module. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

10.8.8 System Power Management Status and Control 2 Register (SPMSC2)

This high page register contains status and control bits to configure the low power run and wait modes as well as configure the stop mode behavior of the MCU.

Offset

	7	6	5	4	3	2	1	0
R	LPR	LPRS	LPWUI	0	PPDF	0	PPDE ¹	PPDC
W						PPDACK		
Reset:	0	0	0	0	0	0	1	0
Stop2 wakeup:	0	0	u	0	1	0	1	1

= Unimplemented or Reserved
 u= Unaffected by reset

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

Figure 10-10. System Power Management Status and Control 2 Register (SPMSC2)
Table 10-11. SPMSC2 Register Field Descriptions

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

Table 10-11. SPMSC2 Register Field Descriptions (continued)

Field	Description
3 PPDF	Partial Power Down Flag — This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	Partial Power Down Acknowledge — Writing a 1 to PPDACK clears the PPDF bit.
1 PPDE	Partial Power Down Enable — The write-once PPDE bit can be used to “lockout” the partial power down mode. 0 Partial power down is not enabled. 1 Partial power down is enabled and controlled via the PPDC bit.
0 PPDC	Partial Power Down Control — The PPDC bit controls which power down mode is selected. This bit cannot be set if LPR=1. If PPDC and LPR are set in a single write instruction, only PPDC actually is set. 0 Stop3 low power mode enabled. 1 Stop2 partial power down mode enabled.

10.8.9 System Power Management Status and Control 3 Register (SPMSC3)

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

Offset

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	LVWIE	0	0	0
W		LVWACK						
Reset:	0 ¹	0	0	0	0	0	0	0
Stop2 wakeup:	u	0	u	u	u	0	0	0
Reset:	0 ¹	0	u	u	0	0	0	0
Any other reset:	0 ¹	0	u	u	0	0	0	0

= Unimplemented or Reserved

u= Unaffected by reset

¹ LVWF is set in the case when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW} .

Figure 10-11. System Power Management Status and Control 3 Register (SPMSC3)

Table 10-12. SPMSC3 Register Field Descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag — The LVWF bit indicates the low voltage warning status. 0 Low voltage warning not present. 1 Low voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge — The LVWF bit indicates the low voltage warning status. Writing a 1 to LVWACK clears LVWF to a 0 if a low voltage warning is not present.
5 LVDV	Low-Voltage Detect Voltage Select —The LVDV bit selects the LVD trip point voltage (V_{LVD}). 0 Low trip point selected ($V_{LVD} = V_{LVDL}$). 1 High trip point selected ($V_{LVD} = V_{LV DH}$).
4 LVWV	Low-Voltage Warning Voltage Select —The LVWV bit selects the LVW trip point voltage (V_{LVW}). 0 Low trip point selected ($V_{LVW} = V_{LVWL}$). 1 High trip point selected ($V_{LVW} = V_{LVWH}$).
3 LVWIE	Low-Voltage Warning Interrupt Enable — This bit enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVWF = 1.

Table 10-13. LVD and LVW trip point typical values¹

LVDV:LVWV	LVW Trip Point	LVD Trip Point
0:0	$V_{LVWL} = 2.15 \text{ V}$	$V_{LVDL} = 1.84 \text{ V}$
0:1	$V_{LVWH} = 2.48 \text{ V}$	
1:0 ²	$V_{LVWL} = 2.15 \text{ V}$	$V_{LVDH} = 2.15 \text{ V}$
1:1	$V_{LVWH} = 2.48 \text{ V}$	

¹ See the data sheet for minimum and maximum values.

² This setting is not recommended.

10.8.10 System Clock Gating Control 1 Register (SCGC1)

This high page register contains control bits to enable or disable the bus clock to the TPMx, ADC, IIC, and SCIx modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents. See [Section 10.7, "Peripheral Clock Gating"](#), for more information.

NOTE

Your software must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by your software.

Offset

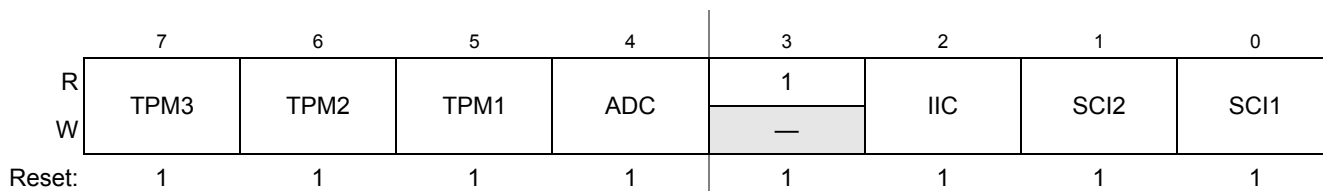


Figure 10-12. System Clock Gating Control 1 Register (SCGC1)

Table 10-14. SCGC1 Register Field Descriptions

Field	Description
7 TPM3	TPM3 Clock Gate Control — This bit controls the clock gate to the TPM3 module. 0 Bus clock to the TPM3 module is disabled. 1 Bus clock to the TPM3 module is enabled.
6 TPM2	TPM2 Clock Gate Control — This bit controls the clock gate to the TPM2 module. 0 Bus clock to the TPM2 module is disabled. 1 Bus clock to the TPM2 module is enabled.
5 TPM1	TPM1 Clock Gate Control — This bit controls the clock gate to the TPM1 module. 0 Bus clock to the TPM1 module is disabled. 1 Bus clock to the TPM1 module is enabled.

Table 10-14. SCGC1 Register Field Descriptions (continued)

Field	Description
4 ADC	ADC Clock Gate Control — This bit controls the clock gate to the ADC module. 0 Bus clock to the ADC module is disabled. 1 Bus clock to the ADC module is enabled.
2 IIC	IIC Clock Gate Control — This bit controls the clock gate to the IIC module. 0 Bus clock to the IIC module is disabled. 1 Bus clock to the IIC module is enabled.
1 SCI2	SCI Clock Gate Control — This bit controls the clock gate to the SCI2 module. 0 Bus clock to the SCI2 module is disabled. 1 Bus clock to the SCI2 module is enabled.
0 SCI1	SCI Clock Gate Control — This bit controls the clock gate to the SCI1 module. 0 Bus clock to the SCI1 module is disabled. 1 Bus clock to the SCI1 module is enabled.

10.8.11 System Clock Gating Control 2 Register (SCGC2)

This high page register contains control bits to enable or disable the bus clock to the RTC and SPI modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents. See [Section 10.7, "Peripheral Clock Gating"](#), for more information.

NOTE

Your software must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by your software.

Offset

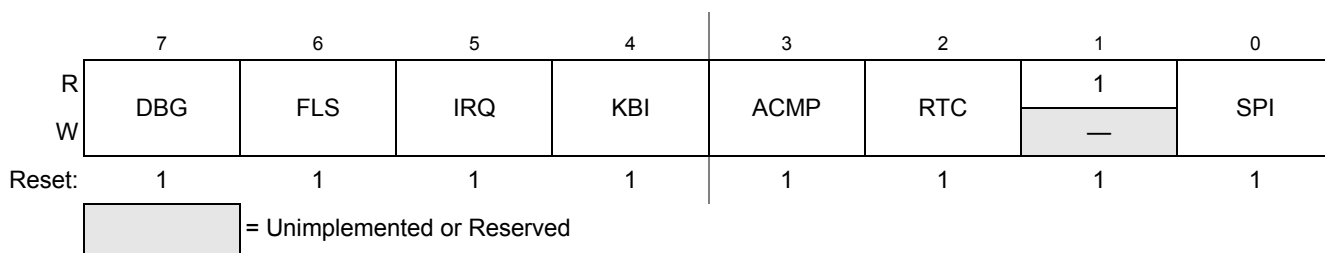

Figure 10-13. System Clock Gating Control 2 Register (SCGC2)

Table 10-15. SCGC2 Register Field Descriptions

Field	Description
7 DBG	DBG Register Clock Gate Control — This bit controls the bus clock gate to the DBG module. 0 Bus clock to the DBG module is disabled. 1 Bus clock to the DBG module is enabled.
6 FLS	Flash Register Clock Gate Control — This bit controls the bus clock gate to the flash registers. This bit does not affect normal program execution from with the flash array. Only the clock to the flash control registers is affected. 0 Bus clock to the flash registers is disabled. 1 Bus clock to the flash registers is enabled.
5 IRQ	IRQ Clock Gate Control — This bit controls the bus clock gate to the IRQ module. 0 Bus clock to the IRQ module is disabled. 1 Bus clock to the IRQ module is enabled.
4 KBI	KBI Clock Gate Control — This bit controls the clock gate to the KBI module. 0 Bus clock to the KBI module is disabled. 1 Bus clock to the KBI module is enabled.
3 ACMP	ACMP Clock Gate Control — This bit controls the clock gate to both of the ACMP modules. 0 Bus clock to the ACMP modules is disabled. 1 Bus clock to the ACMP modules is enabled.
2 RTC	RTC Clock Gate Control — This bit controls the bus clock gate to the RTC module. Only the bus clock is gated, the IC SERCLK and LPOCLK are still available to the RTC. 0 Bus clock to the RTC module is disabled. 1 Bus clock to the RTC module is enabled.
0 SPI	SPI Clock Gate Control — This bit controls the clock gate to the SPI module. 0 Bus clock to the SPI module is disabled. 1 Bus clock to the SPI module is enabled.

Chapter 11

MCU Parallel Input/Output

This section explains software controls related to parallel input/output (I/O) and pin control. The 9S08QE32 has five parallel I/O ports which include a total of 40 I/O pins, including one output-only pin and one input-only pin.

Many of these pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts. The peripheral modules have priority over the general-purpose I/O functions so that when a peripheral is enabled, the I/O functions associated with the shared pins may be disabled.

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

NOTE

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

11.1 Port Data and Data Direction

Reading and writing of parallel I/Os are performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram (Figure 11-1).

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

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

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

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 is not driven momentarily with an old data value that happened to be in the port data register.

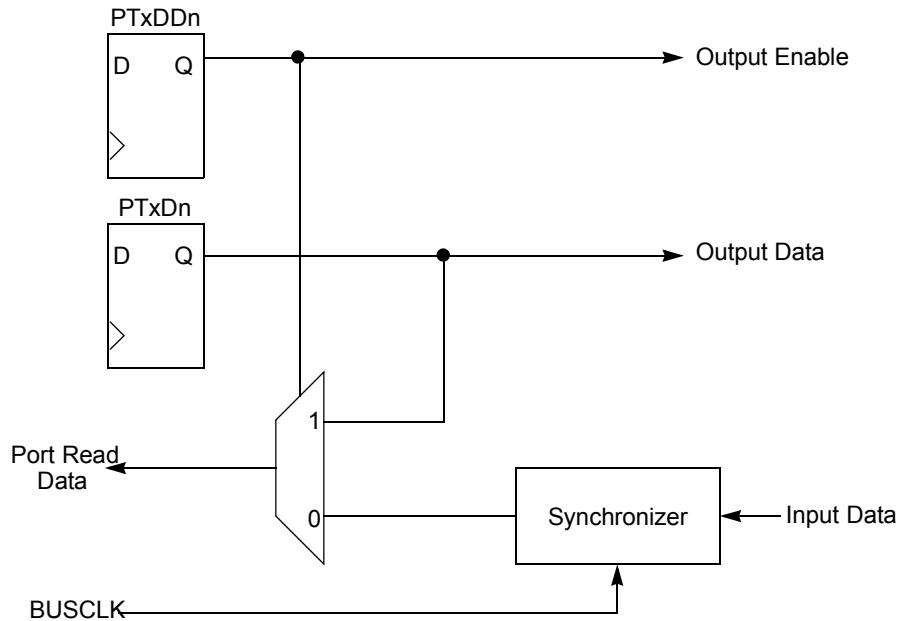


Figure 11-1. Parallel I/O Block Diagram

11.2 Pullup, Slew Rate, and Drive Strength

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

11.2.1 Port Internal Pullup Enable

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

11.2.2 Port Slew Rate Enable

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

11.2.3 Port Drive Strength Select

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

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

11.3 Pin Behavior in Stop Modes

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

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

11.4 Parallel I/O and Pin Control Registers

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

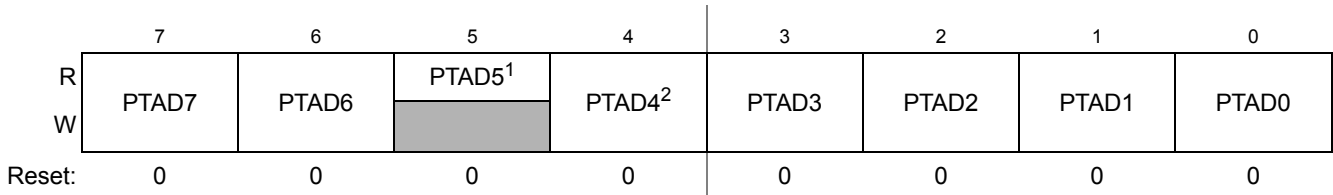
This section refers to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

11.4.1 Port A Registers

Port A is controlled by the registers listed below.

The pins PTA4 and PTA5 are unique. PTA4 is an output only, so the control bits for the input functions has no effect on this pin. PTA5 is an input only, so the control bits for the output functions has no effect on this pin.

11.4.1.1 Port A Data Register (PTAD)



¹ Reads of bit PTAD5 always return the pin value of PTAD5, regardless of the value stored in bit PTADD5.

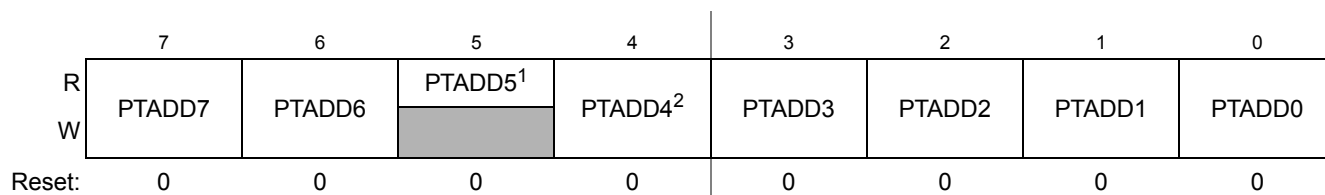
² Reads of bit PTAD4 always return the contents of PTAD4, regardless of the value stored in bit PTADD4.

Figure 11-2. Port A Data Register (PTAD)

Table 11-1. PTAD Register Field Descriptions

Field	Description
7:0 PTAD[7:0]	<p>Port A Data Register Bits — For port A pins that are inputs, reads return the logic level on the pin. For port A pins 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 configured as outputs, the logic level is driven out the corresponding MCU pin.</p> <p>Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups/pulldowns disabled.</p>

11.4.1.2 Port A Data Direction Register (PTADD)



¹ PTADD5 has no effect on the input-only PTA5 pin.

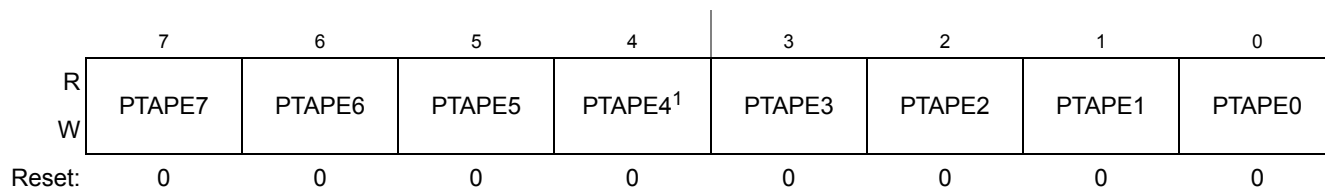
² PTADD4 has no effect on the output-only PTA4 pin.

Figure 11-3. Port A Data Direction Register (PTADD)

Table 11-2. PTADD Register Field Descriptions

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

11.4.1.3 Port A Pull Enable Register (PTAPE)



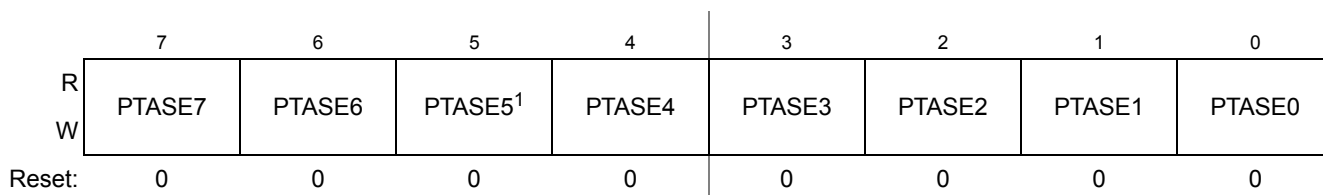
¹ PTAPE4 does not affect the output only PTA4 pin.

Figure 11-4. Internal Pull Enable for Port A Register (PTAPE)

Table 11-3. PTAPE Register Field Descriptions

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

11.4.1.4 Port A Slew Rate Enable Register (PTASE)



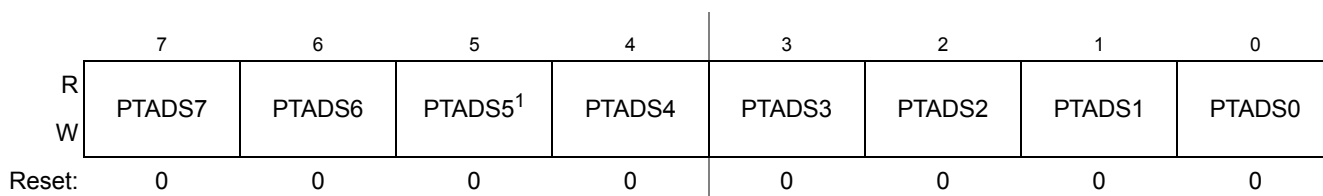
¹ PTASE5 does not affect the input only PTA5 pin.

Figure 11-5. Slew Rate Enable for Port A Register (PTASE)

Table 11-4. PTASE Register Field Descriptions

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

11.4.2 Port A Drive Strength Selection Register (PTADS)



¹ PTADS5 does not affect the input only PTA5 pin

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

Table 11-5. PTADS Register Field Descriptions

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

11.4.3 Port B Registers

Port B is controlled by the registers listed below.

11.4.3.1 Port B Data Register (PTBD)

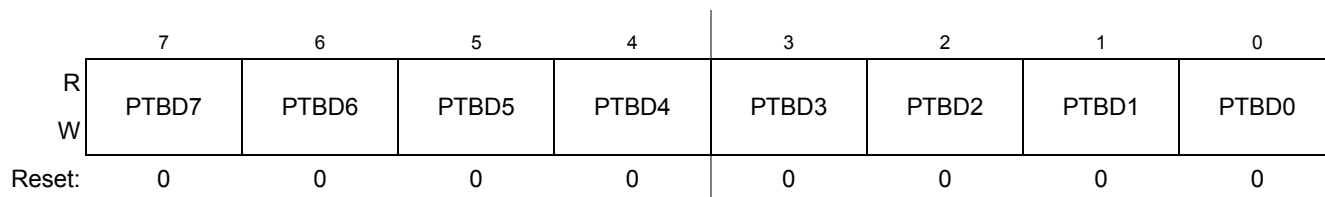


Figure 11-7. Port B Data Register (PTBD)

Table 11-6. PTBD Register Field Descriptions

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

11.4.3.2 Port B Data Direction Register (PTBDD)

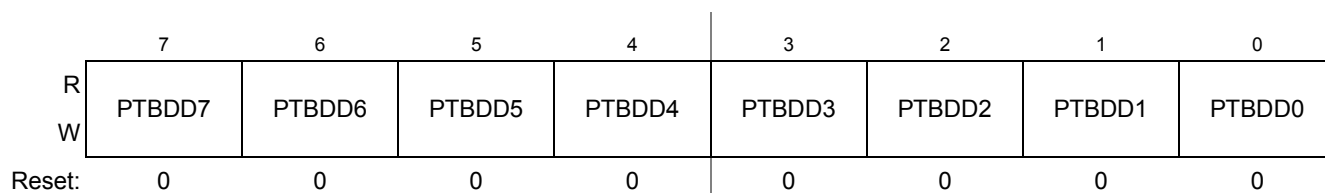


Figure 11-8. Port B Data Direction Register (PTBDD)

Table 11-7. PTBDD Register Field Descriptions

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

11.4.3.3 Port B Pull Enable Register (PTBPE)

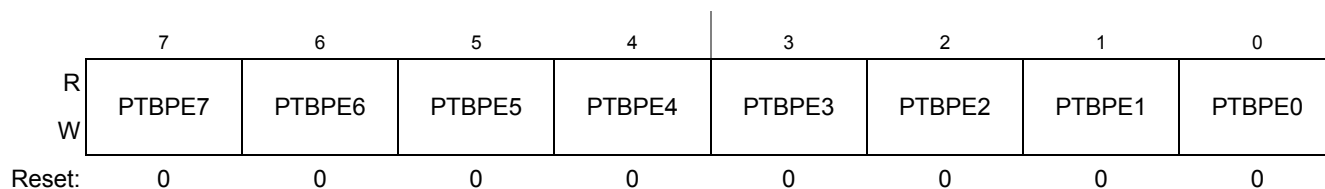


Figure 11-9. Internal Pull Enable for Port B Register (PTBPE)

Table 11-8. PTBPE Register Field Descriptions

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

11.4.3.4 Port B Slew Rate Enable Register (PTBSE)

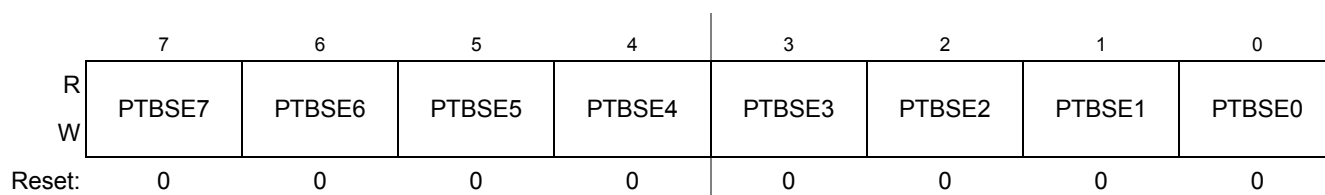


Figure 11-10. Slew Rate Enable for Port B Register (PTBSE)

Table 11-9. PTBSE Register Field Descriptions

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

11.4.3.5 Port B Drive Strength Selection Register (PTBDS)

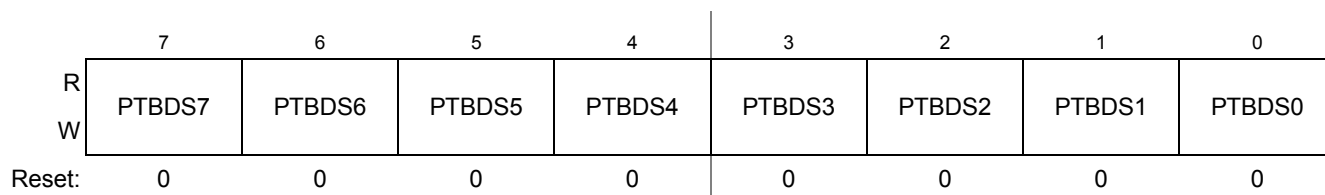


Figure 11-11. Drive Strength Selection for Port B Register (PTBDS)

Table 11-10. PTBDS Register Field Descriptions

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

11.4.4 Port C Registers

Port C is controlled by the registers listed below.

11.4.4.1 Port C Data Register (PTCD)

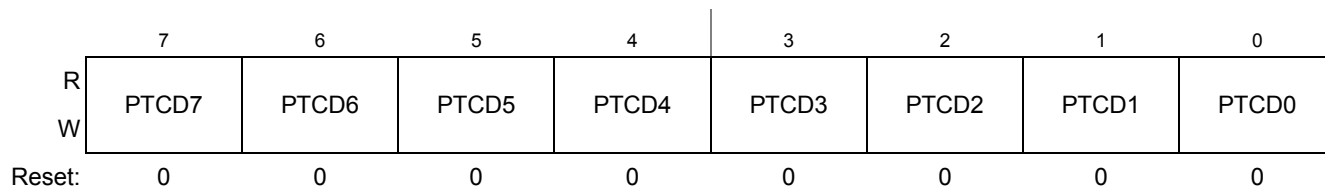


Figure 11-12. Port C Data Register (PTCD)

Table 11-11. PTCD Register Field Descriptions

Field	Description
7:0 PTCD[7:0]	Port C Data Register Bits — For port C pins that are inputs, reads return the logic level on the pin. For port C pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port C pins configured as outputs, the logic level is driven out 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.

11.4.4.2 Port C Data Direction Register (PTCDD)

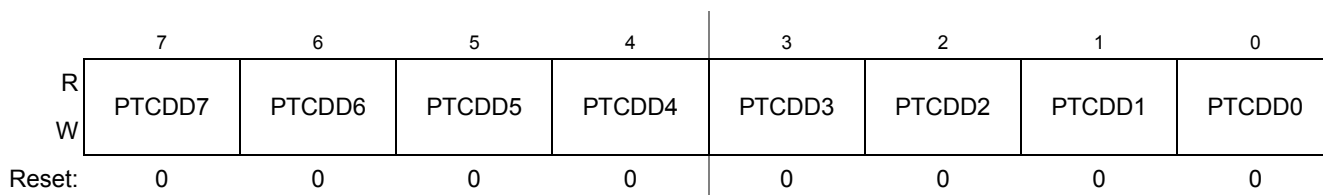


Figure 11-13. Port C Data Direction Register (PTCDD)

Table 11-12. PTCDD Register Field Descriptions

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

11.4.4.3 Port C Pull Enable Register (PTCPE)

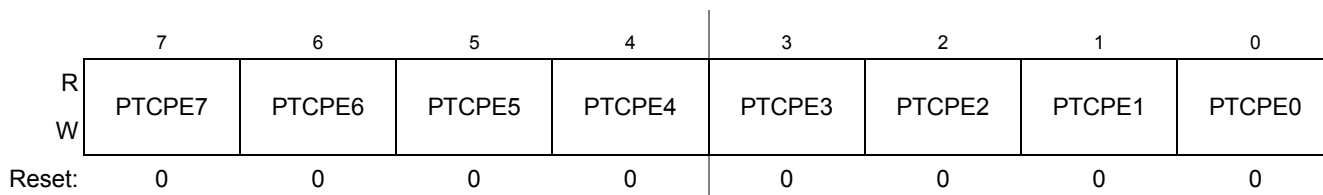


Figure 11-14. Internal Pull Enable for Port C Register (PTCPE)

Table 11-13. PTCPE Register Field Descriptions

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

11.4.4.4 Port C Slew Rate Enable Register (PTCSE)

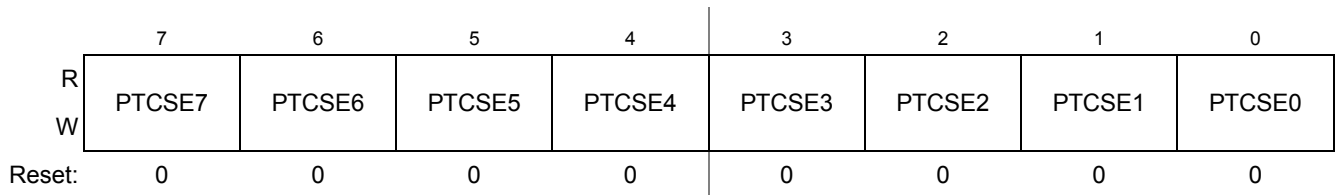


Figure 11-15. Slew Rate Enable for Port C Register (PTCSE)

Table 11-14. PTCSE Register Field Descriptions

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

11.4.4.5 Port C Drive Strength Selection Register (PTCDS)

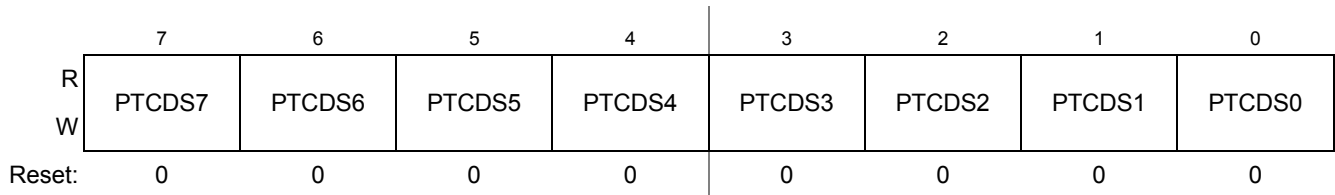


Figure 11-16. Drive Strength Selection for Port C Register (PTCDS)

Table 11-15. PTCDS Register Field Descriptions

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

11.4.5 Port D Registers

Port D is controlled by the registers listed below.

11.4.5.1 Port D Data Register (PTDD)

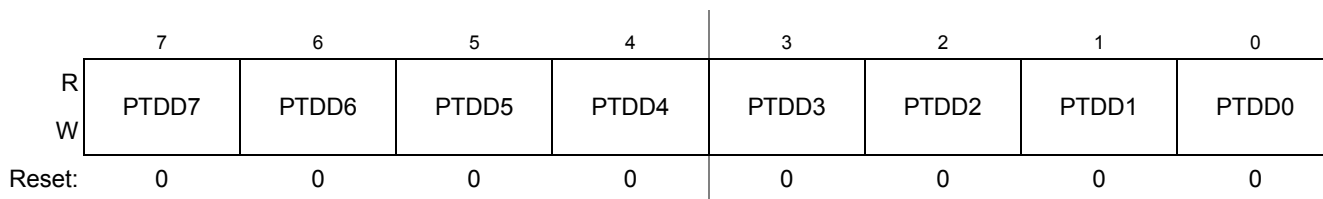


Figure 11-17. Port D Data Register (PTDD)

Table 11-16. PTDD Register Field Descriptions

Field	Description
7:0 PTDD[7:0]	Port D Data Register Bits — For port D pins that are inputs, reads return the logic level on the pin. For port D pins 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 configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups/pulldowns disabled.

11.4.5.2 Port D Data Direction Register (PTDDD)

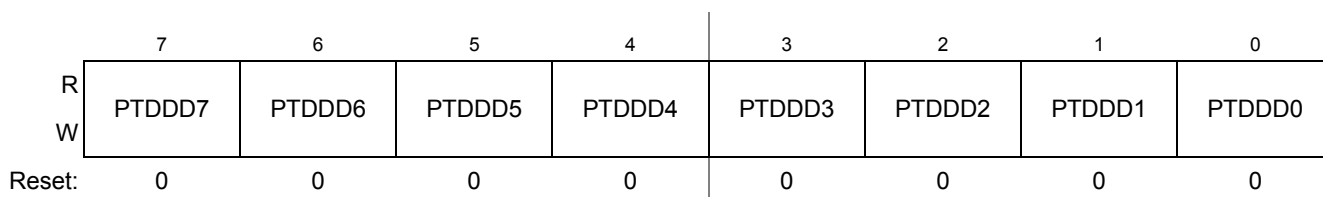


Figure 11-18. Port D Data Direction Register (PTDDD)

Table 11-17. PTDDD Register Field Descriptions

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

11.4.5.3 Port D Pull Enable Register (PTDPE)

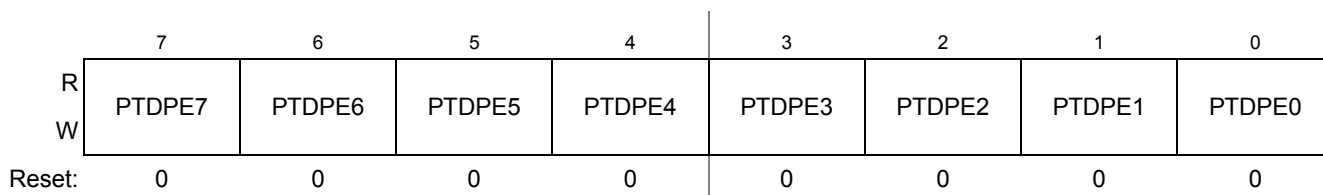


Figure 11-19. Internal Pull Enable for Port D Register (PTDPE)

Table 11-18. PTDPE Register Field Descriptions

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

11.4.5.4 Port D Slew Rate Enable Register (PTDSE)

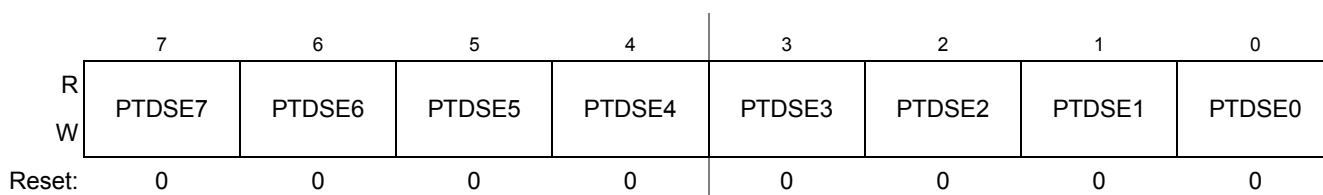


Figure 11-20. Slew Rate Enable for Port D Register (PTDSE)

Table 11-19. PTDSE Register Field Descriptions

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

11.4.5.5 Port D Drive Strength Selection Register (PTDDS)

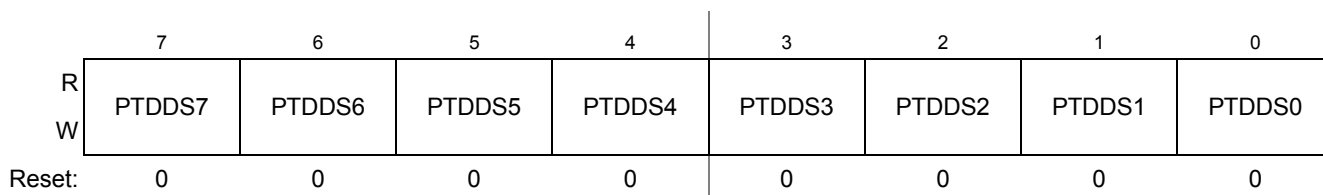


Figure 11-21. Drive Strength Selection for Port D Register (PTDDS)

Table 11-20. PTDDS Register Field Descriptions

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

11.4.6 Port E Registers

Port E is controlled by the registers listed below.

11.4.6.1 Port E Data Register (PTED)

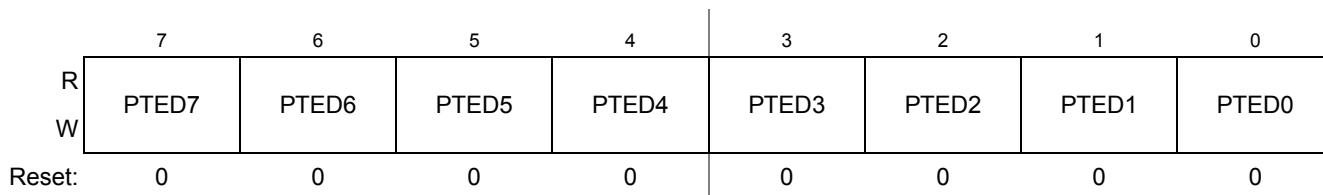


Figure 11-22. Port E Data Register (PTED)

Table 11-21. PTED Register Field Descriptions

Field	Description
7:0 PTED[7:0]	<p>Port E Data Register Bits — For port E pins that are inputs, reads return the logic level on the pin. For port E pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port E pins configured as outputs, the logic level is driven out the corresponding MCU pin.</p> <p>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.</p>

11.4.6.2 Port E Data Direction Register (PTEDD)

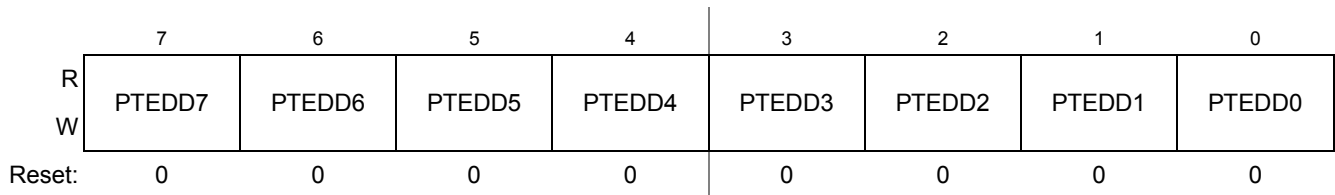


Figure 11-23. Port E Data Direction Register (PTEDD)

Table 11-22. PTEDD Register Field Descriptions

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

11.4.6.3 Port E Pull Enable Register (PTEPE)

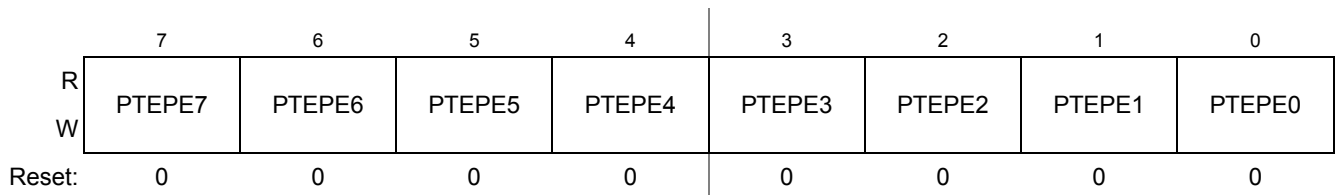


Figure 11-24. Internal Pull Enable for Port E Register (PTEPE)

Table 11-23. PTEPE Register Field Descriptions

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

11.4.6.4 Port E Slew Rate Enable Register (PTESE)

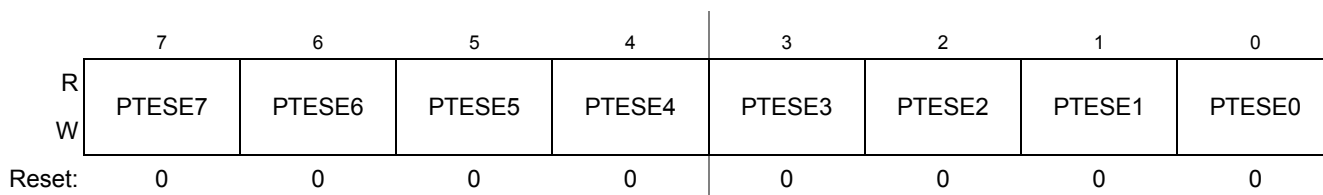


Figure 11-25. Slew Rate Enable for Port E Register (PTESE)

Table 11-24. PTESE Register Field Descriptions

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

11.4.6.5 Port E Drive Strength Selection Register (PTEDS)

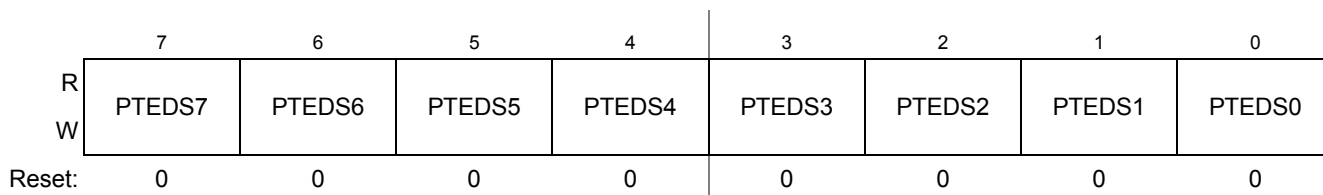


Figure 11-26. Drive Strength Selection for Port E Register (PTEDS)

Table 11-25. PTEDS Register Field Descriptions

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

Chapter 12

MCU Internal Clock Source (ICS)

12.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock.

The ICSTRM and FTRIM bits are normally reset to the factory trim values on any reset. However, any reset that puts the device into BDM (a POR with the BKGD pin held low or a development tool setting SBDFR[BDFR]) results in the ICSTRM and FTRIM bits being set to values of 0x80 and 0. When debugging the MCU, the factory trim value can be used by copying the trim values from the flash locations.

There are also signals provided to control a low power oscillator (XOSCVLP) module to allow the use of an external crystal/resonator as the external reference clock.

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

12.1.1 External Oscillator

The external oscillator module (XOSCVLP) provides the external clock options to the ICS module. The output of this submodule (OSCOUT) can be used as the real-time counter module (RTC) clock source.

12.1.2 Stop2 Mode Considerations

If you are using a low range oscillator during stop2, reconfigure the ICSC2 register (the oscillator control bits) before PPDACK is written. The low range (RANGE=0) oscillator can operate in stop2 to be the clock source for the RTC module. If the low range oscillator is active when entering stop2, it remains active in stop2 regardless of the value of EREFSTEN. To disable the oscillator in stop2, switch the ICS into FBI or FEI mode before executing the STOP instruction.

Whichever ICS clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

12.1.3 Features

Key features of the <<BLOCK NAME>> module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
 - 0.2% resolution using internal 32 kHz reference
 - 2% deviation over voltage and temperature using internal 32 kHz reference
- Internal or external reference clocks can be used to control the FLL

- Reference divider is provided for external clock
- Internal reference clock has 9 trim bits available
- Internal or external reference clocks can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
 - 2-bit select for clock divider is provided
 - Allowable dividers are: 1, 2, 4, 8
- Control signals for a low power oscillator clock generator (OSCOOUT) as the ICS external reference clock are provided
 - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL Engaged Internal mode is automatically selected out of reset
- BDC clock is provided as a constant divide by 2 of the low range DCO output
- Three selectable digitally-controlled oscillators (DCO) optimized for different frequency ranges.
- Option to maximize output frequency for a 32768 Hz external reference clock source.

12.1.4 Block Diagram

Figure 12-1 is the ICS block diagram.

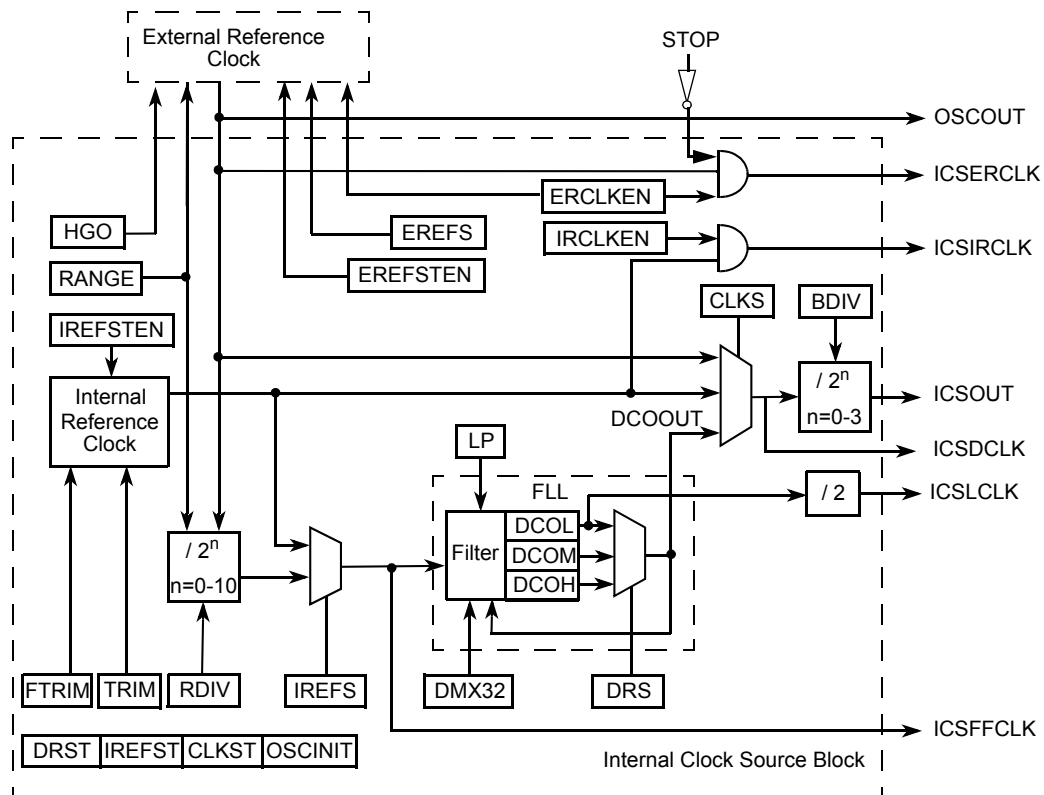


Figure 12-1. Internal Clock Source (ICS) Block Diagram

12.1.5 Modes of Operation

There are seven modes of operation for the <<BLOCK NAME>>: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

12.1.5.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock. The BDC clock is supplied from the FLL.

12.1.5.2 FLL Engaged External (FEE)

In FLL engaged external mode, the <<BLOCK NAME>> supplies a clock derived from the FLL which is controlled by an external reference clock source. The BDC clock is supplied from the FLL.

12.1.5.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The <<BLOCK NAME>> supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

12.1.5.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the <<BLOCK NAME>> supplies a clock derived from the internal reference clock. The BDC clock is not available.

12.1.5.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The <<BLOCK NAME>> supplies a clock derived from the external reference clock source. The BDC clock is supplied from the FLL.

12.1.5.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the <<BLOCK NAME>> supplies a clock derived from the external reference clock. The BDC clock is not available.

12.1.5.7 Stop (STOP)

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

NOTE

The DCO frequency changes from the pre-stop value to its reset value and the FLL will need to re-acquire the lock before the frequency is stable. Timing sensitive operations should wait for the FLL acquisition time, $t_{Acquire}$, before executing.

12.1.5.8 Test Mode (TM)

In Test Mode, all other modes of operation are available and the DCO filter is directly controllable

12.2 External Signal Description

There are no ICS signals that connect off chip.

12.3 Register Definition

Figure 12-1 is a summary of ICS registers.

Table 12-1. <<BLOCK NAME>> Register Summary

Name		7	6	5	4	3	2	1	0
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
	W								
ICSC2	R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
	W								
ICSTRM	R	TRIM							
	W								
ICSSC	R	DRST	DMX32	IREFST	CLKST		OSCINIT	FTRIM	
	W	DRS							
ICST	R	TEST		BDIVEN					
	W								

12.3.1 ICS Control Register 1 (ICSC1)

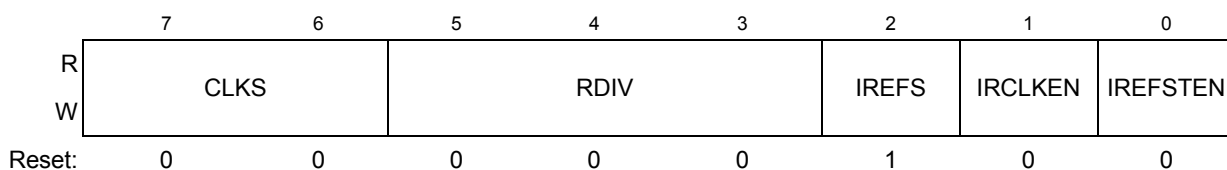


Figure 12-2. ICS Control Register 1 (ICSC1)

Table 12-2. ICS Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	Clock Source Select — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	Reference Divider — Selects the amount to divide down the external reference clock. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. See Table 12-3 for the divide-by factors.
2 IREFS	Internal Reference Select — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected. 0 External reference clock selected.
1 IRCLKEN	Internal Reference Clock Enable — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active. 0 ICSIRCLK inactive.
0 IREFSTEN	Internal Reference Stop Enable — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set before entering stop. 0 Internal reference clock is disabled in stop.

Table 12-3. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
0	1 ¹	32
1	2	64
2	4	128
3	8	256
4	16	512
5	32	1024
6	64	Reserved
7	128	Reserved

¹ Reset default

* Note: When RANGE=1 and when RDIV values equal to 6 or 7, the Reference Divider Factor will default to 1024

12.3.2 ICS Control Register 2 (ICSC2)

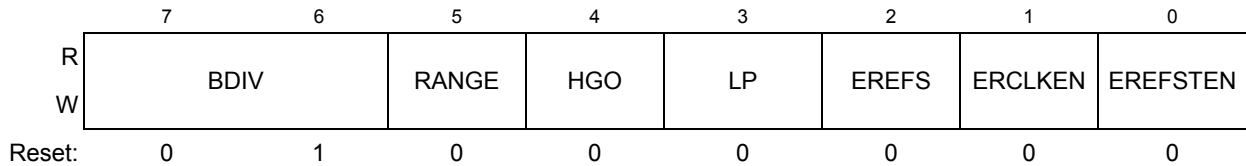
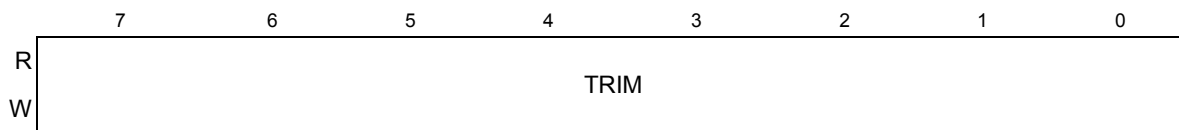


Figure 12-3. ICS Control Register 2 (ICSC2)

Table 12-4. ICS Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	<p>Bus Frequency Divider — Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency.</p> <p>00 Encoding 0 — Divides selected clock by 1. 01 Encoding 1 — Divides selected clock by 2 (reset default). 10 Encoding 2 — Divides selected clock by 4. 11 Encoding 3 — Divides selected clock by 8.</p>
5 RANGE	<p>Frequency Range Select — Selects the frequency range for the external oscillator ics_bypass_mdswitch.asm.</p> <p>1 High frequency range selected for the external oscillator ics_bypass_mdswitch.asm. 0 Low frequency range selected for the external oscillator ics_bypass_mdswitch.asm.</p>
4 HGO	<p>High Gain Oscillator Select — The HGO bit controls the external oscillator mode of operation.</p> <p>1 Configure external oscillator for high gain operation. 0 Configure external oscillator for low power operation.</p>
3 LP	<p>Low Power Select — The LP bit controls whether the FLL is disabled in FLL bypassed modes.</p> <p>1 FLL is disabled in bypass modes unless BDM is active. 0 FLL is not disabled in bypass mode.</p>
2 EREFS	<p>External Reference Select — The EREFS bit selects the source for the external reference clock ics_bypass_mdswitch.asm.</p> <p>1 Oscillator requested ics_bypass_mdswitch.asm. 0 External Clock Source requested ics_bypass_mdswitch.asm.</p>
1 ERCLKEN	<p>External Reference Enable — The ERCLKEN bit enables the external reference clock for use as IC SERCLK.</p> <p>1 IC SERCLK active. 0 IC SERCLK inactive.</p>
0 EREFSTEN	<p>External Reference Stop Enable — The EREFSTEN bit controls whether or not the external reference clock source (OSCOU) remains enabled when the ICS enters stop mode.</p> <p>1 External reference clock source stays enabled in stop if ERCLKEN is set before entering stop. 0 External reference clock source is disabled in stop.</p>

12.3.3 ICS Trim Register (ICSTRM)



Reset: Note: TRIM is loaded during reset from a factory programmed location when not in BDM mode. If in a BDM mode, a default value of 0x80 is loaded.

Figure 12-4. ICS Trim Register (ICSTRM)

Table 12-5. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	<p>ICS Trim Setting — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (in other words, bit 1 adjusts twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period.</p> <p>An additional fine trim bit is available in ICSSC as the FTRIM bit.</p>

12.3.4 ICS Status and Control (ICSSC)

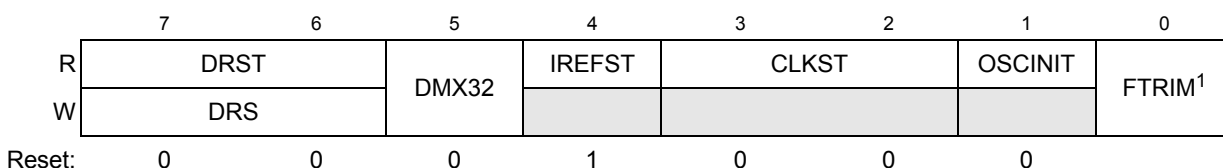


Figure 12-5. ICS Status and Control Register (ICSSC)

¹ FTRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, FTRIM gets loaded with a value of 1'b0.

Table 12-6. ICS Status and Control Register Field Descriptions

Field	Description
7-6 DRST DRS	<p>DCO Range Status — The DRST read field indicates the current frequency range for the FLL output, DCOOUT. See Table 12-7. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. Writing the DRS bits to 2'b11 is ignored and the DRST bits remain with the current setting.</p> <p>DCO Range Select — The DRS field selects the frequency range for the FLL output, DCOOUT. Writes to the DRS field while the LP bit is set are ignored ics_fbilp_mdswitch.asm, ics_fbelp_mdswitch.asm.</p> <p>00 Low range. 01 Mid range. 10 High range. 11 Reserved.</p>
5 DMX32	<p>DCO Maximum frequency with 32.768 kHz reference — The DMX32 bit controls whether or not the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. See Table 12-7.</p> <p>0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.</p>

Table 12-6. ICS Status and Control Register Field Descriptions (continued)

Field	Description
4 IREFST	Internal Reference Status — The IREFST bit indicates the current source for the reference clock ics_clkst_mdswitch.asm . The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains ics_clkst_mdswitch.asm . 0 Source of reference clock is external clock ics_clkst_mdswitch.asm . 1 Source of reference clock is internal clock ics_clkst_mdswitch.asm .
3-2 CLKST	Clock Mode Status — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, Internal reference clock is selected. 10 FLL Bypassed, External reference clock is selected. 11 Reserved.
1 OSCINIT	OSC Initialization — If the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared.
0 FTRIM	ICS Fine Trim — The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible.

Table 12-7. DCO frequency range¹

DRS	DMX32	Reference range	FLL factor	DCO range
00	0	31.25 - 39.0625 kHz	512	16 - 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	0	31.25 - 39.0625 kHz	1024	32 - 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	0	31.25 - 39.0625 kHz	1536	48 - 60 MHz
	1	32.768 kHz	1824	59.77 MHz
11	Reserved			

¹ The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.

12.3.5 ICS Test Register (ICST)

This register has a separate module select signal so it can be mapped anywhere in the MCU memory map.

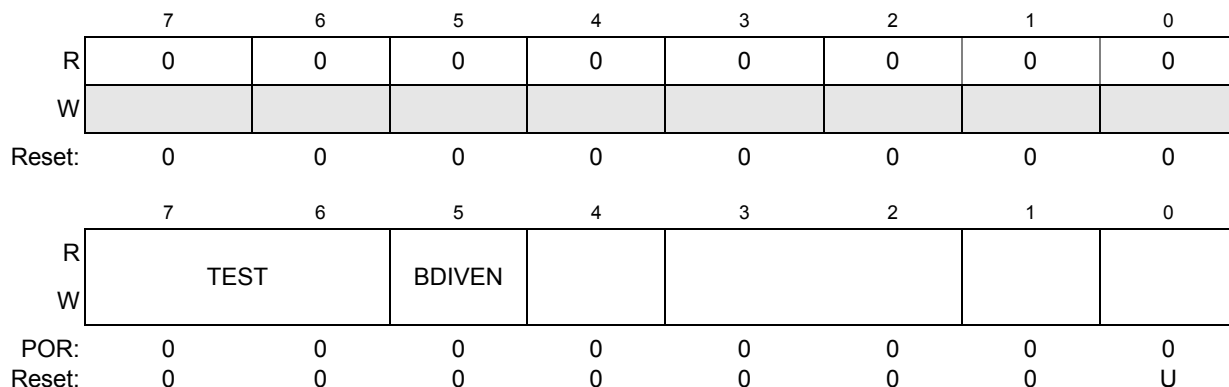


Figure 12-6. ICS Test Register (ICST)

Table 12-8. ICS Test Register Field Descriptions

Field	Description
7:6 TEST	Test Mode Select —Selects the test function for the FLL. 00 FLL is in functional mode. 01 FLL filter driven to 9'b000000000. 10 FLL filter driven to 9'b111111111. 11 FLL filter is driven to value of TRIM in ICSTRM register.
5 BDIVEN	BDIV Clock Out Select —Padi control signal to request BDIV clock signal as pad output.
4:0	Reserved

12.4 Functional Description

12.4.1 Operational Modes

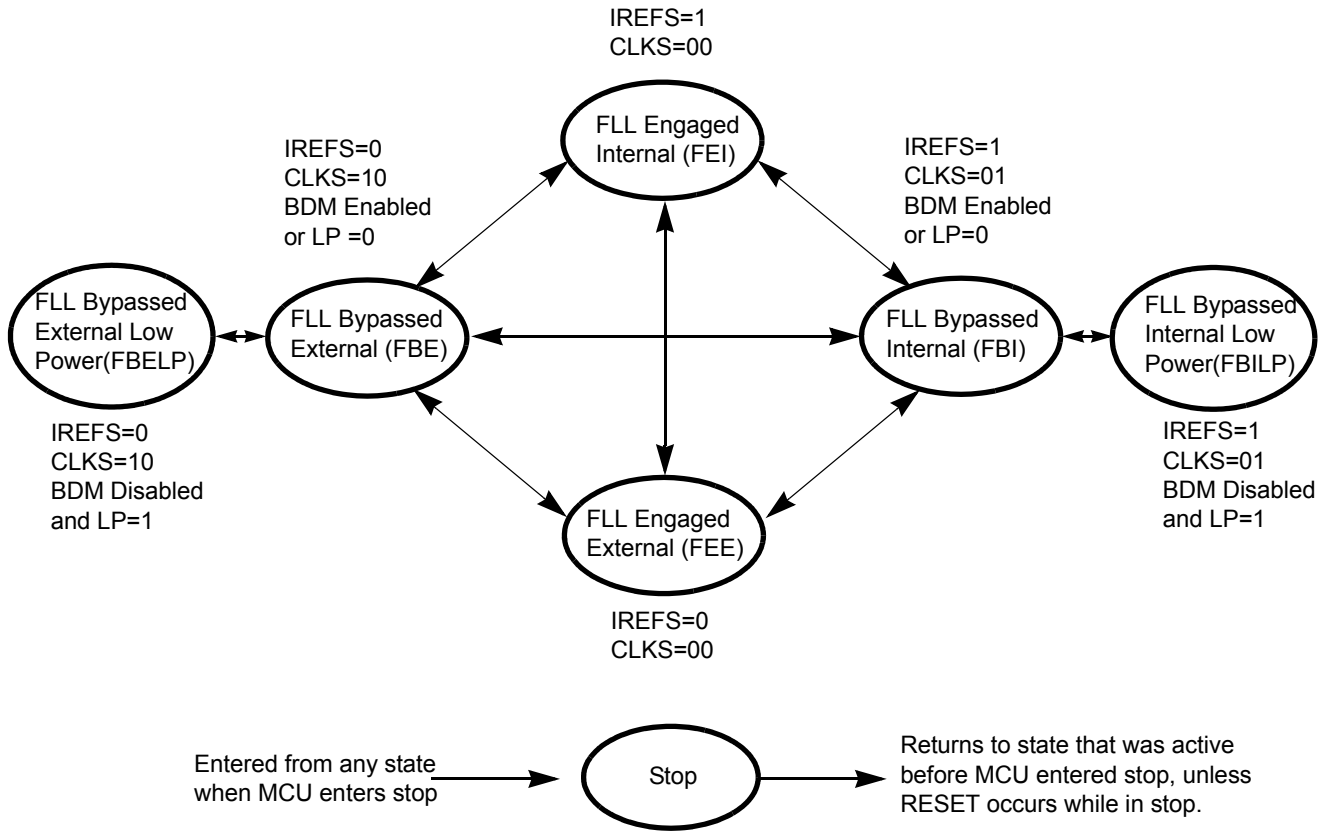


Figure 12-7. Clock Switching Modes

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

12.4.1.1 FLL Engaged Internal (FEI)

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

- CLKS bits are written to 00.
- IREFS bit is written to 1.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop locks the frequency to the FLL factor times the internal reference frequency. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

12.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock source. The FLL loop locks the frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

12.4.1.3 FLL Bypassed Internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the internal reference frequency. The ICSLCLK will be available for BDC communications, and the internal reference clock is enabled.

12.4.1.4 FLL Bypassed Internal Low Power (FBILP)

The FLL bypassed internal low power (FBILP) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed internal low power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The ICSLCLK will be not be available for BDC communications, and the internal reference clock is enabled.

12.4.1.5 FLL Bypassed External (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock source. The FLL clock is controlled by the external reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits, so that the ICSLCLK will be available for BDC communications, and the external reference clock is enabled.

12.4.1.6 FLL Bypassed External Low Power (FBELP)

The FLL bypassed external low power (FBELP) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock source and the FLL is disabled. The ICSLCLK will be not be available for BDC communications. The external reference clock source is enabled.

12.4.1.7 Stop

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

ICSIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1.
- IREFSTEN bit is written to 1.

OSCOUT will be active in stop mode when all the following conditions occur:

- ERCLKEN bit is written to 1.
- EREFSTEN bit is written to 1.

12.4.1.8 Test Mode (TM)

In Test Mode the DCO filter is controllable by driving the filter value to maximum or minimum value, and by controlling it directly using the TRIM register value.

- The output of the Bus Frequency Divider can also be used as pad outputs in test mode in order to allow direct observation of these clock signal.

12.4.2 Mode Switching

The IREF bit can be changed at anytime, but the actual switch to the newly selected clock is shown by the IREFST bit. . When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes, the FLL begins locking again after the switch is completed.

The CLKS bits can also be changed at anytime, but the actual switch to the newly selected clock is shown by the CLKST bits. If the newly selected clock is not available, the previous clock remains selected.

The DRS bits can be changed at anytime except when LP bit is 1. If the DRS bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the bus clock remains at the previous DCO range until the new DCO starts. When the new DCO starts the bus clock switches to it. After switching to the new DCO the FLL remains unlocked for several reference cycles. Once the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the DRST bits.

12.4.3 Bus Frequency Divider

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

12.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. The DRS bits can not be written while LP bit is 1.

However, in some applications it may be desirable to allow the FLL to be enabled and to lock for maximum accuracy before switching to an FLL engaged mode. To do this, write the LP bit to 0.

12.4.5 DCO Maximum Frequency with 32.768 kHz Oscillator

The FLL has an option to change the clock multiplier for the selected DCO range such that it results in the maximum bus frequency with a common 32.768 kHz crystal reference clock.

12.4.6 Internal Reference Clock

This module can support an internal reference clock with frequencies between 31.25kHz - 5MHz

When IRCLKEN is set the internal reference clock signal is presented as ICSIRCLK, which can be used as an additional clock source. To re-target the ICSIRCLK frequency, write a new value to the TRIM bits in the ICSTRM register to trim the period of the internal reference clock:

- Writing a larger value slows down the ICSIRCLK frequency.
- Writing a smaller value to the ICSTRM register speeds up the ICSIRCLK frequency.

The TRIM bits effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the [Device Overview](#) chapter).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock keeps running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value is uploaded to the ICSTRM register and ICS FTRIM register during any reset initialization. For finer precision, trim the internal oscillator in the application and set the FTRIM bit accordingly.

12.4.7 External Reference Clock

The ICS module supports an external reference clock with frequencies between 31.25 kHz to 40 MHz in all modes. When the ERCLKEN is set, the external reference clock signal is presented as ICSECLK, which can be used as an additional clock source in run mode. When IREFS = 1, the external reference clock is not used by the FLL and will only be used as ICSECLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications support (see the [Device Overview](#) chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock source (OSCOUT) keeps running during stop mode in order to provide a fast recovery upon exiting stop.

12.4.8 Fixed Frequency Clock

The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source. ICSFFCLK frequency must be no more than 1/4 of the ICSOUT frequency to be valid. When ICSFFCLK is valid, ICS output signal (ICSFFE) gets asserted high. Because of this requirement, in bypass modes the ICSFFCLK is valid only in bypass external modes (FBE and FBELP) for the following combinations of BDIV, RDIV and RANGE values:

- RANGE=1
- BDIV=00 (divide by 1), RDIV ≥ 010
- BDIV=01 (divide by 2), RDIV ≥ 011
- BDIV=10 (divide by 4), RDIV ≥ 100
- BDIV=11 (divide by 8), RDIV ≥ 101

12.4.9 BDC Clock Operation

When the bdm_enable signal is high, the <<BLOCK NAME>> will generate a clock (ICS8MCLK). This clock will be muxed with CORECLK (the core clock from the SIM module) and the output of this mux (ICSLCLK) will be provided for system communication with the Background Debug Controller (BDC). The mux will select CORECLK during reset or CLKSW is high, and it will select ICS8MCLK when CLKSW is low and not in reset. The frequency of ICS8MCLK is targeted at 512 times the filter frequency(rdiv_clk).

12.4.10 Scan Coverage

The ICS hardblock will contain a ripple counter to be used for the reference divider. This counter will not be scannable. The operation of the ripple counter can be tested by observing the RDIV output clock signal.

The ICS hardblock will also contain a ripple counter to be used for the bus frequency divider. This counter will not be scannable. The operation of this ripple counter can be tested by observing the BDIV output clock signal. There will be additional flip-flops in the ICS hardblock and they will be included on a scan chain available for chip level scan testing.

12.5 S08ICSV2 Electricals

12.5.1 External Oscillator (XOSC) and Internal Clock Source (ICS) Characteristics

See [Section 3.4.1.1, “Crystal Resonator Specification”](#) for crystal resonator circuit.

Table 12-9. XOSC and ICS Specifications (Temperature Range = –40 to 85°C Ambient)

Characteristic	Symbol	Min	Typ ¹	Max	Unit
Oscillator crystal or resonator (EREFS = 1, ERCLKEN = 1)					
Low range (RANGE = 0)	f_{lo}	32	—	38.4	kHz
High range (RANGE = 1) FEE or FBE mode ²	f_{hi}	1	—	5	MHz
High range (RANGE = 1), high gain (HGO = 1), FBELP mode	f_{hi}	1	—	16	MHz
High range (RANGE = 1), low power (HGO = 0), FBELP mode	f_{hi}	1	—	8	MHz
Load capacitors	C_1 C_2	See Note ³			
Feedback resistor					
Low range (32 kHz to 38.4 kHz)	R_F		10		MΩ
High range (1 MHz to 16 MHz)			1		MΩ
Series resistor — Low range					
Low Gain (HGO = 0)	R_S	—	0	—	kΩ
High Gain (HGO = 1)		—	100	—	
Series resistor — High range					
Low Gain (HGO = 0)	R_S	—	0	0	kΩ
High Gain (HGO = 1)		—	0	10	
≥ 8 MHz		—	0	20	
4 MHz		—	0		
1 MHz		—	0		
Crystal start-up time ^{4, 5}					
Low range, low power	t_{CSTL}	—	200	—	ms
Low range, high power		—	400	—	
High range, low power	t_{CSTH}	—	5	—	
High range, high power		—	15	—	
Internal reference start-up time	t_{IRST}	—	60	100	μs
Square wave input clock frequency (EREFS = 0, ERCLKEN = 1)					
FEE or FBE mode ²	f_{extal}	0.03125	—	5	MHz
FBELP mode		0	—	40	MHz
Average internal reference frequency - trimmed	f_{int_t}	—	32.768	—	kHz
DCO output frequency range - trimmed ⁶	f_{dco_t}	32	—	40	MHz

Table 12-9. XOSC and ICS Specifications (Temperature Range = -40 to 85°C Ambient)

Characteristic		Symbol	Min	Typ ¹	Max	Unit
DCO output frequency range -trimmed	Low range (DRS=00)	f_{dco_t}	16	—	20	MHz
	Mid Range (DRS=10)		32	—	40	
	High range (DRS=10)		48	—	60	
Total deviation of DCO output from trimmed frequency ⁴ Over full voltage and temperature range Over fixed voltage and temperature range of 0 to 70°C		Δf_{dco_t}	—	-1.0 to +0.5 ± 0.5	± 2 ± 1	% f_{dco}
FLL acquisition time ^{4,7}		$t_{Acquire}$			1	ms
Long term jitter of DCO output clock (averaged over 2-ms interval) ⁸		C_{Jitter}	—	0.02	0.2	% f_{dco}

- ¹ Data in Typical column was characterized at 3.0 V, 25°C or is typical recommended value.
- ² When ICS is configured for FEE or FBE mode, input clock source must be divisible using RDIV to within the range of 31.25 kHz to 39.0625 kHz.
- ³ See crystal or resonator manufacturer’s recommendation.
- ⁴ This parameter is characterized and not tested on each device.
- ⁵ Proper PC board layout procedures must be followed to achieve specifications.
- ⁶ The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.
- ⁷ This specification applies to any time the FLL reference source or reference divider is changed, trim value changed, DMX32 bit is changed, DRS bit is changed, or changing from FLL disabled (FBELP, FBILP) to FLL enabled (FEI, FEE, FBE, FBI). If a crystal/resonator is being used as the reference, this specification assumes it is already running.
- ⁸ Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum f_{BUS} . Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the FLL circuitry via V_{DD} and V_{SS} and variation in crystal oscillator frequency increase the C_{Jitter} percentage for a given interval.

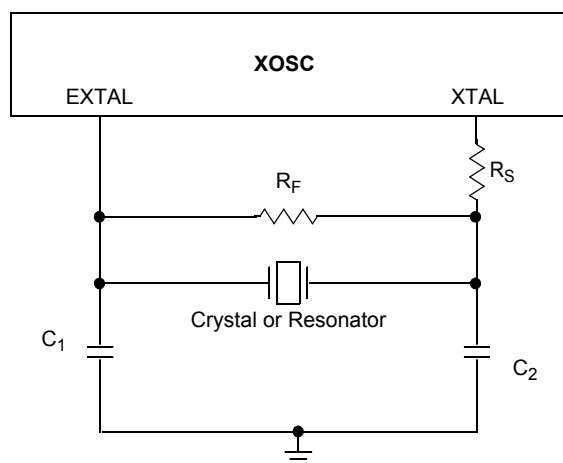


Figure 0-1. Typical Crystal or Resonator Circuit

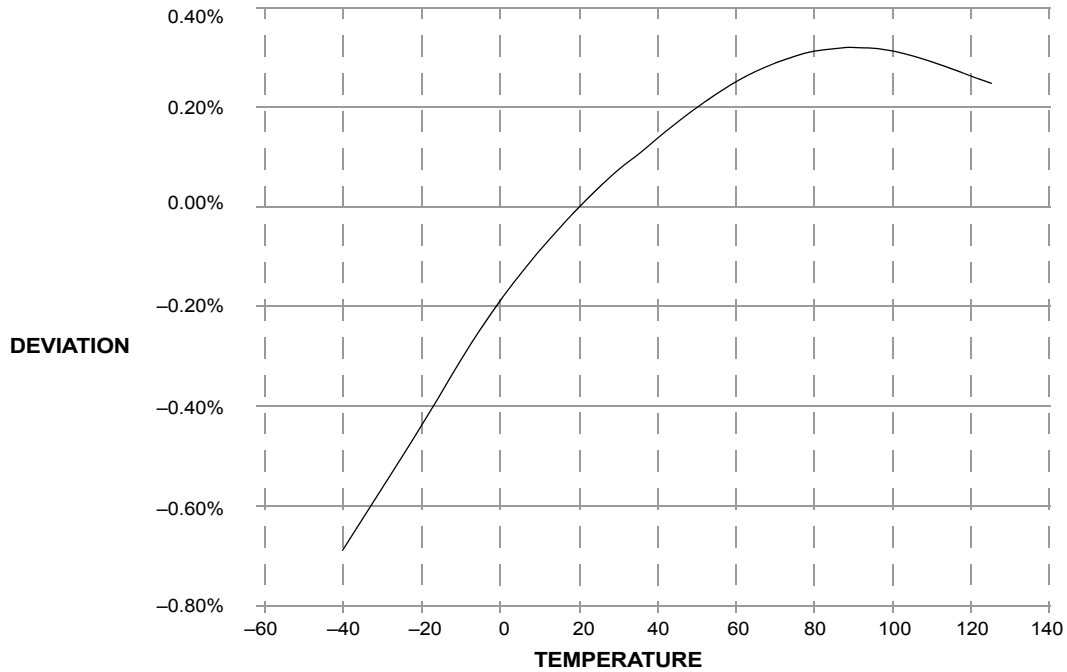


Figure 12-8. Deviation of DCO Output from Trimmed Frequency (16 MHz, 3.6 V)

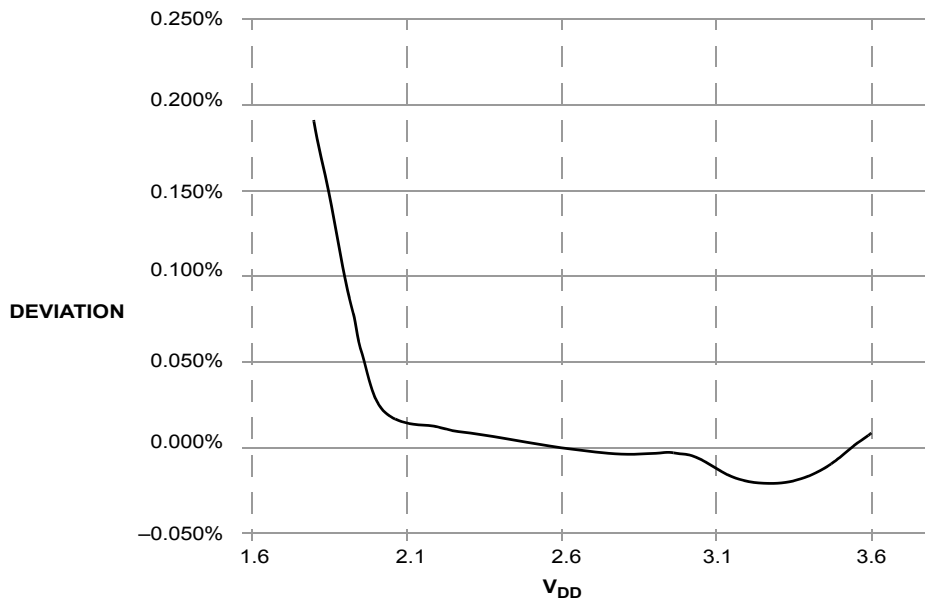


Figure 12-9. Deviation of DCO Output from Trimmed Frequency (16 MHz, 25°C)

12.5.2 Local Clock

The ICS presents the low range DCO output clock divided by two as ICSLCLK for use as a clock source for BDC communications. ICSLCLK is not available in FLL bypassed internal low power (FBILP) and FLL bypassed external low power (FBELP) modes.



Chapter 13

Central Processor UnitS08CPUV4

13.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the QE32 Family. For a more detailed discussion, refer to the *QE32 Family Reference Manual, volume 1*, Freescale Semiconductor document order number HCS08QE32RMV1/D.

The QE32 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).

13.1.1 Features

Features of the QE32 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
- 64-KB CPU address space with banked memory management unit for greater than 64 KB
- 16-bit stack pointer (any size stack anywhere in 64-KB CPU address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
 - Inherent — Operands in internal registers
 - Relative — 8-bit signed offset to branch destination
 - Immediate — Operand in next object code byte(s)
 - Direct — Operand in memory at 0x0000–0x00FF
 - Extended — Operand anywhere in 64-Kbyte address space
 - Indexed relative to H:X — Five submodes including auto increment
 - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions

- STOP and WAIT instructions to invoke low-power operating modes

13.2 Programmer's Model and CPU Registers

Figure 13-1 shows the five CPU registers. CPU registers are not part of the memory map.

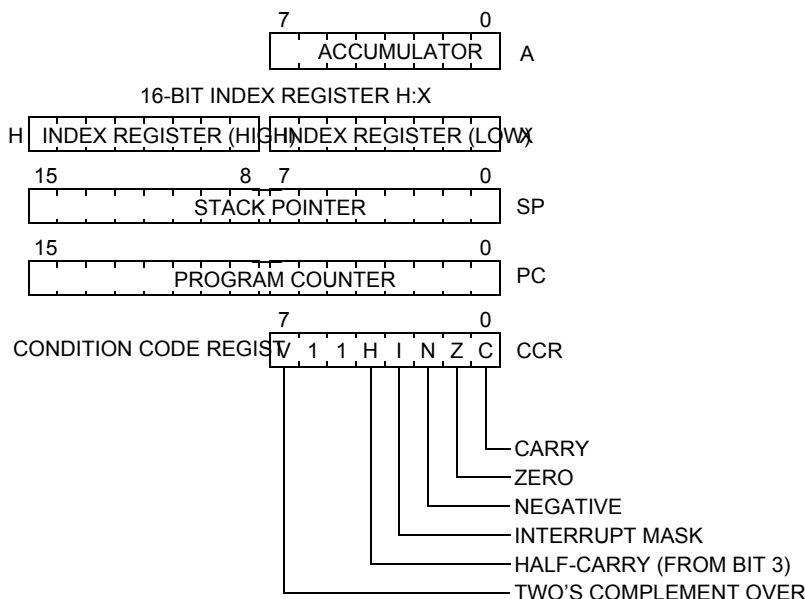


Figure 13-1. CPU Registers

13.2.1 Accumulator (A)

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

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

13.2.2 Index Register (H:X)

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

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

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

13.2.3 Stack Pointer (SP)

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

SP is forced to 0x00FF at reset for compatibility with the earlier M68HC05 Family. QE32 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 QE32 programs because it only affects the low-order half of the stack pointer.

13.2.4 Program Counter (PC)

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

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

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

13.2.5 Condition Code Register (CCR)

The 8-bit condition code register contains the interrupt mask (I) and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to 1. The following paragraphs describe the functions of the condition code bits in general terms. For a more detailed explanation of how each instruction sets the CCR bits, refer to the *QE32 Family Reference Manual, volume 1*, Freescale Semiconductor document order number HCS08RMv1.

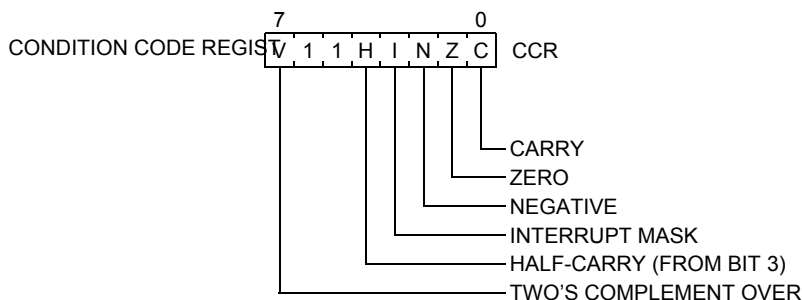


Figure 13-2. Condition Code Register

Table 13-1. CCR Register Field Descriptions

Field	Description
7 V	Two's Complement Overflow Flag — 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	Half-Carry Flag — 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	Interrupt Mask Bit — 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	Negative Flag — 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	Zero Flag — 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	Carry/Borrow Flag — 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

13.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the QE32, memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte CPU address space. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

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

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

13.3.1 Inherent Addressing Mode (INH)

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

13.3.2 Relative Addressing Mode (REL)

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

13.3.3 Immediate Addressing Mode (IMM)

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

13.3.4 Direct Addressing Mode (DIR)

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

13.3.5 Extended Addressing Mode (EXT)

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

13.3.6 Indexed Addressing Mode

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

13.3.6.1 Indexed, No Offset (IX)

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

13.3.6.2 Indexed, No Offset with Post Increment (IX+)

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

13.3.6.3 Indexed, 8-Bit Offset (IX1)

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

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

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

13.3.6.5 Indexed, 16-Bit Offset (IX2)

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

13.3.6.6 SP-Relative, 8-Bit Offset (SP1)

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

13.3.6.7 SP-Relative, 16-Bit Offset (SP2)

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

13.4 Special Operations

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

13.4.1 Reset Sequence

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

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

13.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.
3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.

6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

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

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

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

13.4.3 Wait Mode Operation

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

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

13.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the QE32 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 QE32 and whether the oscillator was stopped in stop mode. Refer to the [Modes of Operation](#) chapter for more details.

13.4.5 BGND Instruction

The BGND instruction is new to the QE32 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

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

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

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

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

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

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

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

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

1. The return value of the 8-bit PPAGE register is pulled from the stack.
2. The 16-bit return address is pulled from the stack and loaded into the PC.
3. The return PPAGE value is written to the PPAGE register.

4. The queue is refilled and execution begins at the new address.

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

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

13.5 QE32 Instruction Set Summary

Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in [Table 13-2](#).

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
- pg = Page
- 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 QE32 has a 16-bit address bus, this can be either a signed or an unsigned value.
- page* — Any label or expression that evaluates to a valid bank number for the PPAGE register. For a 128-Kbyte derivative, any value between 0 and 7 is valid.
- 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 13-2. QE32 Instruction Set Summary (Sheet 1 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
ADC # <i>opr8i</i> ADC <i>opr8a</i> ADC <i>opr16a</i> ADC <i>opr16,X</i> ADC <i>opr8,X</i> ADC <i>,X</i> ADC <i>opr16,SP</i> ADC <i>opr8,SP</i>	Add with Carry	$A \leftarrow (A) + (M) + (C)$	p	p	-	p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 dd C9 hh ll D9 ee ff E9 ff F9 9ED9 ee ff 9EE9 ff	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
ADD # <i>opr8i</i> ADD <i>opr8a</i> ADD <i>opr16a</i> ADD <i>opr16,X</i> ADD <i>opr8,X</i> ADD <i>,X</i> ADD <i>opr16,SP</i> ADD <i>opr8,SP</i>	Add without Carry	$A \leftarrow (A) + (M)$	p	p	-	p	p	p	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB dd CB hh ll DB ee ff EB ff FB 9EDB ee ff 9EEB ff	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4

Table 13-2. QE32 Instruction Set Summary (Sheet 2 of 7)

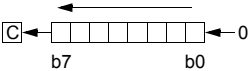
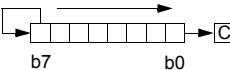
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
AIS <i>#opr8i</i>	Add Immediate Value (Signed) to Stack Pointer	$SP \leftarrow (SP) + (M)$ M is sign extended to a 16-bit value	-	-	-	-	-	-	IMM	A7	ii	2
AIX <i>#opr8i</i>	Add Immediate Value (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 <i>#opr8i</i> AND <i>opr8a</i> AND <i>opr16a</i> AND <i>opr16,X</i> AND <i>opr8,X</i> AND <i>,X</i> AND <i>opr16,SP</i> AND <i>opr8,SP</i>	Logical AND	$A \leftarrow (A) \& (M)$	0	-	-	p	p	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 C4 D4 E4 F4 9ED4 9EE4	ii dd hh ll ee ff ff ff ff ff	2 3 4 4 3 3 5 4
ASL <i>opr8a</i> ASLA ASLX ASL <i>opr8,X</i> ASL <i>,X</i> ASL <i>opr8,SP</i>	Arithmetic Shift Left (Same as LSL)		p	-	-	p	p	p	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
ASR <i>opr8a</i> ASRA ASRX ASR <i>opr8,X</i> ASR <i>,X</i> ASR <i>opr8,SP</i>	Arithmetic Shift Right		p	-	-	p	p	p	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	5 1 1 5 4 6
BCC <i>rel</i>	Branch if Carry Bit Clear	Branch if (C) = 0	-	-	-	-	-	-	REL	24	rr	3
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

Table 13-2. QE32 Instruction Set Summary (Sheet 3 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
BIT #opr8i BIT opr8a BIT opr16a BIT oprx16,X BIT oprx8,X BIT ,X BIT oprx16,SP BIT oprx8,SP	Bit Test	(A) & (M) (CCR Updated but Operands Not Changed)	0	-	-	p	p	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee 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 ⊕ 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 ⊕ 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
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear	Branch if (Mn) = 0	-	-	-	-	-	p	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 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) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory	Mn ← 1	-	-	-	-	-	-	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 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
CALL <i>page, opr16a</i>	Call Subroutine	PC ← PC + 4 Push (PCL); SP ← (SP) - 0x0001 Push (PCH); SP ← (SP) - 0x0001 Push (PPAGE); SP ← (SP) - 0x0001 PPAGE ← <i>page</i> PC ← Unconditional Address	-	-	-	-	-	-	EXT	AC	pghll	8

Table 13-2. QE32 Instruction Set Summary (Sheet 4 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
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 IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr rr ff rr	5 4 4 5 5 6
CLC	Clear Carry Bit	$C \leftarrow 0$	-	-	-	-	-	0	INH	98		1
CLI	Clear Interrupt Mask Bit	$I \leftarrow 0$	-	-	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	$M \leftarrow 0x00$ $A \leftarrow 0x00$ $X \leftarrow 0x00$ $H \leftarrow 0x00$ $M \leftarrow 0x00$ $M \leftarrow 0x00$ $M \leftarrow 0x00$	0	-	-	0	1	-	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd rr rr rr ff ff ff	5 1 1 1 5 4 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 DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 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)	$M \leftarrow (\overline{M}) = 0xFF - (M)$ $A \leftarrow (\overline{A}) = 0xFF - (A)$ $X \leftarrow (\overline{X}) = 0xFF - (X)$ $M \leftarrow (\overline{M}) = 0xFF - (M)$ $M \leftarrow (\overline{M}) = 0xFF - (M)$ $M \leftarrow (\overline{M}) = 0xFF - (M)$	0	-	-	p	p	1	DIR INH INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd hh ll ff ff ff ff	5 1 1 5 4 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 IMM DIR SP1	3E 65 75 9EF3	hh ll jj kk dd ff	6 3 5 6
CPX <i>#opr8i</i> CPX <i>opr8a</i> CPX <i>opr16a</i> CPX <i>opr16,X</i> CPX <i>opr8,X</i> CPX <i>,X</i> CPX <i>opr16,SP</i> CPX <i>opr8,SP</i>	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 ee ff ff	2 3 4 4 3 3 5 4
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	(A) ₁₀	U	-	-	p	p	p	INH	72		1
DBNZ <i>opr8a,rel</i> DBNZ A <i>rel</i> DBNZ X <i>rel</i> DBNZ <i>opr8,X,rel</i> DBNZ <i>,X,rel</i> DBNZ <i>opr8,SP,rel</i>	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 ff rr rr rr ff rr	7 4 4 7 6 8
DEC <i>opr8a</i> DECA DEC X DEC <i>opr8,X</i> DEC <i>,X</i> DEC <i>opr8,SP</i>	Decrement	$M \leftarrow (M) - 0x01$ $A \leftarrow (A) - 0x01$ $X \leftarrow (X) - 0x01$ $M \leftarrow (M) - 0x01$ $M \leftarrow (M) - 0x01$ $M \leftarrow (M) - 0x01$	p	-	-	p	p	-	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd rr rr ff ff ff	5 1 1 5 4 6
DIV	Divide	$A \leftarrow (H:A) \div (X)$ H ← Remainder	-	-	-	-	p	p	INH	52		6

Table 13-2. QE32 Instruction Set Summary (Sheet 5 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
EOR #opr8i EOR opr8a EOR opr16a EOR oprx16,X EOR oprx8,X EOR ,X EOR oprx16,SP EOR oprx8,SP	Exclusive OR Memory with Accumulator	$A \leftarrow (A \oplus M)$	0	-	-	p	p	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 B8 C8 D8 E8 F8 9ED8 9EE8	ii dd hh ll ff ff 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	$M \leftarrow (M) + 0x01$ $A \leftarrow (A) + 0x01$ $X \leftarrow (X) + 0x01$ $M \leftarrow (M) + 0x01$ $M \leftarrow (M) + 0x01$ $M \leftarrow (M) + 0x01$	p	-	-	p	p	-	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff ff ff	5 1 1 5 4 6
JMP opr8a JMP opr16a JMP oprx16,X JMP oprx8,X JMP ,X	Jump	$PC \leftarrow \text{Jump Address}$	-	-	-	-	-	-	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff ff	3 4 4 3 3
JSR opr8a JSR opr16a JSR oprx16,X JSR oprx8,X JSR ,X	Jump to Subroutine	$PC \leftarrow (PC) + n$ ($n = 1, 2, \text{ or } 3$) Push (PCL); $SP \leftarrow (SP) - 0x0001$ Push (PCH); $SP \leftarrow (SP) - 0x0001$ $PC \leftarrow \text{Unconditional Address}$	-	-	-	-	-	-	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff 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	$A \leftarrow (M)$	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 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	$H:X \leftarrow (M:M + 0x0001)$	0	-	-	p	p	-	IMM DIR EXT IX IX2 IX1 SP1	45 55 32 9EAE 9EBE 9ECE 9EFE	jj dd hh ll ff ff ff ff	3 4 5 5 6 5 5
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	-	-	p	p	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEE	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)		p	-	-	p	p	p	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		p	-	-	0	p	p	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff 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	-	-	p	p	-	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd dd ii dd dd	5 5 4 5
MUL	Unsigned multiply	$X:A \leftarrow (X) \times (A)$	-	0	-	-	-	0	INH	42		5

Table 13-2. QE32 Instruction Set Summary (Sheet 6 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
NEG <i>opr8a</i> NEGA NEGX NEG <i>opr8,X</i> NEG ,X NEG <i>opr8,SP</i>	Negate (Two's Complement)	$M \leftarrow \neg(M) = 0x00 - (M)$ $A \leftarrow \neg(A) = 0x00 - (A)$ $X \leftarrow \neg(X) = 0x00 - (X)$ $M \leftarrow \neg(M) = 0x00 - (M)$ $M \leftarrow \neg(M) = 0x00 - (M)$ $M \leftarrow \neg(M) = 0x00 - (M)$	p	-	-	p	p	p	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 # <i>opr8i</i> ORA <i>opr8a</i> ORA <i>opr16a</i> ORA <i>opr8,X</i> ORA <i>opr8,X</i> ORA ,X ORA <i>opr8,SP</i> ORA <i>opr16,SP</i> ORA <i>opr8,SP</i>	Inclusive OR Accumulator and Memory	$A \leftarrow (A) (M)$	0	-	-	p	p	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ee 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 <i>opr8a</i> ROLA ROLX ROL <i>opr8,X</i> ROL ,X ROL <i>opr8,SP</i>	Rotate Left through Carry		p	-	-	p	p	p	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	5 1 1 5 4 6
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR ,X 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	$SP \leftarrow 0xFF$ (High Byte Not Affected)	-	-	-	-	-	-	INH	9C		1
RTC	Return from CALL	$SP \leftarrow (SP) + 0x0001$; Pull (PPAGE) $SP \leftarrow (SP) + 0x0001$; Pull (PCH) $SP \leftarrow (SP) + 0x0001$; Pull (PCL)	-	-	-	-	-	-	INH	8D		7
RTI	Return from Interrupt	$SP \leftarrow (SP) + 0x0001$; Pull (CCR) $SP \leftarrow (SP) + 0x0001$; Pull (A) $SP \leftarrow (SP) + 0x0001$; Pull (X) $SP \leftarrow (SP) + 0x0001$; Pull (PCH) $SP \leftarrow (SP) + 0x0001$; Pull (PCL)	p	p	p	p	p	p	INH	80		9
RTS	Return from Subroutine	$SP \leftarrow SP + 0x0001$; Pull (PCH) $SP \leftarrow SP + 0x0001$; Pull (PCL)	-	-	-	-	-	-	INH	81		6
SBC # <i>opr8i</i> SBC <i>opr8a</i> SBC <i>opr16a</i> SBC <i>opr8,X</i> SBC <i>opr8,X</i> SBC ,X SBC <i>opr8,SP</i> SBC <i>opr16,SP</i> SBC <i>opr8,SP</i>	Subtract with Carry	$A \leftarrow (A) - (M) - (C)$	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 ee ff ff	2 3 4 4 3 3 5 4

Table 13-2. QE32 Instruction Set Summary (Sheet 7 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
SEC	Set Carry Bit	$C \leftarrow 1$	-	-	-	-	-	1	INH	99		1
SEI	Set Interrupt Mask Bit	$I \leftarrow 1$	-	-	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	$M \leftarrow (A)$	0	-	-	b	b	-	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff ff ee 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.)	$(M:M + 0x0001) \leftarrow (H:X)$	0	-	-	b	b	-	DIR EXT SP1	35 96 9EFF	dd hh ll ff	4 5 5
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	$I \text{ bit} \leftarrow 0$; 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	$M \leftarrow (X)$	0	-	-	b	b	-	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ff ee 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	$A \leftarrow (A) - (M)$	b	-	-	b	b	b	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
SWI	Software Interrupt	$PC \leftarrow (PC) + 0x0001$ Push (PCL); $SP \leftarrow (SP) - 0x0001$ Push (PCH); $SP \leftarrow (SP) - 0x0001$ Push (X); $SP \leftarrow (SP) - 0x0001$ Push (A); $SP \leftarrow (SP) - 0x0001$ Push (CCR); $SP \leftarrow (SP) - 0x0001$ $I \leftarrow 1$; PCH \leftarrow Interrupt Vector High Byte PCL \leftarrow Interrupt Vector Low Byte	-	-	1	-	-	-	INH	83		11
TAP	Transfer Accumulator to CCR	$CCR \leftarrow (A)$	b	b	b	b	b	b	INH	84		1
TAX	Transfer Accumulator to X (Index Register Low)	$X \leftarrow (A)$	-	-	-	-	-	-	INH	97		1
TPA	Transfer CCR to Accumulator	$A \leftarrow (CCR)$	-	-	-	-	-	-	INH	85		1
TST <i>opr8a</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	-	-	b	b	-	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	$H:X \leftarrow (SP) + 0x0001$	-	-	-	-	-	-	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	$A \leftarrow (X)$	-	-	-	-	-	-	INH	9F		1
TXS	Transfer Index Reg. to SP	$SP \leftarrow (H:X) - 0x0001$	-	-	-	-	-	-	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	$I \text{ bit} \leftarrow 0$; Halt CPU	-	-	0	-	-	-	INH	8F		2+

¹ Bus clock frequency is one-half of the CPU clock frequency.

Table 13-3. Opcode Map (Sheet 1 of 2)

Bit-Manipulation		Branch		Read-Modify-Write				Control				Register/Memory			
00 BRSET0 3 DIR	10 BSET0 2 DIR	20 BRA 2 REL	30 NEG 2 DIR	40 NEGA 1 INH	50 NEGX 1 INH	60 NEG 2 IX1	70 NEG 1 IX	80 RTI 1 INH	90 BGE 2 REL	A0 SUB 2 IMM	B0 SUB 2 DIR	C0 SUB 3 EXT	D0 SUB 3 IX2	E0 SUB 2 IX1	F0 SUB 1 IX
01 BRCLR0 3 DIR	11 BCLR0 2 DIR	21 BRN 2 REL	31 CBEQ 3 DIR	41 CBEQA 3 IMM	51 CBEQX 3 IMM	61 CBEQ 3 IX1+	71 CBEQ 2 IX+	81 RTS 1 INH	91 BLT 2 REL	A1 CMP 2 IMM	B1 CMP 2 DIR	C1 CMP 3 EXT	D1 CMP 3 IX2	E1 CMP 2 IX1	F1 CMP 1 IX
02 BRSET1 3 DIR	12 BSET1 2 DIR	22 BHI 2 REL	32 LDHX 3 EXT	42 MUL 1 INH	52 DIV 6	62 NSA 1 INH	72 DAA 1 INH	82 BGND 5+ 1 INH	92 BGT 2 REL	A2 SBC 2 IMM	B2 SBC 2 DIR	C2 SBC 3 EXT	D2 SBC 3 IX2	E2 SBC 2 IX1	F2 SBC 1 IX
03 BRCLR1 3 DIR	13 BCLR1 2 DIR	23 BLS 2 REL	33 COM 2 DIR	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	B3 CPX 2 DIR	C3 CPX 3 EXT	D3 CPX 3 IX2	E3 CPX 2 IX1	F3 CPX 1 IX
04 BRSET2 3 DIR	14 BSET2 2 DIR	24 BCC 2 REL	34 LSR 2 DIR	44 LSRA 1 INH	54 LSRX 1 INH	64 LSR 2 IX1	74 LSR 1 IX	84 TAP 1 INH	94 TXS 1 INH	A4 AND 2 IMM	B4 AND 2 DIR	C4 AND 3 EXT	D4 AND 3 IX2	E4 AND 2 IX1	F4 AND 1 IX
05 BRCLR2 3 DIR	15 BCLR2 2 DIR	25 BCS 2 REL	35 STHX 2 DIR	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	B5 BIT 2 DIR	C5 BIT 3 EXT	D5 BIT 3 IX2	E5 BIT 2 IX1	F5 BIT 1 IX
06 BRSET3 3 DIR	16 BSET3 2 DIR	26 BNE 2 REL	36 ROR 2 DIR	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	B6 LDA 2 DIR	C6 LDA 3 EXT	D6 LDA 3 IX2	E6 LDA 2 IX1	F6 LDA 1 IX
07 BRCLR3 3 DIR	17 BCLR3 2 DIR	27 BEQ 2 REL	37 ASR 2 DIR	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	B7 STA 2 DIR	C7 STA 3 EXT	D7 STA 3 IX2	E7 STA 2 IX1	F7 STA 1 IX
08 BRSET4 3 DIR	18 BSET4 2 DIR	28 BHCC 2 REL	38 LSL 2 DIR	48 LSLA 1 INH	58 LSLX 1 INH	68 LSL 2 IX1	78 LSL 1 IX	88 PULX 1 INH	98 CLC 1 INH	A8 EOR 2 IMM	B8 EOR 2 DIR	C8 EOR 3 EXT	D8 EOR 3 IX2	E8 EOR 2 IX1	F8 EOR 1 IX
09 BRCLR4 3 DIR	19 BCLR4 2 DIR	29 BHCS 2 REL	39 ROL 2 DIR	49 ROLA 1 INH	59 ROLX 1 INH	69 ROL 2 IX1	79 ROL 1 IX	89 PSHX 1 INH	99 SEC 1 INH	A9 ADC 2 IMM	B9 ADC 2 DIR	C9 ADC 3 EXT	D9 ADC 3 IX2	E9 ADC 2 IX1	F9 ADC 1 IX
0A BRSET5 3 DIR	1A BSET5 2 DIR	2A BPL 2 REL	3A DEC 2 DIR	4A DECA 1 INH	5A DECX 1 INH	6A DEC 2 IX1	7A DEC 1 IX	8A PULH 1 INH	9A CLI 1 INH	AA ORA 2 IMM	BA ORA 2 DIR	CA ORA 3 EXT	DA ORA 3 IX2	EA ORA 2 IX1	FA ORA 1 IX
0B BRCLR5 3 DIR	1B BCLR5 2 DIR	2B BBI 2 REL	3B DBNZ 3 DIR	4B DBNZA 2 INH	5B DBNZX 2 INH	6B DBNZ 3 IX1	7B DBNZ 2 IX	8B PSHH 1 INH	9B SEI 1 INH	AB ADD 2 IMM	BB ADD 2 DIR	CB ADD 3 EXT	DB ADD 3 IX2	EB ADD 2 IX1	FB ADD 1 IX
0C BRSET6 3 DIR	1C BSET6 2 DIR	2C BMC 2 REL	3C INC 2 DIR	4C INCA 1 INH	5C INCX 1 INH	6C INC 2 IX1	7C INC 1 IX	8C CLRH 1 INH	9C RSP 1 INH	AC CALL 4 EXT	BC JMP 2 DIR	CC JMP 3 EXT	DC JMP 3 IX2	EC JMP 2 IX1	FC JMP 1 IX
0D BRCLR6 3 DIR	1D BCLR6 2 DIR	2D BMS 2 REL	3D TST 2 DIR	4D TSTA 1 INH	5D TSTX 1 INH	6D TST 2 IX1	7D TST 1 IX	8D RTC 1 INH	9D NOP 1 INH	AD BSR 2 REL	BD JSR 2 DIR	CD JSR 3 EXT	DD JSR 3 IX2	ED JSR 2 IX1	FD JSR 1 IX
0E BRSET7 3 DIR	1E BSET7 2 DIR	2E BIL 2 REL	3E CPHX 3 EXT	4E MOV 3 DD	5E MOV 2 DIX+	6E MOV 3 IMD	7E MOV 2 IX+D	8E STOP 2+ 1 INH	9E Page 2	AE LDX 2 IMM	BE LDX 2 DIR	CE LDX 3 EXT	DE LDX 3 IX2	EE LDX 2 IX1	FE LDX 1 IX
0F BRCLR7 3 DIR	1F BCLR7 2 DIR	2F BIH 2 REL	3F CLR 2 DIR	4F CLRA 1 INH	5F CLR 1 INH	6F CLR 2 IX1	7F CLR 1 IX	8F WAIT 2+ 1 INH	9F TXA 1 INH	AF AIX 2 IMM	BF STX 2 DIR	CF STX 3 EXT	DF STX 3 IX2	EF STX 2 IX1	FF STX 1 IX

INH Inherent REL Relative SP1 Stack Pointer, 8-Bit Offset
 IMM Immediate IX Indexed, No Offset SP2 Stack Pointer, 16-Bit Offset
 DIR Direct IX1 Indexed, 8-Bit Offset IX+ Indexed, No Offset with
 EXT Extended IX2 Indexed, 16-Bit Offset Post Increment
 DD DIR to DIR IMD IMM to DIR IX1+ Indexed, 1-Byte Offset with
 IX+D IX+ to DIR DIX+ DIR to IX+ Post Increment

Opcode in F0 3
 Hexadecimal SUB IX
 Number of Bytes 1

QE32 Cycles
 Instruction Mnemonic
 Addressing Mode

Table 13-3. Opcode Map (Sheet 2 of 2)

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

INH Inherent **REL** Relative **SP1** Stack Pointer, 8-Bit Offset
IMM Immediate **IX** Indexed, No Offset **SP2** Stack Pointer, 16-Bit Offset
DIR Direct **IX1** Indexed, 8-Bit Offset **IX+** Indexed, No Offset with
EXT Extended **IX2** Indexed, 16-Bit Offset **IX+** Post Increment
DD DIR to DIR **IMD** IMM to DIR **IX1+** Indexed, 1-Byte Offset with
IX+D IX+ to DIR **DIX+** DIR to IX+ **IX+** Post Increment

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

Prebyte (9E) and Opcode in Hexadecimal	9E60 6 NEG 3 SP1	QE32 Cycles Instruction Mnemonic Addressing Mode
Number of Bytes		

Chapter 14

MCU Keyboard Interrupt (KBI)

14.1 Introduction

The keyboard interrupt module provides up to eight independently enabled external interrupt sources. 9S08QE32 series devices contain two KBI modules, called KBI1 and KBI2. Each KBI module has up to eight interrupt sources.

14.1.1 KBI Clock Gating

The bus clock to the KBI can be gated on and off using the KBI bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, this bit can be cleared to disable the clock to this module when not in use.

14.1.2 Features

The KBI features include:

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

14.1.3 Modes of Operation

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

14.1.3.1 KBI in Wait Mode

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

14.1.3.2 KBI in Stop Modes

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

During stop2 mode, the KBI is disabled. Upon wakeup from stop2 mode, the KBI module is in the reset state.

14.1.3.3 KBI in Active Background Mode

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

14.1.4 Block Diagram

Figure 14-1 shows the block diagram for the keyboard interrupt module.

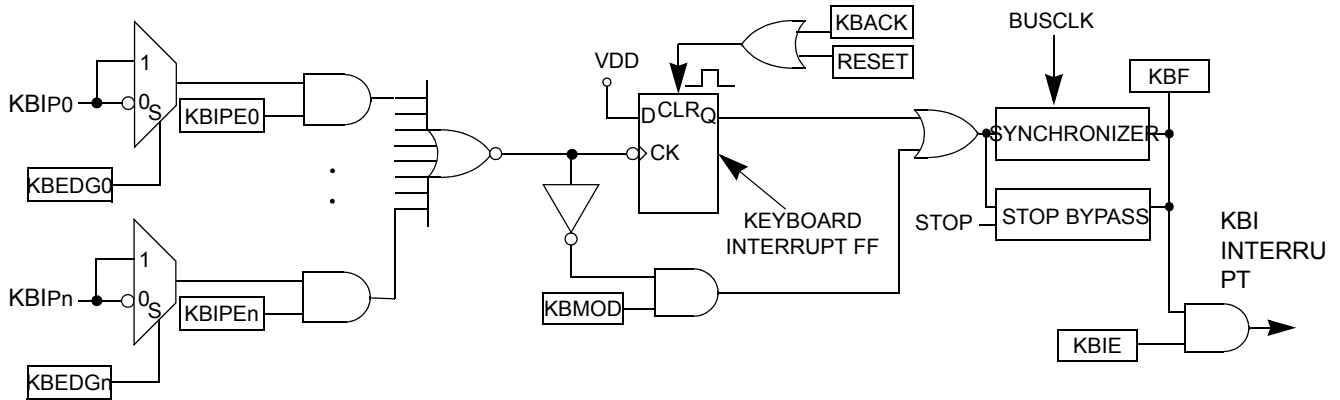


Figure 14-1. Keyboard Interrupt (KBI) Block Diagram

14.2 External Signal Description

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

Table 14-1. KBI1 Pin Mapping

Port pin	PTB3	PTB2	PTB1	PTB0	PTA3	PTA2	PTA1	PTA0
KBI1 pin	KBI1P7	KBI1P6	KBI1P5	KBI1P4	KBI1P3	KBI1P2	KBI1P1	KBI1P0

Table 14-2. KBI2 Pin Mapping

Port pin	PTD7	PTD6	PTD5	PTD4	PTD3	PTAD2	PTD1	PTD0
KBI2 pin	KBI2P7	KBI2P6	KBI2P5	KBI2P4	KBI2P3	KBI2P2	KBI2P1	KBI2P0

14.3 Register Definition

14.3.1 Register Descriptions

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

This section refers to registers and control bits only by their names and relative address offsets.

14.3.2 KBI Interrupt Status and Control Register (KBISC)

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

Figure 14-2. KBI Interrupt Status and Control Register (KBISC)

Table 14-3. KBISC Register Field Descriptions

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

14.3.3 KBI Interrupt Pin Select Register (KBIPE)

Offset



Figure 14-3. KBI Interrupt Pin Select Register (KBIPE)

Table 14-4. KBIPE Register Field Descriptions

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

14.3.4 KBI Interrupt Edge Select Register (KBIES)

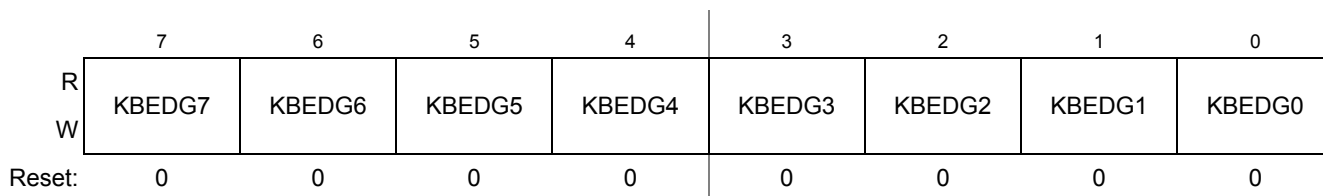


Figure 14-4. KBI Edge Select Register (KBIES)

Table 14-5. KBIES Register Field Descriptions

Field	Description
7:0 KBEDG[7:0]	KBI Edge Selects — Each of the KBEDG _n bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pullup or pulldown device if enabled. 0 A pullup device is connected to the associated pin and detects falling edge/low level for interrupt generation. 1 A pulldown device is connected to the associated pin and detects rising edge/high level for interrupt generation.

14.4 Functional Description

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

Writing to the KBIPE_n bits in the keyboard interrupt pin enable register (KBIPE) independently enables or disables each port pin. Each port can be configured as edge sensitive or edge and level sensitive based on the KBIMOD bit in the keyboard interrupt status and control register (KBISC). Edge sensitivity can

be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the KBEDGn bits in the keyboard interrupt edge select register (**KBIES**).

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

14.4.1 Edge Only Sensitivity

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

14.4.2 Edge and Level Sensitivity

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

14.4.3 Pullup/Pulldown Resistors

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

14.4.4 Keyboard Interrupt Initialization

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

1. Mask interrupts by clearing KBIE in **KBISC**.
2. Select the pin polarity by setting the appropriate KBEDGn bits in **KBIES**.
3. If using internal pullup/pulldown device, configure the associated pull enable bits in **KBIPEn**.
4. Enable the interrupt pins by setting the appropriate KBIPEn bits in **KBIPEn**.
5. Write to KBACK in **KBISC** to clear any false interrupts.
6. Set KBIE in **KBISC** to enable interrupts.

Chapter 15

Timer/Pulse-Width Modulator

15.0.1 Features

The TPM includes these distinctive features:

- Channels:
 - Each channel may be input capture, output compare, or edge-aligned PWM
 - Rising-Edge, falling-edge, or any-edge input capture trigger
 - Set, clear, or toggle output compare action
 - Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
 - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
 - Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
 - External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

15.0.2 ACMP/TPM Configuration Information

The ACMP modules can be configured to connect the output of the analog comparator to a TPM input capture channel 0 by setting the corresponding ACICx bit in SOPT2. With ACICx set, the TPMxCH0 pin is not available externally regardless of the configuration of the TPMx module.

The ACMP1 output can be connected to TPM1CH0; The ACMP2 output can be connected to TPM2CH0.

15.0.3 TPM Clock Gating

The bus clock to the TPMs can be gated on and off using the TPM3 bit, TPM2 bit, and TPM1 bit in SCGC1. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use.

15.0.4 Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- **Input capture mode**
When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- **Output compare mode**
When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).
- **Edge-aligned PWM mode**
The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- **Center-aligned PWM mode**
Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

15.0.5 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPM1CH_n (timer channel *n*) where *n* is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 15-1 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and

edge-aligned PWM functions. The timer counter modulo registers, TPM1MODH:TPM1MODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPM1CNT counter resets the counter, regardless of the data value written.

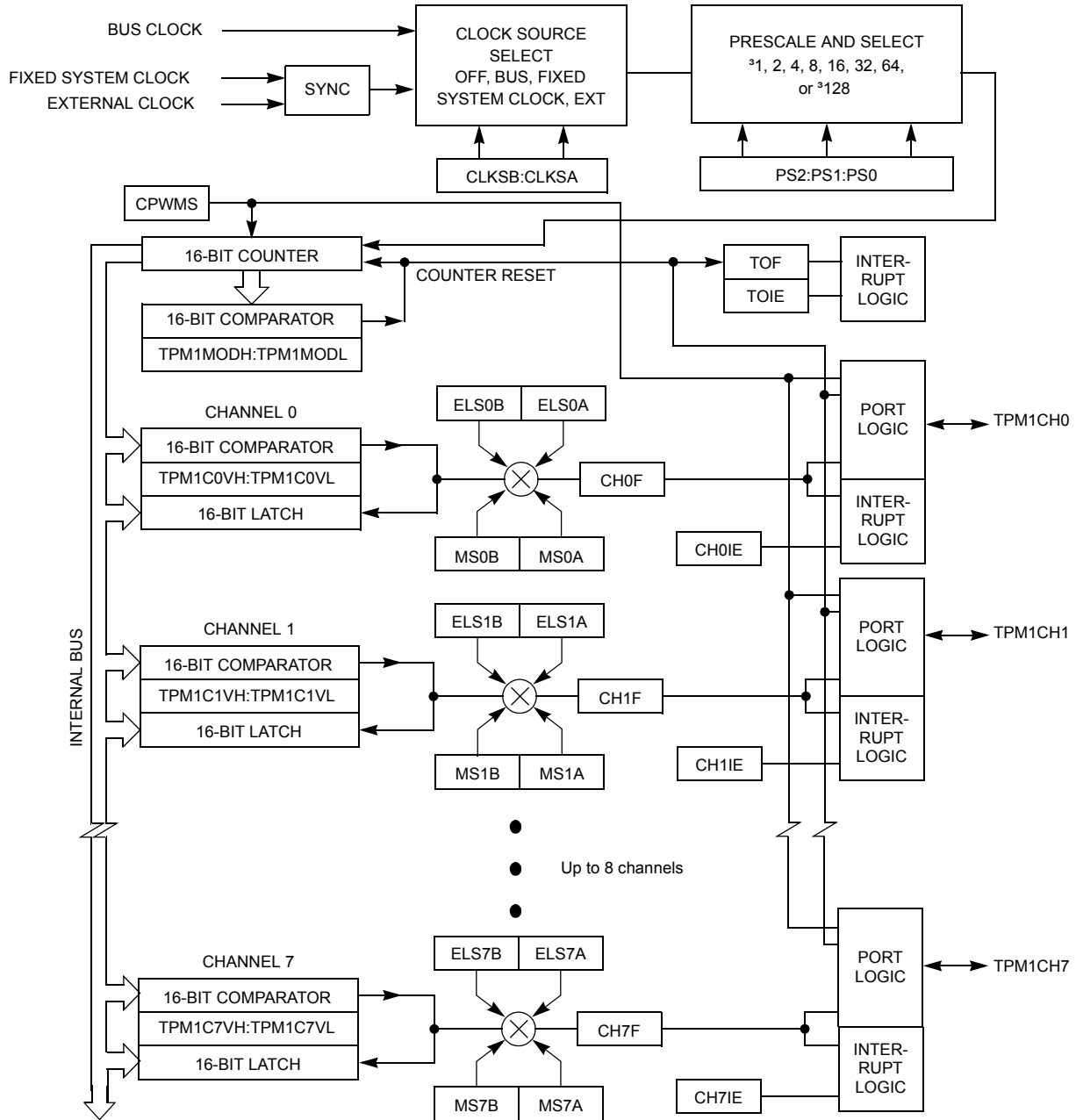


Figure 15-1. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

15.1 Signal Description

Table 15-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 15-1. Signal Properties

Name	Function
EXTCLK ¹	External clock source which may be selected to drive the TPM counter.
TPM1CHn ²	I/O pin associated with TPM channel n

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

² n=channel number (1 to 8)

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

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

15.1.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 15-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

15.1.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

15.1.1.2 TPM1CHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPM1CHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPM1CHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPM1CHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPM1CHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPM1CHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPM1CHn pin is forced high at the start of each new period (TPM1CNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPM1CHn pin is forced low at the start of each new period (TPM1CNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

TPM1MODH:TPM1MODL = 0x0008
 TPM1CnVH:TPM1CnVL = 0x0005

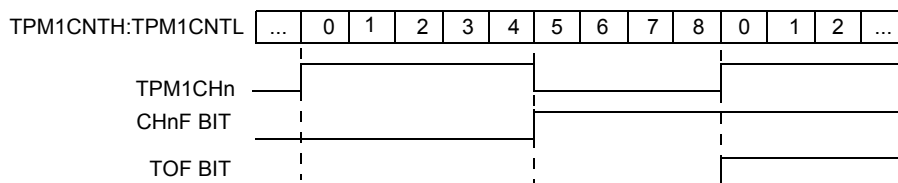


Figure 15-2. High-True Pulse of an Edge-Aligned PWM

TPM1MODH:TPM1MODL = 0x0008
 TPM1CnVH:TPM1CnVL = 0x0005

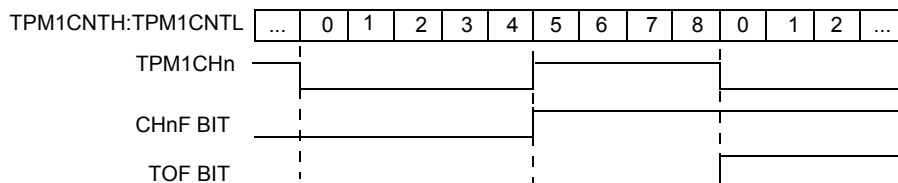


Figure 15-3. Low-True Pulse of an Edge-Aligned PWM

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPM1CHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPM1CHn output. If ELSnB:ELSnA=1:0, the corresponding TPM1CHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPM1CHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPM1CHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPM1CHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

TPM1MODH:TPM1MODL = 0x0008
 TPM1CnVH:TPM1CnVL = 0x0005

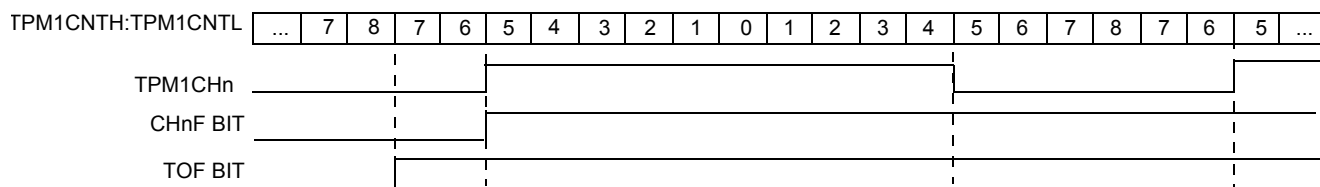


Figure 15-4. High-True Pulse of a Center-Aligned PWM

TPM1MODH:TPM1MODL = 0x0008
 TPM1CnVH:TPM1CnVL = 0x0005

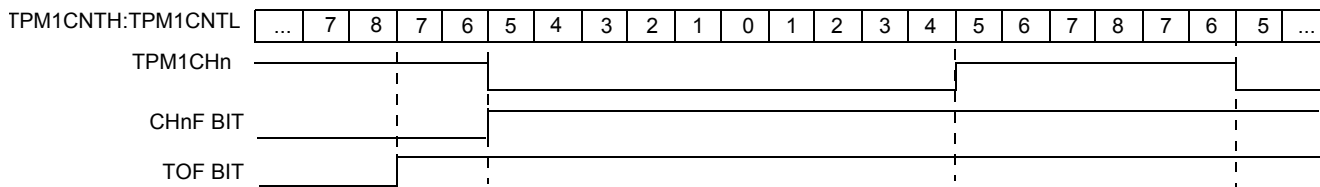


Figure 15-5. Low-True Pulse of a Center-Aligned PWM

15.2 Memory Map and Register Definition

This section provides a detailed description of all TPM registers accessible to the end user.

15.2.1 Module Memory Map

This section presents a high-level view of the TPM registers and how they are mapped. [Table 15-2](#) shows the registers for a full-featured, 8-channel variation of the TPM. Register names include a timer number (x) because it is common for MCU systems to include more than one TPM.

This section consists of register descriptions in address order.

15.2.2 TPM Status and Control Register (TPM1SC)

TPM1SC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

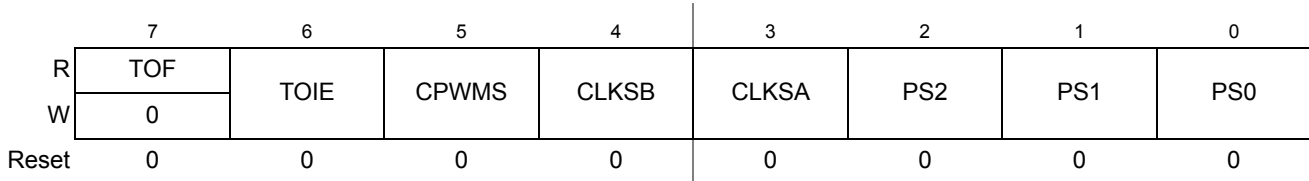


Figure 15-6. TPM Status and Control Register (TPM1SC)

Table 15-2. TPM1SC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.

Table 15-2. TPM1SC Field Descriptions (continued)

Field	Description
4–3 CLKS[B:A]	Clock source selects. As shown in Table 15-3 , this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based or FLL-based system clock. When there is no PLL or FLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL or FLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL or FLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 15-4 . This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.

Table 15-3. TPM-Clock-Source Selection

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

Table 15-4. Prescale Factor Selection

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

15.2.3 TPM-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 half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPM1SC).

Reset clears the TPM counter registers. Writing any value to TPM1CNTH or TPM1CNTL also clears the TPM counter (TPM1CNTH:TPM1CNTL) and resets the coherency mechanism, regardless of the data involved in the write.

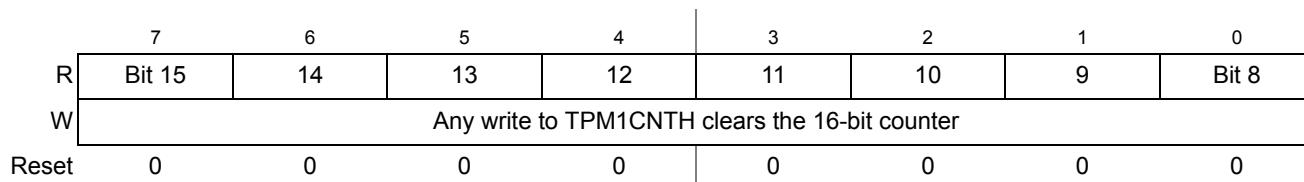


Figure 15-7. TPM Counter Register High (TPM1CNTH)

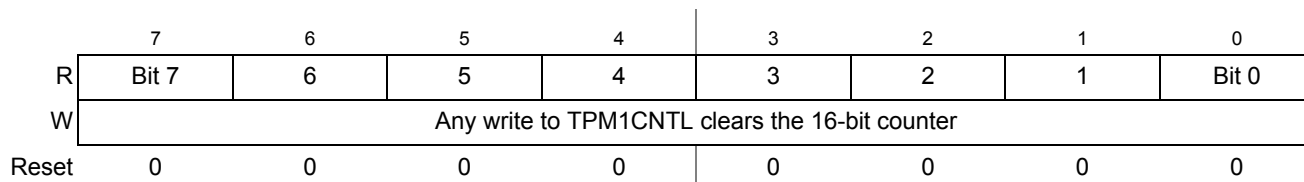


Figure 15-8. TPM Counter Register Low (TPM1CNTL)

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPM1SC, TPM1CNTH or TPM1CNTL registers resets the read coherency mechanism of the TPM1CNTH:L registers, regardless of the data involved in the write.

15.2.4 TPM 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 0x0000 at the next clock, 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 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPM1MODH or TPM1MODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), then the registers are updated after both bytes were written, and the TPM counter changes from (TPM1MODH:TPM1MODL - 1) to (TPM1MODH:TPM1MODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPM1SC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPM1SC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

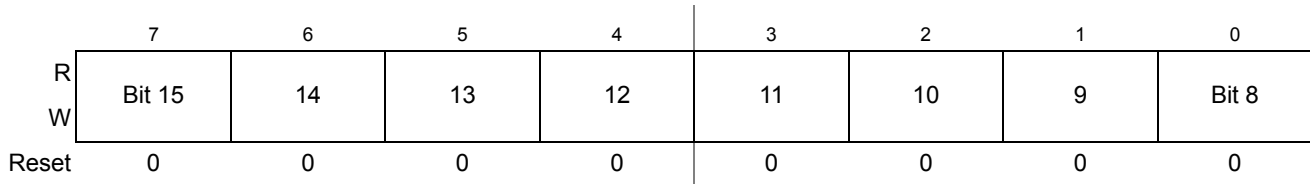


Figure 15-9. TPM Counter Modulo Register High (TPM1MODH)

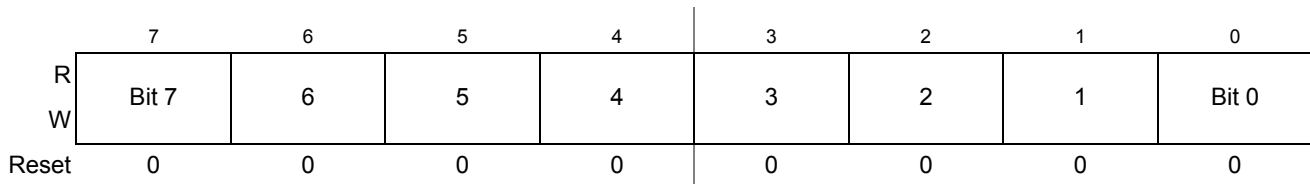


Figure 15-10. TPM Counter Modulo Register Low (TPM1MODL)

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

15.2.5 TPM Channel n Status and Control Register (TPM1CnSC)

TPM1CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

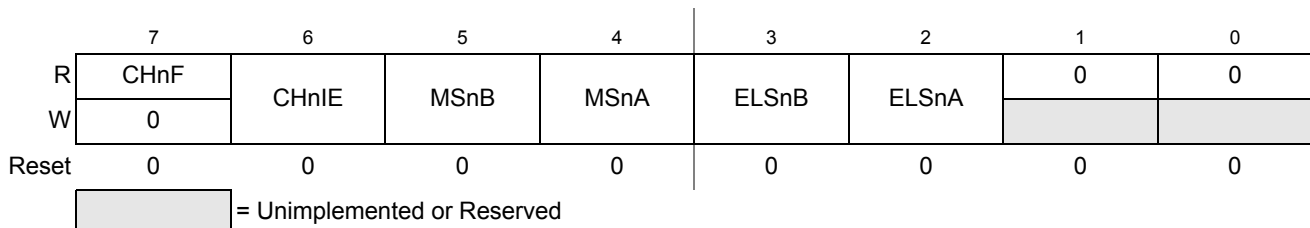


Figure 15-11. TPM Channel n Status and Control Register (TPM1CnSC)

Table 15-5. TPM1CnSC Field Descriptions

Field	Description
7 CHnF	<p>Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPM1CnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF.</p> <p>Reset clears the CHnF bit. Writing a logic 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 on channel n</p>
6 CHnIE	<p>Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE.</p> <p>0 Channel n interrupt requests disabled (use for software polling) 1 Channel n interrupt requests enabled</p>
5 MSnB	<p>Mode select B for TPM channel n. When CPWMS=0, MSnB=1 configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 15-6.</p>
4 MSnA	<p>Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 15-6 for a summary of channel mode and setup controls.</p> <p>Note: If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.</p>
3–2 ELSnB ELSnA	<p>Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 15-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output.</p> <p>Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.</p>

Table 15-6. Mode, Edge, and Level Selection

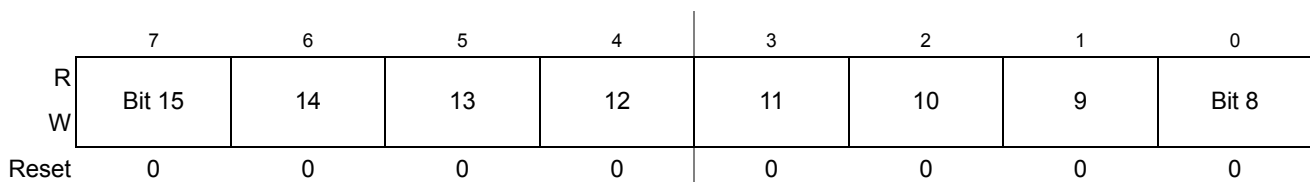
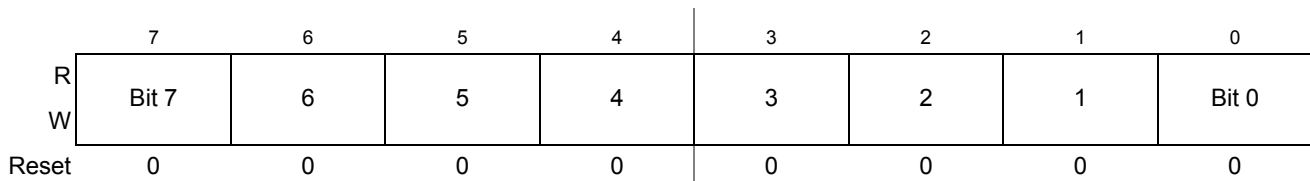
CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00		Pin not used for TPM - revert to general purpose I/O or other peripheral control

Table 15-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
	1	XX	10	Center-aligned PWM
X1			Low-true pulses (set output on compare-up)	

15.2.6 TPM 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 registers are cleared by reset.


Figure 15-12. TPM Channel Value Register High (TPM1CnVH)

Figure 15-13. TPM 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 half is read. This latching mechanism also resets

(becomes unlatched) when the TPM1CnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPM1CnSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPM1CnVH and TPM1CnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPM1CnVH or TPM1CnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKS_B:CLKS_A bits and the selected mode, so:

- If (CLKS_B:CLKS_A = 0:0), then the registers are updated when the second byte is written.
- If (CLKS_B:CLKS_A not = 0:0 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKS_B:CLKS_A not = 0:0 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPM1MODH:TPM1MODL - 1) to (TPM1MODH:TPM1MODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPM1CnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

15.3 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

15.3.1 Counter

All timer functions are based on the main 16-bit counter (TPM1CNTH:TPM1CNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

15.3.1.1 Counter Clock Source

The 2-bit field, CLKS_B:CLKS_A, in the timer status and control register (TPM1SC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See [Table 15-3](#). After any MCU reset, CLKS_B:CLKS_A=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKS_B:CLKS_A field) does not affect the values in the counter or other timer registers.

Table 15-7. TPM Clock Source Selection

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

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

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

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

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

15.3.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

15.3.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPM1MODH:TPM1MODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

15.3.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPM1CNTH or TPM1CNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

15.3.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

15.3.2.1 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.

In input capture mode, the TPM1CnVH and TPM1CnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPM1CnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

15.3.2.2 Output Compare Mode

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

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

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

The coherency sequence can be manually reset by writing to the channel status/control register (TPM1CnSC).

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

15.3.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPM1MODH:TPM1MODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPM1CnVH:TPM1CnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 15-14). 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.

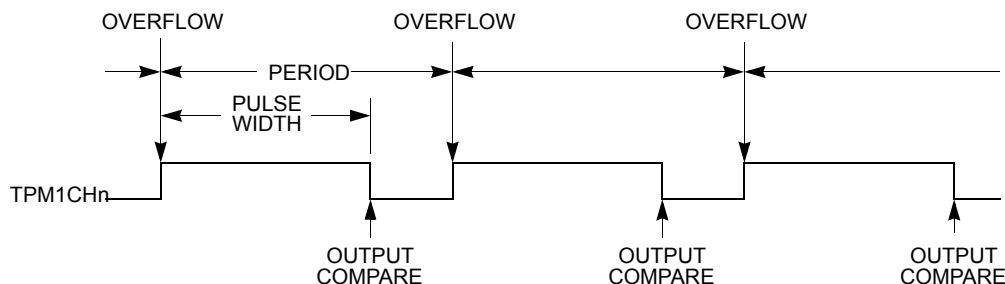


Figure 15-14. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPM1CnVH:TPM1CnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPM1CnVH and TPM1CnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPM1MODH:TPM1MODL - 1) to (TPM1MODH:TPM1MODL). If

the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

15.3.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPM1CnVH:TPM1CnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPM1MODH:TPM1MODL. TPM1MODH:TPM1MODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPM1CnVH:TPM1CnVL})$$

$$\text{period} = 2 \times (\text{TPM1MODH:TPM1MODL}); \text{TPM1MODH:TPM1MODL}=0\text{x}0001\text{-}0\text{x}7\text{FFF}$$

If the channel-value register TPM1CnVH:TPM1CnVL is zero or negative (bit 15 set), the duty cycle will be 0%. 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% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPM1MODH:TPM1MODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 15-15). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPM1MODH:TPM1MODL, then counts down until it reaches zero. This sets the period equal to two times TPM1MODH:TPM1MODL.

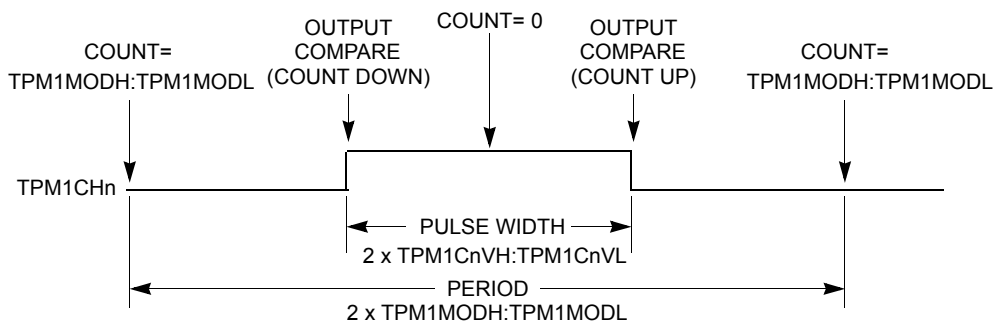


Figure 15-15. CPWM Period and Pulse Width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPM1MODH, TPM1MODL, TPM1CnVH, and TPM1CnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPM1CnVH:L registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPM1MODH:TPM1MODL - 1) to (TPM1MODH:TPM1MODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

When TPM1CNTH:TPM1CNTL=TPM1MODH:TPM1MODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

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.

15.4 Reset Overview

15.4.1 General

The TPM is reset whenever any MCU reset occurs.

15.4.2 Description of Reset Operation

Reset clears the TPM1SC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

15.5 Interrupts

15.5.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in [Table 15-8](#) which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

Table 15-8. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user's guide for the interrupt module or to the chip's complete documentation for details.

15.5.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user's software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

15.5.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

15.5.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

15.5.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

15.5.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

15.5.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in [Section 15.5.2, "Description of Interrupt Operation"](#).

15.5.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described [Section 15.5.2, "Description of Interrupt Operation"](#).

15.5.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described [Section 15.5.2, "Description of Interrupt Operation"](#).

Chapter 16

MCU Serial Communications Interface (SCI)

16.1 Introduction

This chapter describes the MCU Serial Communications Interface (SCI).

NOTE

Stop1 mode is not available in this device.

16.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
 - Transmit data register empty and transmission complete
 - Receive data register full
 - Receive overrun, parity error, framing error, and noise error
 - Idle receiver detect
 - Active edge on receive pin
 - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

16.1.2 Modes of Operation

See [Section 16.3, “Functional Description”](#), for details concerning SCI operation in the following modes:

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

16.1.3 Block Diagram

Figure 16-1 shows the transmitter portion of the SCI.

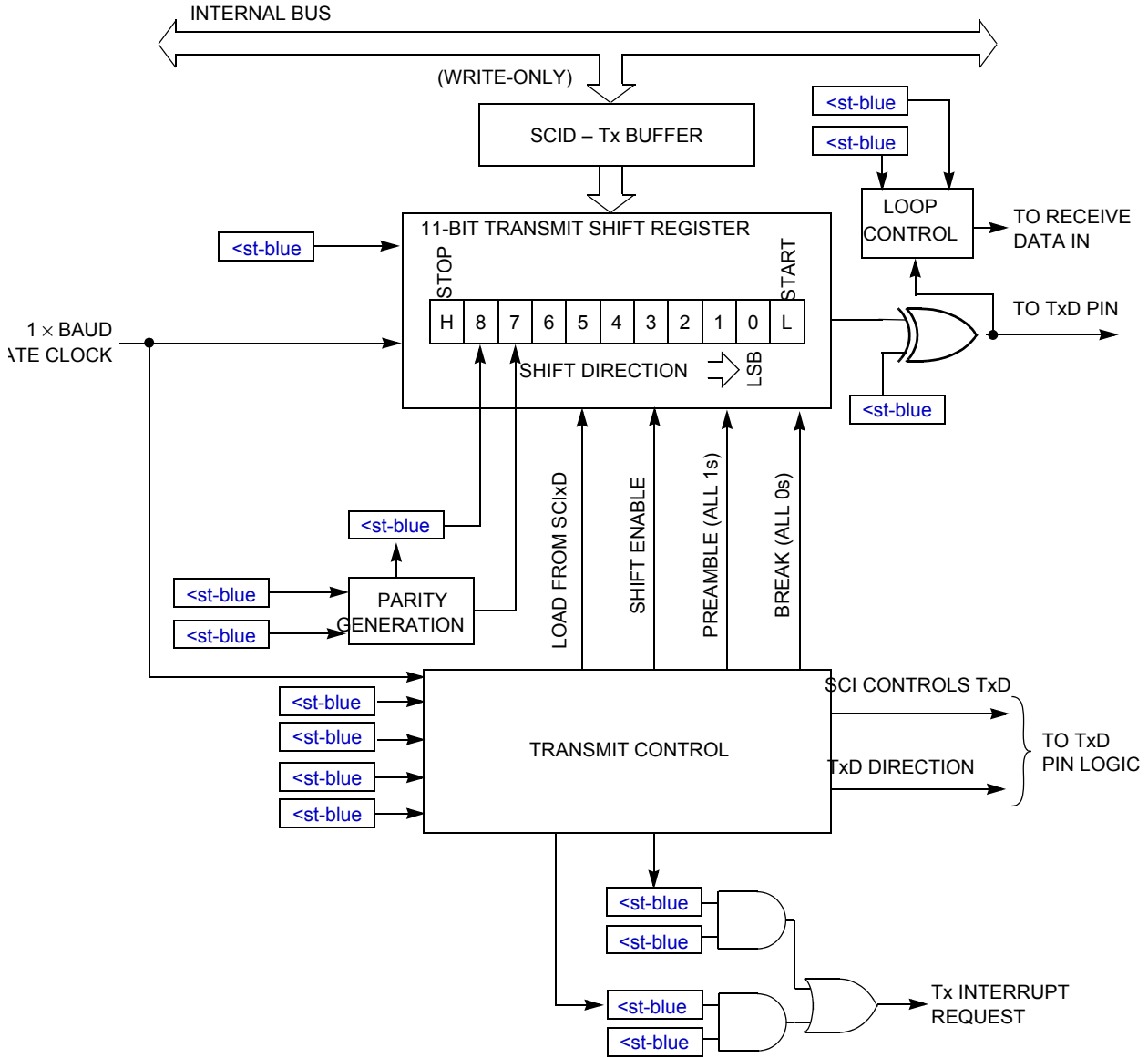


Figure 16-1. SCI Transmitter Block Diagram

Figure 16-2 shows the receiver portion of the SCI.

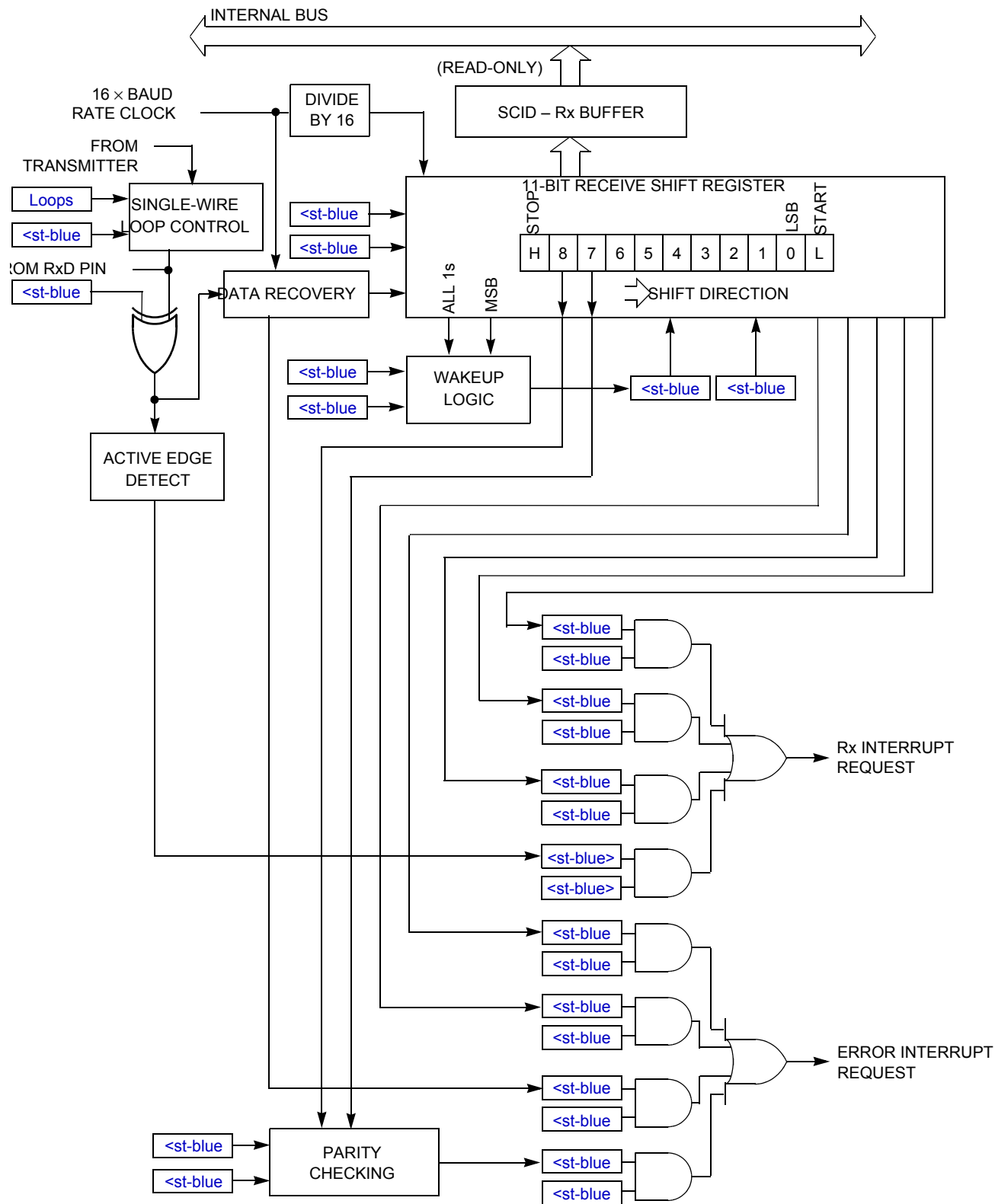


Figure 16-2. SCI Receiver Block Diagram

16.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the [Memory](#) chapter of the MC12311 *Reference Manual* 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.

16.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).

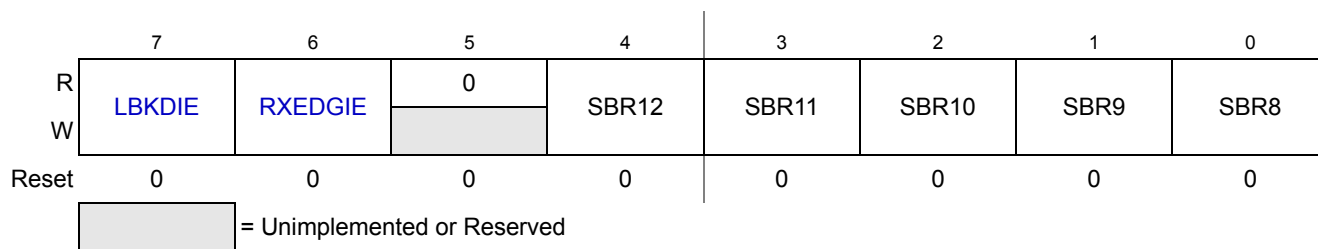
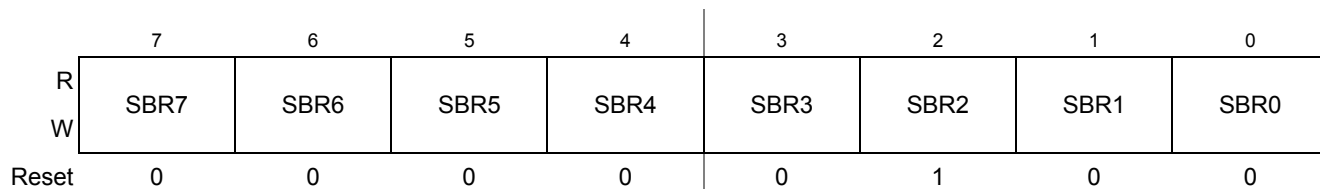


Figure 16-3. SCI Baud Rate Register (SCIxBDH)

Table 16-1. SCIxBDH Field Descriptions

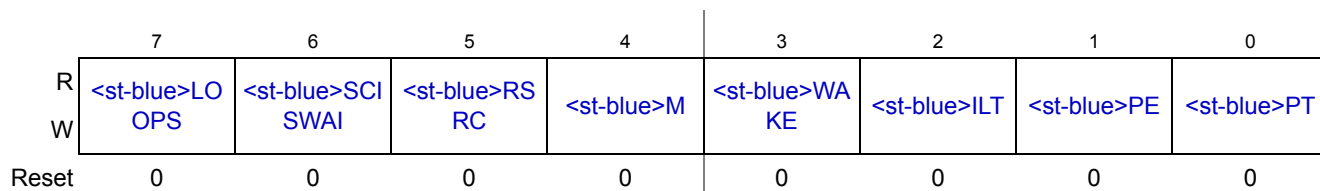
Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable (for LDKDIF) 0 Hardware interrupts from LDKDIF disabled (use polling). 1 Hardware interrupt requested when LDKDIF flag is 1.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable (for RXEDGIF) 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	Baud Rate Modulo Divisor — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 16-2 .


Figure 16-4. SCI Baud Rate Register (SClxBDL)
Table 16-2. SClxBDL Field Descriptions

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

16.2.2 SCI Control Register 1 (SClxC1)

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


Figure 16-5. SCI Control Register 1 (SClxC1)
Table 16-3. SClxC1 Field Descriptions

Field	Description
7 LOOPS	Loop Mode Select — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See <st-blue>RSRC bit that follows.) RxD pin is not used by SCI.
6 SCISWAI	SCI Stops in Wait Mode 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	Receiver Source Select — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.

Table 16-3. SC1xC1 Field Descriptions (continued)

Field	Description
3 WAKE	Receiver Wakeup Method Select — Refer to Section 16.3.3.2, “Receiver Wakeup Operation” for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	Idle Line Type Select — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 16.3.3.2.1, “Idle-Line Wakeup” for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	Parity Enable — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

16.2.3 SCI Control Register 2 (SC1xC2)

This register can be read or written at any time.



Figure 16-6. SCI Control Register 2 (SC1xC2)

Table 16-4. SC1xC2 Field Descriptions

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

Table 16-4. SC1xC2 Field Descriptions (continued)

Field	Description
3 TE	Transmitter Enable 0 Transmitter off. 1 Transmitter on. TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to Section 16.3.2.1, “Send Break and Queued Idle” for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.
2 RE	Receiver Enable — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 16.3.3.2, “Receiver Wakeup Operation” for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	Send Break — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 16.3.2.1, “Send Break and Queued Idle” for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

16.2.4 SCI Status Register 1 (SC1xS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) clear these status flags.

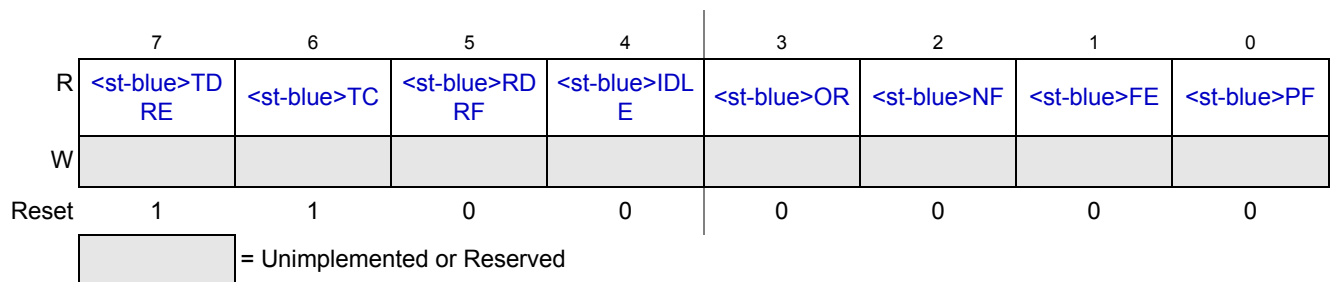

Figure 16-7. SCI Status Register 1 (SC1xS1)

Table 16-5. SC1xS1 Field Descriptions

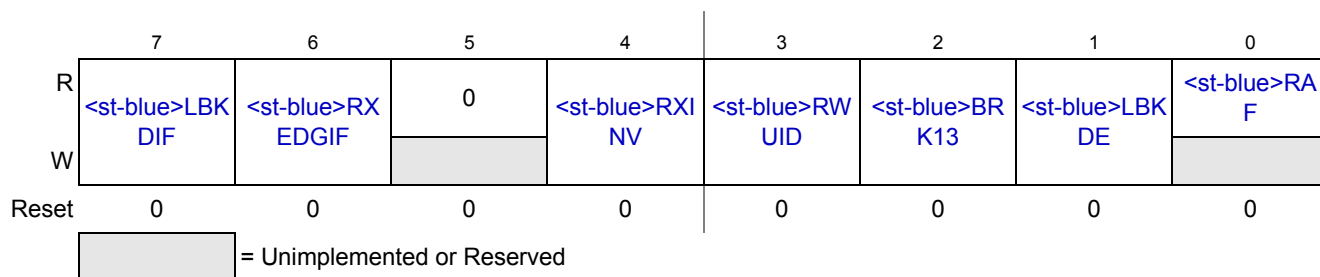
Field	Description
7 TDRE	<p>Transmit Data Register Empty Flag — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SC1xS1 with TDRE = 1 and then write to the SCI data register (SC1xD).</p> <p>0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.</p>
6 TC	<p>Transmission Complete Flag — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.</p> <p>0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).</p> <p>TC is cleared automatically by reading SC1xS1 with TC = 1 and then doing one of the following three things:</p> <ul style="list-style-type: none"> • Write to the SCI data register (SC1xD) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SC1xC2
5 RDRF	<p>Receive Data Register Full Flag — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SC1xD). To clear RDRF, read SC1xS1 with RDRF = 1 and then read the SCI data register (SC1xD).</p> <p>0 Receive data register empty. 1 Receive data register full.</p>
4 IDLE	<p>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.</p> <p>To clear IDLE, read SC1xS1 with IDLE = 1 and then read the SCI data register (SC1xD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period.</p> <p>0 No idle line detected. 1 Idle line was detected.</p>
3 OR	<p>Receiver Overrun Flag — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SC1xD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SC1xD. To clear OR, read SC1xS1 with OR = 1 and then read the SCI data register (SC1xD).</p> <p>0 No overrun. 1 Receive overrun (new SCI data lost).</p>
2 NF	<p>Noise Flag — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF is set at the same time as the RDRF flag is set for the character. To clear NF, read SC1xS1 and then read the SCI data register (SC1xD).</p> <p>0 No noise detected. 1 Noise detected in the received character in SC1xD.</p>

Table 16-5. SC1xS1 Field Descriptions (continued)

Field	Description
1 FE	Framing Error Flag — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SC1xS1 with FE = 1 and then read the SCI data register (SC1xD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	Parity Error Flag — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SC1xS1 and then read the SCI data register (SC1xD). 0 No parity error. 1 Parity error.

16.2.5 SCI Status Register 2 (SC1xS2)

This register contains one read-only status flag.


Figure 16-8. SCI Status Register 2 (SC1xS2)
Table 16-6. SC1xS2 Field Descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV ¹	Receive Data Inversion — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	Receive Wake Up Idle Detect — RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length — BRK13 selects a longer transmitted break character length. The state of this bit does not affect the detection of a framing error. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

Table 16-6. SCIxS2 Field Descriptions (continued)

Field	Description
1 LBKDE	LIN Break Detection Enable — LBKDE selects a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	Receiver Active Flag — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

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

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

16.2.6 SCI Control Register 3 (SCIxC3)

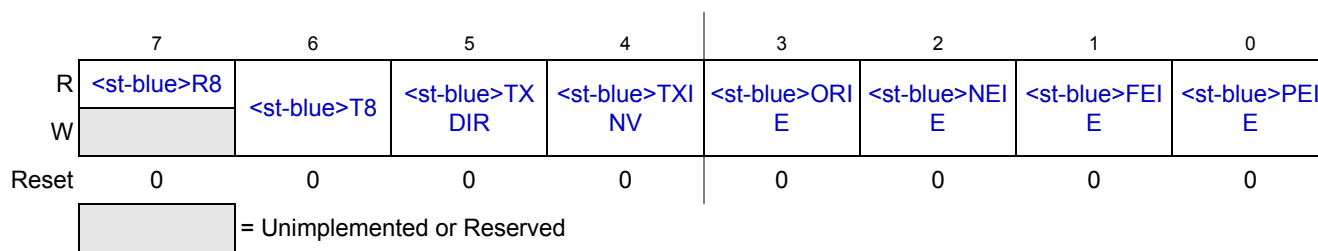


Figure 16-9. SCI Control Register 3 (SCIxC3)

Table 16-7. SCIxC3 Field Descriptions

Field	Description
7 R8	Ninth Data Bit for Receiver — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCIxR register. When reading 9-bit data, read R8 before reading SCIxR because reading SCIxR completes automatic flag clearing sequences which could allow R8 and SCIxR to be overwritten with new data.
6 T8	Ninth Data Bit for Transmitter — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCIxR register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxR is written so T8 should be written (if it needs to change from its previous value) before SCIxR 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 SCIxR is written.
5 TXDIR	TxD Pin Direction in Single-Wire Mode — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

Table 16-7. SCIxC3 Field Descriptions (continued)

Field	Description
4 TXINV ¹	Transmit Data Inversion — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	Overrun Interrupt Enable — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	Noise Error Interrupt Enable — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	Framing Error Interrupt Enable — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	Parity Error Interrupt Enable — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

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

16.2.7 SCI 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.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 16-10. SCI Data Register (SCIxD)

16.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

16.3.1 Baud Rate Generation

As shown in [Figure 16-11](#), the clock source for the SCI baud rate generator is the bus-rate clock.

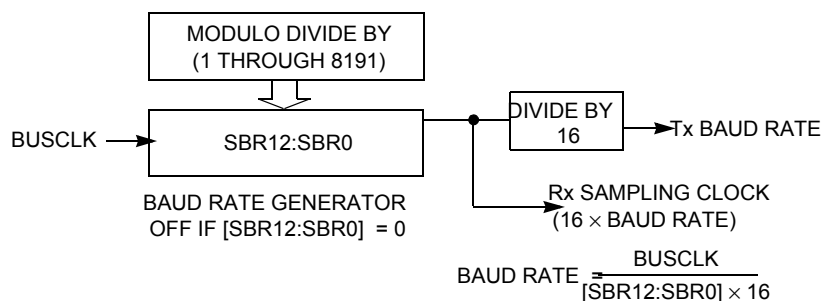


Figure 16-11. SCI Baud Rate Generation

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

The MCU resynchronizes to bit boundaries on every high-to-low transition. In the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For an SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about $\pm 4.5\%$ for 8-bit data format and about $\pm 4\%$ 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.

16.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in [Figure 16-1](#).

The transmitter output (TxD) idle state defaults to logic high ($TXINV = 0$ following reset). The transmitter output is inverted by setting $TXINV = 1$. The transmitter is enabled by setting the TE bit in $SCIx C2$. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register ($SCIxD$).

The central element of the SCI transmitter is the transmit shift register that is 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, assume $M = 0$, 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 can be written to the transmit data buffer at $SCIxD$.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

16.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIx2 sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK remains 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters are received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin shared with TxD is an output driving a logic 1. This ensures that the TxD line looks like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 16-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

16.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 16-2) is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

Set RXINV = 1 to invert the receiver input. Set the RE bit in SCIx2 to enable the receiver. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section •, “8- and 9-bit data modes.” For the remainder of this discussion, assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ($RDRF = 1$), it reads $SCIxD$ to get the data from the receive data register. The $RDRF$ flag is cleared automatically when the program that handles receive data issues a 2-step sequence. Refer to [Section 16.3.4, “Interrupts and Status Flags”](#) for more details about flag clearing.

16.3.3.1 Data Sampling Technique

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

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

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

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

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

16.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in $SCIxC2$. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, $IDLE$, when $RWUID$ bit is set) are inhibited from setting. This eliminates the need for software overhead to handle 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.

16.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which sets the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

16.3.3.2.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters, but requires the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case, the character with the MSB set is received even though the receiver was sleeping during most of this character time.

16.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events. A third vector is used for OR, NF, FE, and PF error conditions. Local interrupt enable masks can separately mask each of these ten interrupt sources. Software can still poll the flags when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates available room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt is requested when TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1. Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading $SCIxD$. The $RDRF$ flag is cleared by reading $SCIxS1$ while $RDRF = 1$ and then reading $SCIxD$.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, $SCIxS1$ must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading $SCIxS1$ while $IDLE = 1$ and then reading $SCIxD$. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set $RDRF$.

If the associated error was detected in the received character that caused $RDRF$ to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — are set at the same time as $RDRF$. These flags are not set in overrun cases.

If $RDRF$ was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the $RXEDGIF$ flag to set. The $RXEDGIF$ flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled ($RE = 1$).

16.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

16.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in $SCIxC1$. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in $T8$ in $SCIxC3$. For the receiver, the ninth bit is held in $R8$ in $SCIxC3$.

For coherent writes to the transmit data buffer, write to the $T8$ bit before writing to $SCIxD$.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to $T8$ again. When data is transferred from the transmit data buffer to the transmit shifter, the value in $T8$ is copied at the same time data is transferred from $SCIxD$ to the shifter.

Typically, use the 9-bit data mode in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or use it 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.

16.3.5.2 Stop Mode Operation

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

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

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

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

16.3.5.3 Loop Mode

When $LOOPS = 1$, the RSRC bit in the same register chooses between loop mode ($RSRC = 0$) or single-wire mode ($RSRC = 1$). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

16.3.5.4 Single-Wire Operation

When $LOOPS = 1$, the RSRC bit in the same register chooses between loop mode ($RSRC = 0$) or single-wire mode ($RSRC = 1$). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When $TXDIR = 0$, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When $TXDIR = 1$, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

16.3.6 SCI Clock Gating

The bus clock to the SCI1 and SCI2 can be gated on and off using the SCI1 and SCI2 bit respectively in SCGC1. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use.

MCU Serial Communications Interface (SCI)

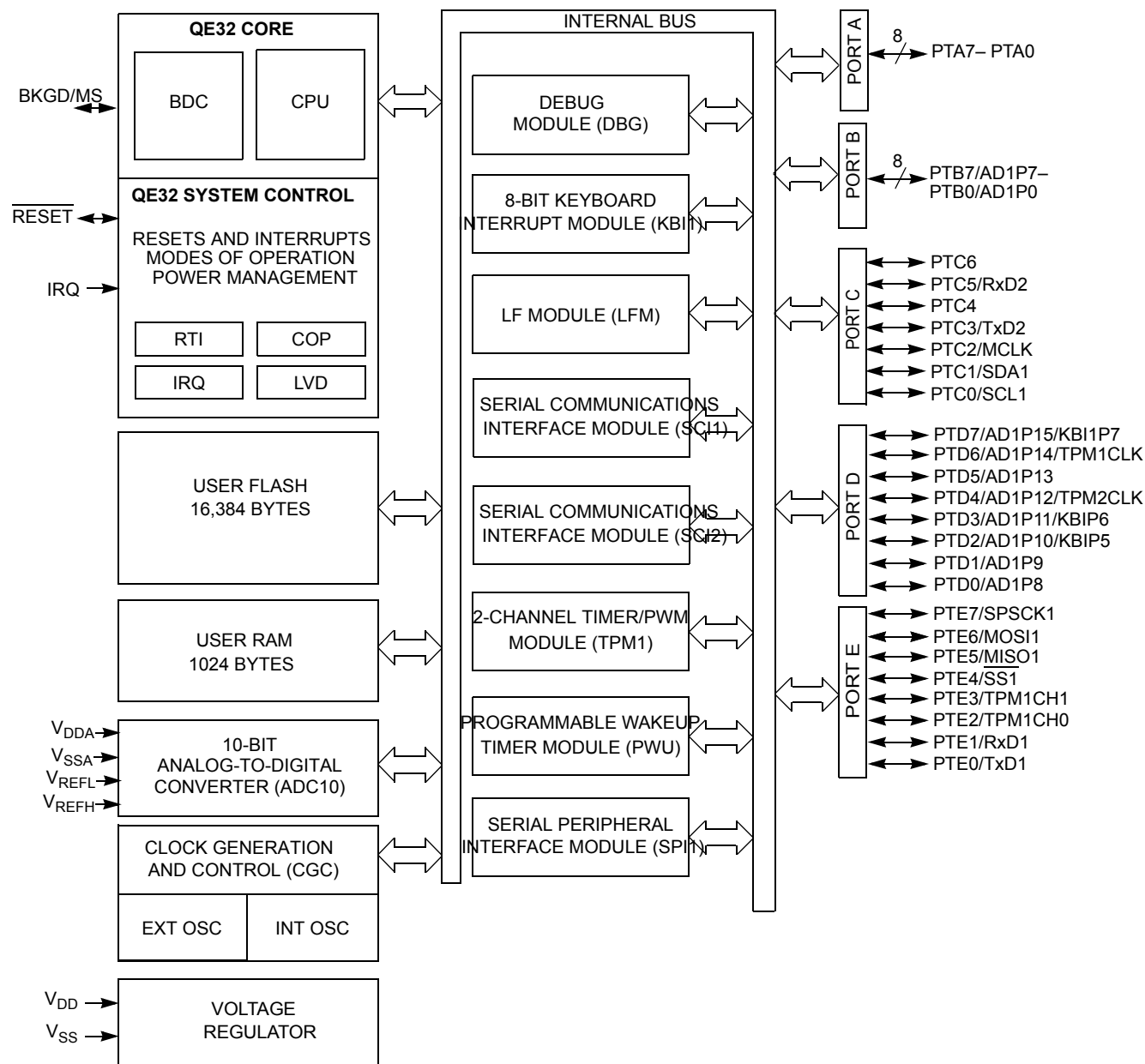


Figure 16-12. MC9S08QE128 Series Block Diagram

Module Initialization:

Write: SCIBDH:SCIBDL to set baud rate
 Write: SCFC1 to configure 1-wire/2-wire, 9/8-bit data, wakeup, and parity, if used.
 Write; SCIC2 to configure interrupts, enable Rx and Tx, RWU
 Enable Rx wakeup, SBK sends break character
 Write: SCIC3 to enable Rx error interrupt sources. Also controls pin direction in 1-wire modes. R8 and T8 only used in 9-bit data modes.

Module Use:

Wait for TDRE, then write data to SCID
 Wait for RDRF, then read data from SCID
 A small number of applications will use RWU to manage automatic receiver wakeup, SBK to send break characters, and R8 and T8 for 9-bit data.

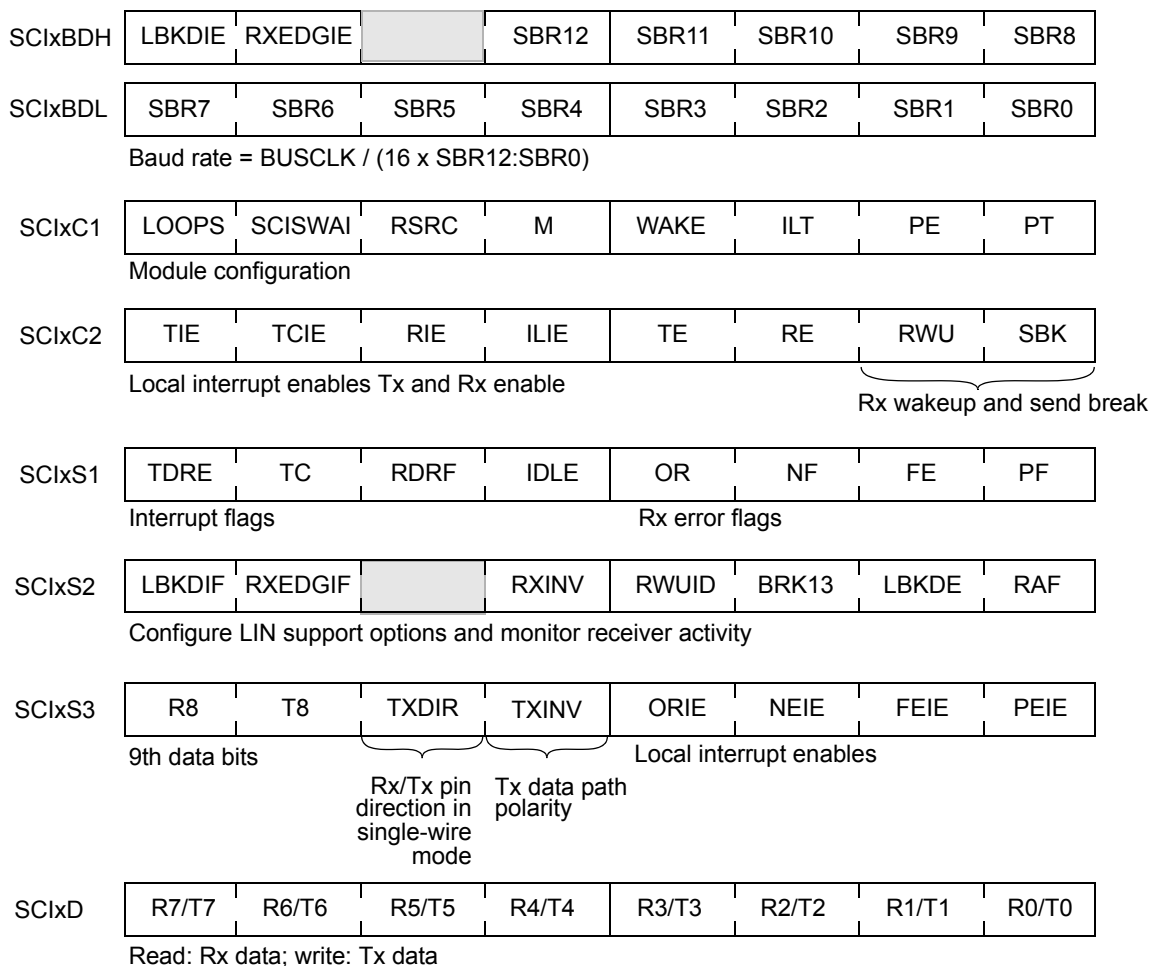


Figure 16-13. SCI Module Quick Start



Chapter 17

Inter-Integrated Circuit (IIC)

17.1 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.

NOTE

9S08QE32 series devices do not include stop1 mode. Please ignore reference to stop1.

NOTE

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

17.1.1 Module Configuration

The IIC module pins, SDA and SCL can be repositioned under software control using IICPS in SOPT2 as shown in [Table 17-1](#). IICPS in SOPT2 selects which general-purpose I/O ports are associated with IIC operation.

Table 17-1. IIC Position Options

IICPS in SOPT2	Port Pin for SDA	Port Pin for SCL
0 (default)	PTA2	PTA3
1	PTB6	PTB7

17.1.2 IIC Clock Gating

The bus clock to the IIC can be gated on and off using the IIC bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the IIC bit can be cleared to disable the clock to this module when not in use.

17.1.3 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 {`iic_ack.asm`}

- 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 {iic_transmit.asm} {iic_receive.asm} {iic_receive_addon.asm}
- Repeated start signal generation {iic_transmit.asm}
- Acknowledge bit generation/detection {iic_ack.asm}
- Bus busy detection {iic_bus_busy.asm}
- General call recognition
- 10-bit address extension

17.1.4 Modes of Operation

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

- **Run mode** — This is the basic mode of operation. To conserve power in this mode, disable the module.
- **Wait mode** — The module continues to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- **Stop mode** — The IIC is inactive in stop3 mode for reduced power consumption. The stop instruction does not affect IIC register states. Stop2 and stop1 resets the register contents.

17.1.5 Block Diagram

Figure 17-1 is a block diagram of the IIC.

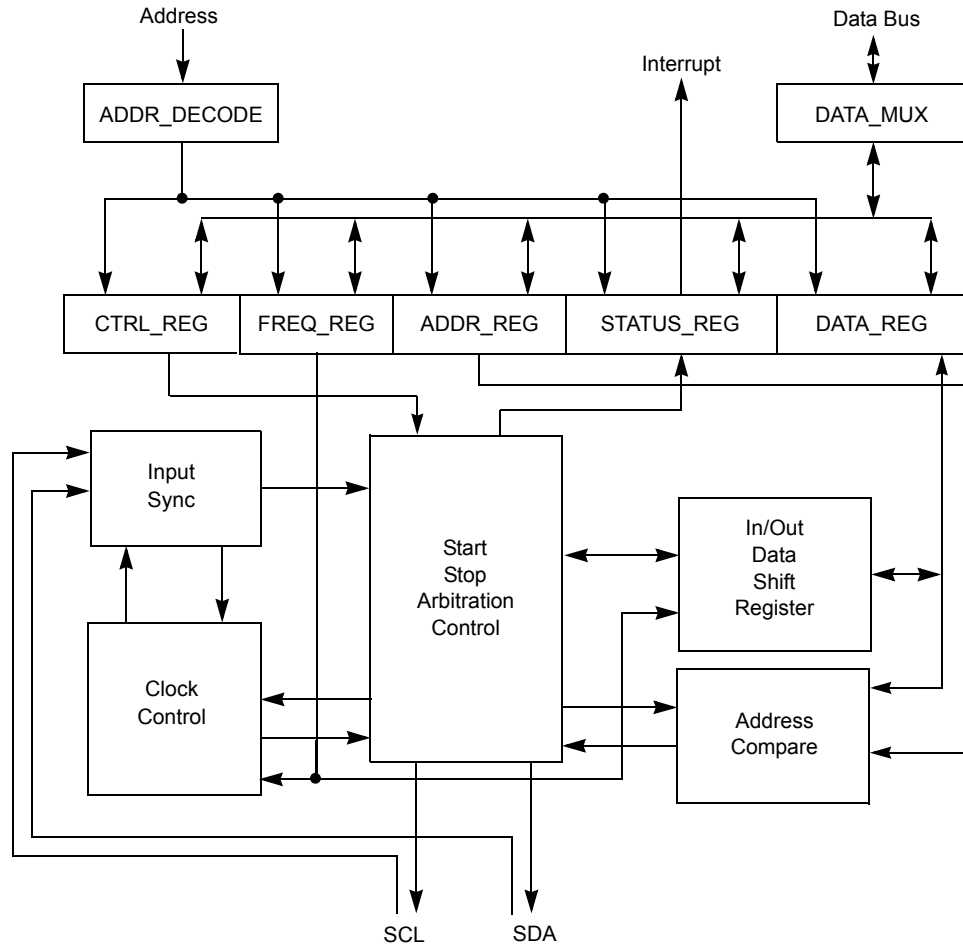


Figure 17-1. IIC Functional Block Diagram

17.2 Signal Description

Table 17-2 shows the user-accessible signals for the IIC.

Table 17-2. Signal Properties

Name	Function
SCL	Serial clock line
SDA	Serial data line

17.3 External Signal Description

This section describes each user-accessible pin signal.

17.3.1 SCL — Serial Clock Line

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

17.3.2 SDA — Serial Data Line

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

17.4 Register Definition

17.4.1 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. [Table 17-3](#) summarizes the IIC module’s address space. The following section describes the bit-level arrangement and functionality of each register.

Table 17-3.

Address	Use	Access
Base + 0x0000	IIC Address Register (IICA)	read/write
Base + 0x0001	IIC Frequency Divider Register (IICF)	read/write
Base + 0x0002	IIC Control Register (IICC1)	read/write
Base + 0x0003	IIC Status Register (IICS)	read
Base + 0x0004	IIC Data IO Register (IICD)	read/write
Base + 0x0005	IIC Control Register 2 (IICC2)	read/write

17.4.2 Register Descriptions

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

Refer to the direct-page register summary in the [memory](#) chapter of this document for the absolute address assignments for all IIC registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

17.4.3 IIC Address Register (IIC1A)

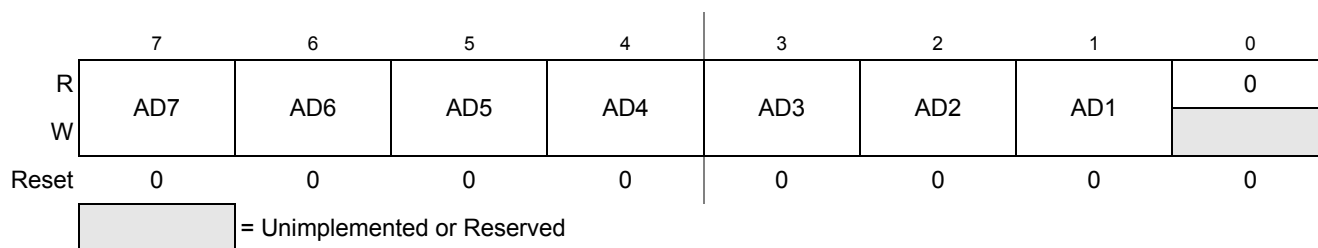


Figure 17-2. IIC Address Register (IIC1A)

Table 17-4. IIC1A Field Descriptions

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

17.4.4 IIC Frequency Divider Register (IIC1F)


Figure 17-3. IIC Frequency Divider Register (IIC1F)
Table 17-5. IIC1F Field Descriptions

Field	Description
7–6 MULT	<p>IIC Multiplier Factor. The MULT bits define the multiplier factor, mul. This factor, along with the SCL divider, generates the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below.</p> <p>00 mul = 01 01 mul = 02 10 mul = 04 11 Reserved</p>
5–0 ICR	<p>IIC Clock Rate. The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits determine the IIC baud rate, the SDA hold time, the SCL Start hold time, and the SCL Stop hold time. Table 17-8 provides the SCL divider and hold values for corresponding values of the ICR.</p> <p>The SCL divider multiplied by multiplier factor mul generates IIC baud rate.</p> $\text{IIC baud rate} = \frac{\text{bus speed (Hz)}}{\text{mul} \times \text{SCLdivider}} \quad \text{Eqn. 17-1}$ <p>SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data).</p> $\text{SDA hold time} = \text{bus period (s)} \times \text{mul} \times \text{SDA hold value} \quad \text{Eqn. 17-2}$ <p>SCL start hold time is the delay from the falling edge of SDA (IIC data) while SCL is high (Start condition) to the falling edge of SCL (IIC clock).</p> $\text{SCL Start hold time} = \text{bus period (s)} \times \text{mul} \times \text{SCL Start hold value} \quad \text{Eqn. 17-3}$ <p>SCL stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA (IIC data) while SCL is high (Stop condition).</p> $\text{SCL Stop hold time} = \text{bus period (s)} \times \text{mul} \times \text{SCL Stop hold value} \quad \text{Eqn. 17-4}$

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

Table 17-6. Hold Time Values for 8 MHz Bus Speed

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

Non-Customer Text:

Table 17-7. IIC Tap and Prescale Values

TAP1 (bin)	scltap (clocks)	sdatap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

TAP2 (bin)	scl2tap (clocks)	tap2tap (clocks)
000	4	1
001	4	2
010	6	4
011	6	8
100	14	16
101	30	32
110	62	64
111	126	128

The number of clocks from the falling edge of SCL to the first tap (TAP[1]) is defined by the values shown in the scl2tap column of [Table 17-7](#), all subsequent tap points are separated by 2^{TAP2} as shown in the tap2tap column in [Table 17-7](#). The scltap is used to generate the SCL period and the sdatap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.

MULT defines the multiplier factor mul. The values of mul are shown in the [Table 17-5](#).

The equation used to generate the divider values from the IICF bits is:

$$\text{SCL divider} = \text{mul} \times \{2 \times (\text{scl2tap} + [(\text{scltap} - 1) \times \text{tap2tap}] + 2)\} \quad \text{Eqn. 17-5}$$

The SDA hold delay is equal to the CPU clock period multiplied by the SDA hold value.

The equation used to generate the SDA hold value from the IICF bits is:

$$\text{SDA hold} = \text{mul} \times \{ \text{scl2tap} + [(\text{sdatap} - 1) \times \text{tap2tap}] + 3 \}$$

Table 17-8. IIC Divider and Hold Values

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

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

17.4.5 IIC Control Register (IIC1C1)

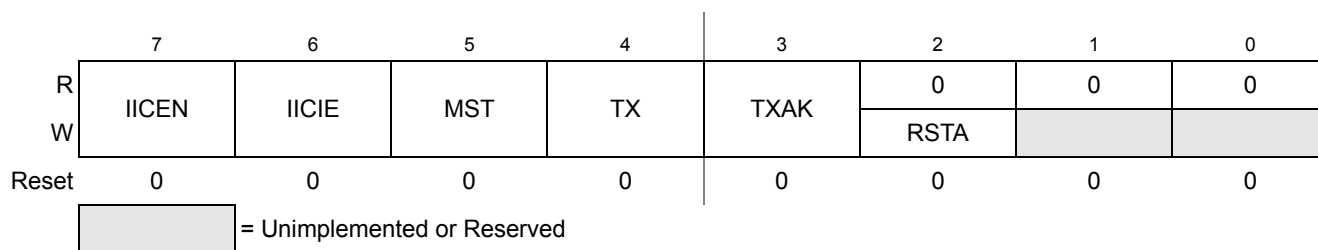


Figure 17-4. IIC Control Register (IIC1C1)

Table 17-9. IIC1C1 Field Descriptions

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

17.4.6 IIC Status Register (IIC1S)

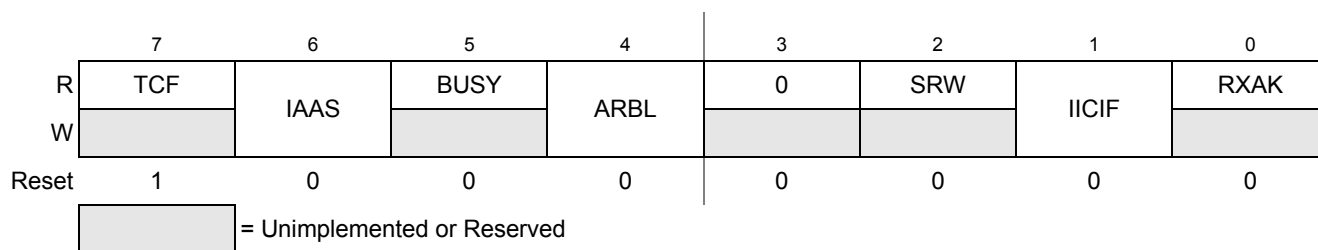


Figure 17-5. IIC Status Register (IIC1S)

Table 17-10. IIC1S Field Descriptions

Field	Description
7 TCF	<p>Transfer Complete Flag. This bit is set on the completion of a byte transfer. This bit is only valid during or immediately following a transfer to the IIC module or from the IIC module. The TCF bit is cleared by reading the IIC1D register in receive mode or writing to the IIC1D in transmit mode.</p> <p>0 Transfer in progress 1 Transfer complete</p>
6 IAAS	<p>Addressed as a Slave. The IAAS bit is set when the calling address matches the programmed slave address or when the GCAEN bit is set and a general call is received. Writing the IIC1C register clears this bit.</p> <p>0 Not addressed 1 Addressed as a slave</p>
5 BUSY	<p>Bus Busy. The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a start signal is detected and cleared when a stop signal is detected.</p> <p>0 Bus is idle 1 Bus is busy</p>
4 ARBL	<p>Arbitration Lost. This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software by writing a 1 to it.</p> <p>0 Standard bus operation 1 Loss of arbitration</p>
2 SRW	<p>Slave Read/Write. When addressed as a slave, the SRW bit indicates the value of the R/W command bit of the calling address sent to the master.</p> <p>0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave</p>
1 IICIF	<p>IIC Interrupt Flag. The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit:</p> <ul style="list-style-type: none"> • One byte transfer completes • Match of slave address to calling address • Arbitration lost <p>0 No interrupt pending 1 Interrupt pending</p>
0 RXAK	<p>Receive Acknowledge. When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected.</p> <p>0 Acknowledge received 1 No acknowledge received</p>

17.4.7 IIC Data I/O Register (IIC1D)

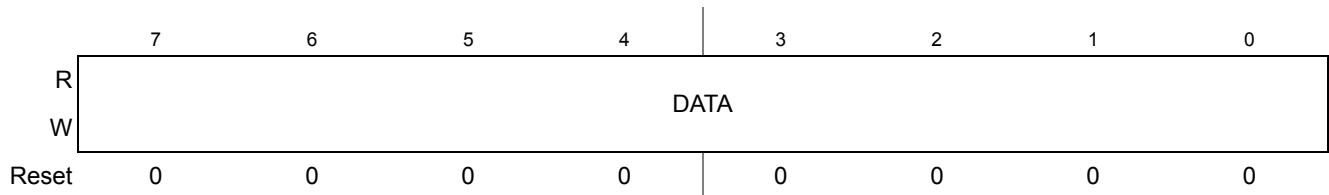


Figure 17-6. IIC Data I/O Register (IIC1D)

Table 17-11. IIC1D Field Descriptions

Field	Description
7-0 DATA	Data — 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.

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, reading the IIC1D does not initiate the receive.

Reading the IIC1D returns the last byte received while the IIC is configured in master receive or slave receive modes. The IIC1D does not reflect every byte 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 to bit 1) concatenated with the required R/W bit (in position bit 0).

17.4.8 IIC Control Register 2 (IIC1C2)

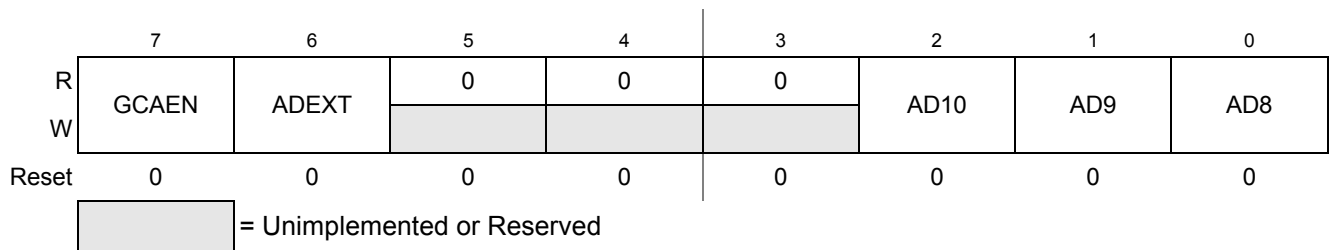


Figure 17-7. IIC Control Register (IIC1C2)

Table 17-12. IIC1C2 Field Descriptions

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

17.5 Functional Description

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

17.5.1 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 17-8](#).

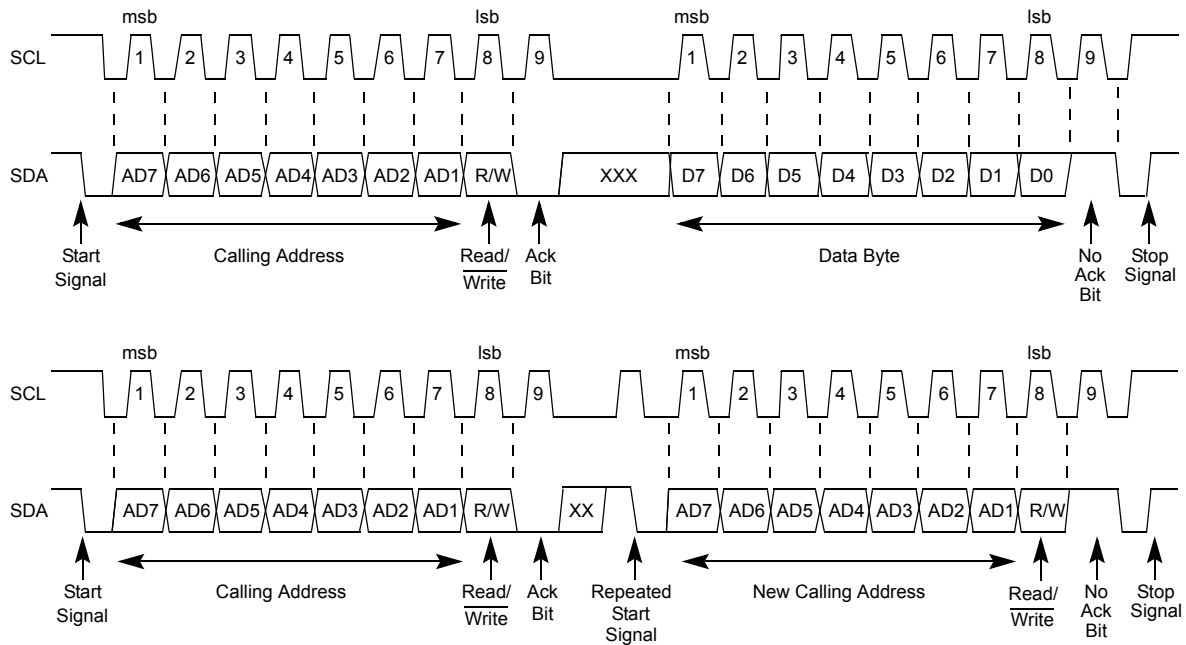


Figure 17-8. IIC Bus Transmission Signals

17.5.1.1 Start Signal

When the bus is free, no master device is engaging the bus (SCL and SDA lines are at logical high), a master may initiate communication by sending a start signal. As shown in Figure 17-8, 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.

17.5.1.2 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/\bar{W} bit. The R/\bar{W} bit tells the slave the desired direction of data transfer.

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

Only the slave with a calling address that matches the one transmitted by the master responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 17-8).

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

17.5.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the $\overline{R/W}$ 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 17-8](#). There is one clock pulse on SCL for each data bit, the msb being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

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

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

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

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

17.5.1.4 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 17-8](#)).

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

17.5.1.5 Repeated Start Signal

As shown in [Figure 17-8](#), 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.

17.5.1.6 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.

17.5.1.7 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 17-9](#)). 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.

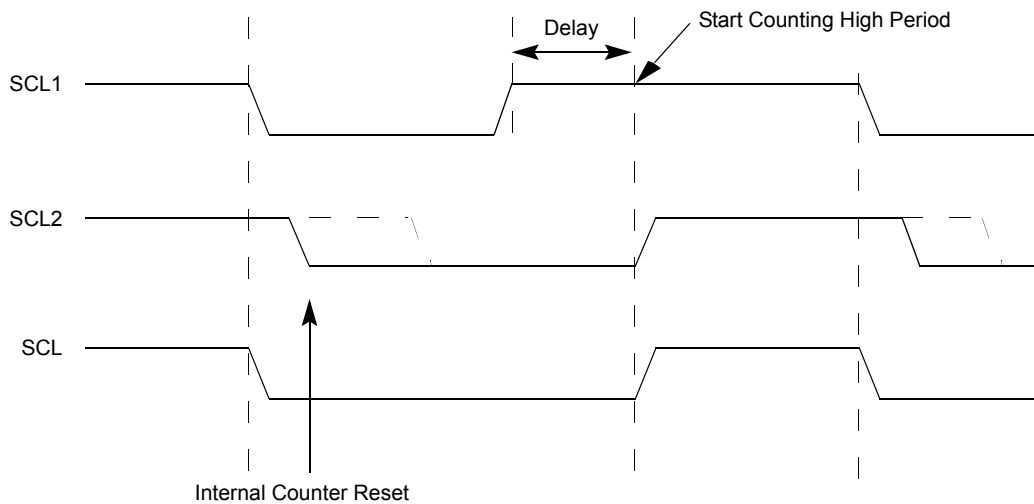


Figure 17-9. IIC Clock Synchronization

17.5.1.8 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 a case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

17.5.1.9 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.

17.5.2 10-bit Address

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

17.5.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 17-13). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/\overline{W} direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

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

Table 17-13. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

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

17.5.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/\overline{W} bit (see Table 17-14). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/\overline{W}) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

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

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

Table 17-14. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address

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

17.5.3 General Call Address

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

17.6 Resets

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

17.7 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in [Table 17-15](#) occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. You can determine the interrupt type by reading the status register.

Table 17-15. 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

17.7.1 Byte Transfer Interrupt

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

17.7.2 Address Detect Interrupt

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

17.7.3 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 writing a 1 to it.

17.8 Initialization/Application Information

Module Initialization (Slave)

1. Write: IICC2
 - to enable or disable general call
 - to select 10-bit or 7-bit addressing mode
2. Write: IICA
 - to set the slave address
3. Write: IICC1
 - to enable IIC and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in [Figure 17-11](#)

Module Initialization (Master)

1. Write: IICF
 - to set the IIC baud rate (example provided in this chapter)
2. Write: IICC1
 - to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in [Figure 17-11](#)
5. Write: IICC1
 - to enable TX
6. Write: IICC1
 - to enable MST (master mode)
7. Write: IICD
 - with the address of the target slave. (The lsb of this byte determines whether the communication is master receive or transmit.)

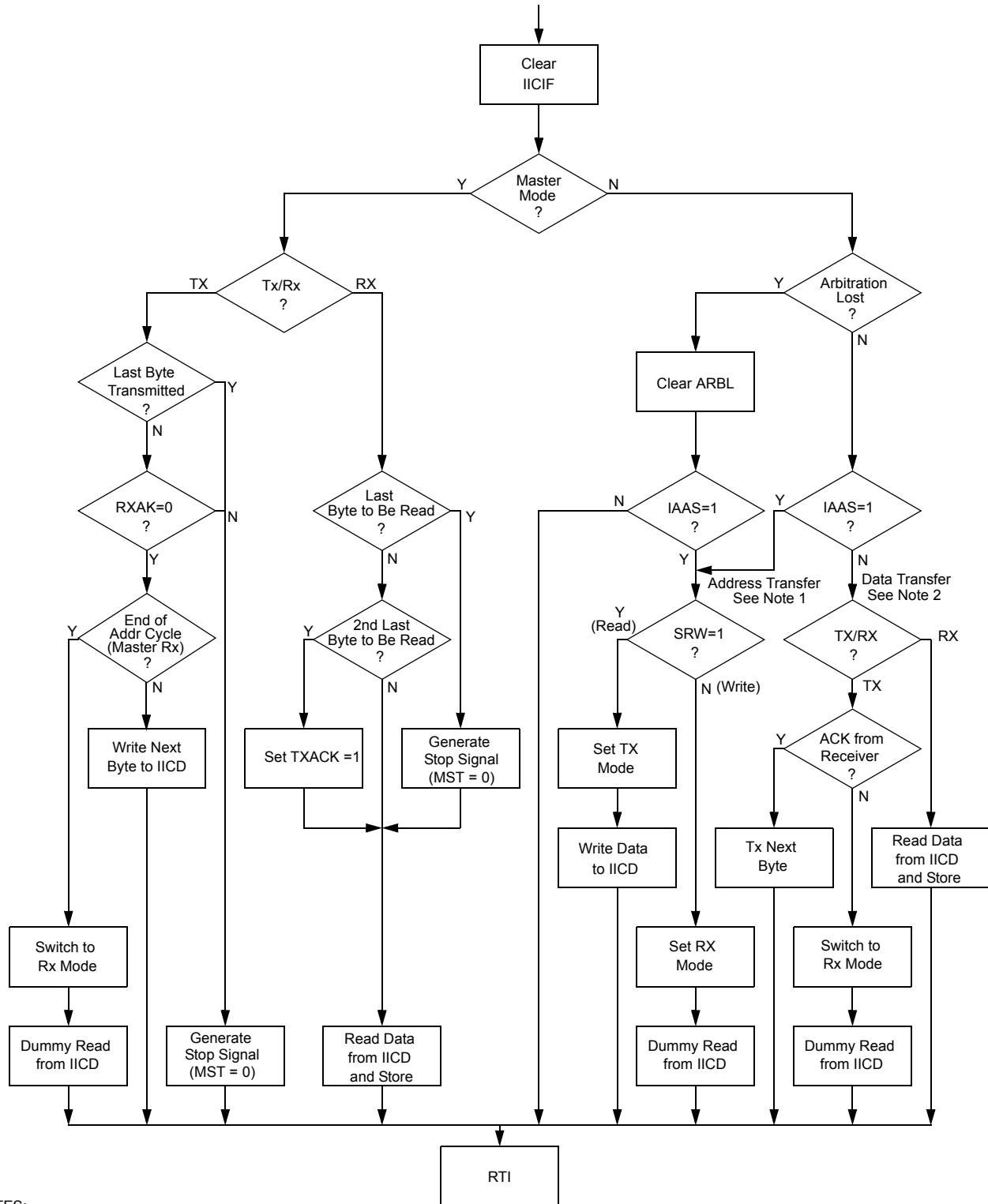
Module Use

The routine shown in [Figure 17-11](#) can handle both master and slave IIC operations. For slave operation, an incoming IIC message that contains the proper address begins IIC communication. For master operation, communication must be initiated by writing to the IICD register.

Register Model

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

Figure 17-10. IIC Module Quick Start



NOTES:

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

Figure 17-11. Typical IIC Interrupt Routine

Chapter 18

Analog to Digital (ATD) Module

18.1 Introduction

The 12-bit analog-to-digital converter (ADC12) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

The bandgap channel cannot be converted in low power run mode or low power wait mode.

NOTE

Stop1 mode is not available in this device.

18.1.1 ADC Clock Gating

The bus clock to the ADC can be gated on and off using the ADC bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ADC bit can be cleared to disable the clock to this module when not in use.

18.1.2 Module Configurations

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

18.1.2.1 Channel Assignments

The ADC channel assignments for the series devices are shown in [Table 18-1](#). Reserved channels convert to an unknown value.

Table 18-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	AD0	PTA0/ADP0	ADPC0	10000	AD16	Reserved	N/A
00001	AD1	PTA1/ADP1	ADPC1	10001	AD17	Reserved	N/A
00010	AD2	PTA2/ADP2	ADPC2	10010	AD18	Reserved	N/A
00011	AD3	PTA3/ADP3	ADPC3	10011	AD19	Reserved	N/A
00100	AD4	PTB0/ADP4	ADPC4	10100	AD20	Reserved	N/A
00101	AD5	PTB1/ADP5	ADPC5	10101	AD21	Reserved	N/A
00110	AD6	PTB2/ADP6	ADPC6	10110	AD22	Reserved	N/A
00111	AD7	PTB3/ADP7	ADPC7	10111	AD23	Reserved	N/A
01000	AD8	PTA6/ADP8	ADPC8	11000	AD24	Reserved	N/A
01001	AD9	PTA7/ADP9	ADPC9	11001	AD25	Reserved	N/A
01010	AD10	Reserved	N/A	11010	AD26	Temperature Sensor	N/A
01011	AD11	Reserved	N/A	11011	AD27	Internal Bandgap	N/A
01100	AD12	Reserved	N/A	11100	—	Reserved	N/A

Table 18-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
01101	AD13	Reserved	N/A	11101	V _{REFH}	V _{DD}	N/A
01110	AD14	Reserved	N/A	11110	V _{REFL}	V _{SS}	N/A
01111	AD15	Reserved	N/A	11111	Module Disabled	None	N/A

NOTE

Selecting the internal bandgap channel requires BGBE = 1 in the SPMSC1 register.

18.1.2.2 Alternate Clock

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

18.1.2.3 Hardware Trigger

This RTC can be enabled as a hardware trigger for the ADC module by setting the ADTRG bit in the ADCSC2 register. When enabled, the ADC will be triggered every time RTCINT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the ADC.

The RTC can be clocked by either ICSIRCLK, OSCOUT or LPO. The period of the RTC is determined by the input clock frequency and the RTC configuration bits. When the hardware trigger is enabled, a conversion is initiated upon a RTC overflow.

The RTC can be configured to cause a hardware trigger in MCU run, wait, low power run, low power wait, and stop3.

18.1.2.4 Temperature Sensor

The ADC1 module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 18-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{\text{TEMP}} - V_{\text{TEMP}25}) \div m) \quad \text{Eqn. 18-1}$$

where:

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

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

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in Equation 18-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 18-1.

18.1.3 Features

Features of the ADC module include:

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

18.1.4 ADC Module Block Diagram

Figure 18-1 provides a block diagram of the ADC module.

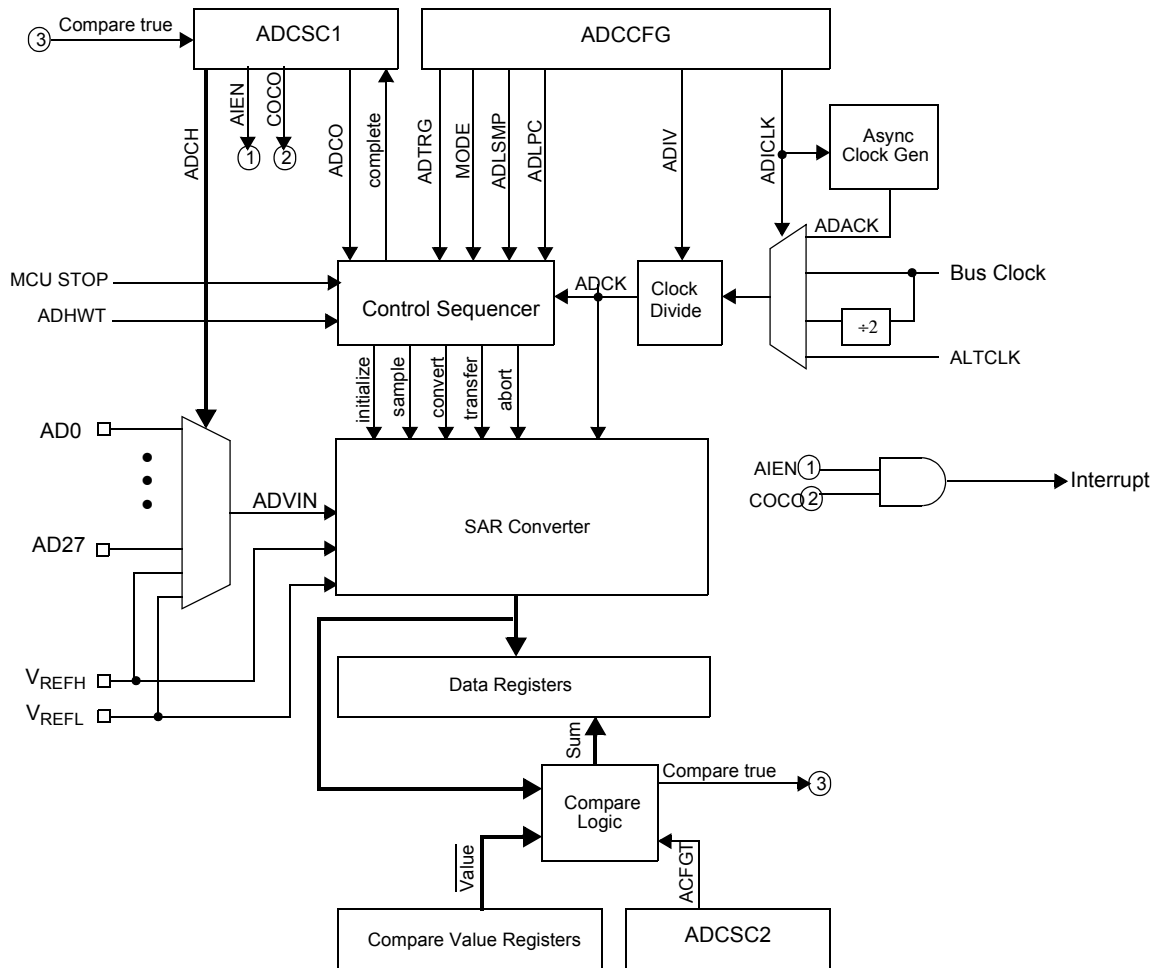


Figure 18-1. ADC Block Diagram

18.2 External Signal Description

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

Table 18-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V _{REFH}	High reference voltage
V _{REFL}	Low reference voltage
V _{DDA}	Analog power supply
V _{SSA}	Analog ground

18.2.1 Analog Power (V_{DDA})

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

18.2.2 Analog Ground (V_{SSA})

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

18.2.3 Voltage Reference High (V_{REFH})

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

18.2.4 Voltage Reference Low (V_{REFL})

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

18.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

18.3 Register Definition

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

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

18.3.1 Status and Control Register 1 (ADCSC1)

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


Figure 18-2. Status and Control Register (ADCSC1)
Table 18-3. ADCSC1 Field Descriptions

Field	Description
7 COCO	Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared when ADCSC1 is written or when ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	Interrupt Enable AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	Continuous Conversion Enable. ADCO enables continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	Input Channel Select. The ADCH bits form a 5-bit field that selects one of the input channels. The input channels are detailed in Table 18-4 . The successive approximation converter subsystem is turned off when the channel select bits are all set. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Table 18-4. Input Channel Select

ADCH	Input Select
00000–01111	AD0–15
10000–11011	AD16–27
11100	Reserved
11101	V _{REFH}
11110	V _{REFL}
11111	Module disabled

18.3.2 Status and Control Register 2 (ADCSC2)

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

	7	6	5	4	3	2	1	0
Reset:	0	0	0	0	0	0	0	0

Figure 18-3. Status and Control Register 2 (ADCSC2)

Table 18-5. ADCSC2 Register Field Descriptions

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

18.3.3 Data Result High Register (ADCRH)

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

When automatic compare is not enabled, the value stored in ADCRH are the upper bits of the conversion result. When automatic compare is enabled, the conversion result is manipulated as described in [Section 18.4.5, “Automatic Compare Function”](#) prior to storage in ADCRH:ADCRL registers.

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

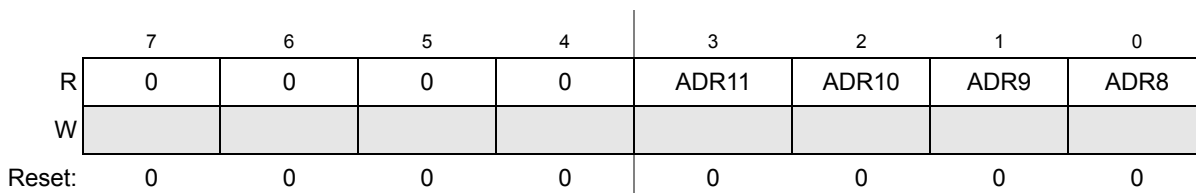


Figure 18-4. Data Result High Register (ADCRH)

18.3.4 Data Result Low Register (ADCRL)

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

When automatic compare is not enabled, the value stored in ADCRL is the lower eight bits of the conversion result. When automatic compare is enabled, the conversion result is manipulated as described in [Section 18.4.5, “Automatic Compare Function”](#) prior to storage in ADCRH:ADCRL registers.

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

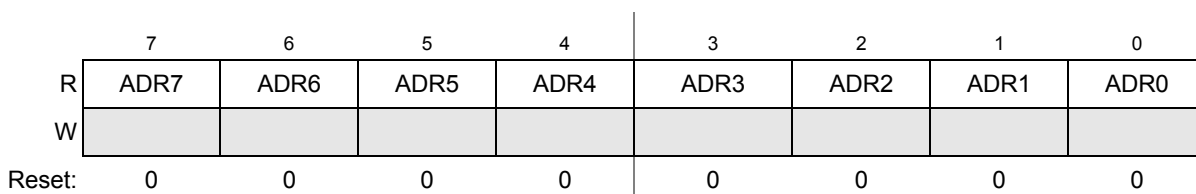


Figure 18-5. Data Result Low Register (ADCRL)

18.3.5 Compare Value High Register (ADCCVH)

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

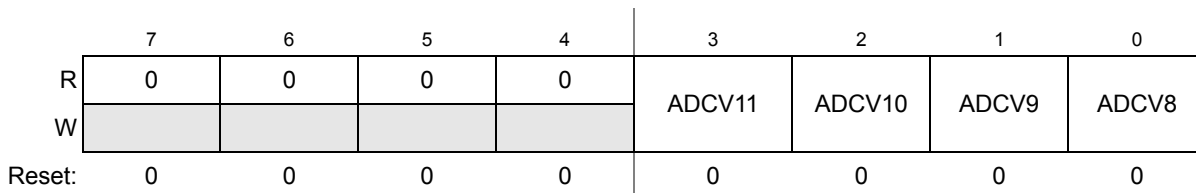


Figure 18-6. Compare Value High Register (ADCCVH)

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

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

18.3.6 Compare Value Low Register (ADCCVL)

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

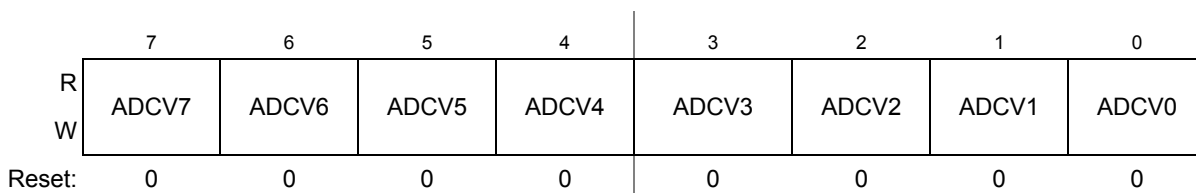


Figure 18-7. Compare Value Low Register(ADCCVL)

18.3.7 Configuration Register (ADCCFG)

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

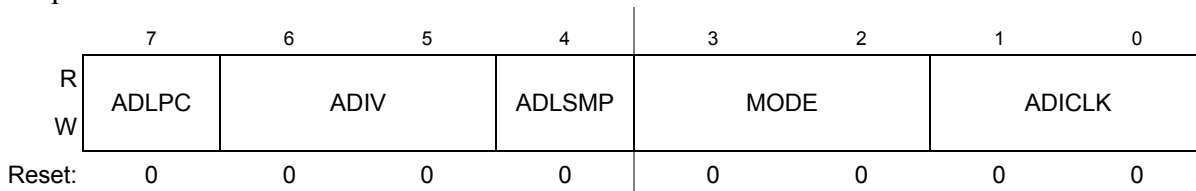


Figure 18-8. Configuration Register (ADCCFG)

Table 18-6. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	Low-Power Configuration. ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: The power is reduced at the expense of maximum clock speed.
6:5 ADIV	Clock Divide Select. ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 18-7 shows the available clock configurations.
4 ADLSMP	Long Sample Time Configuration. ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	Conversion Mode Selection. MODE bits are used to select between 12-, 10-, or 8-bit operation. See Table 18-8 .
1:0 ADICLK	Input Clock Select. ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 18-9 .

Table 18-7. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 18-8. Conversion Modes

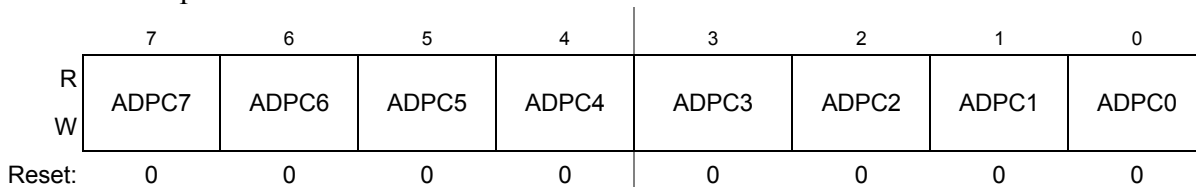
MODE	Mode Description
00	8-bit conversion (N=8)
01	12-bit conversion (N=12)
10	10-bit conversion (N=10)
11	Reserved

Table 18-9. Input Clock Select

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

18.3.8 Pin Control 1 Register (APCTL1)

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


Figure 18-9. Pin Control 1 Register (APCTL1)
Table 18-10. APCTL1 Register Field Descriptions

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

Table 18-10. APCTL1 Register Field Descriptions (continued)

Field	Description
1 ADPC1	ADC Pin Control 1. ADPC1 controls the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	ADC Pin Control 0. ADPC0 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

18.3.9 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.

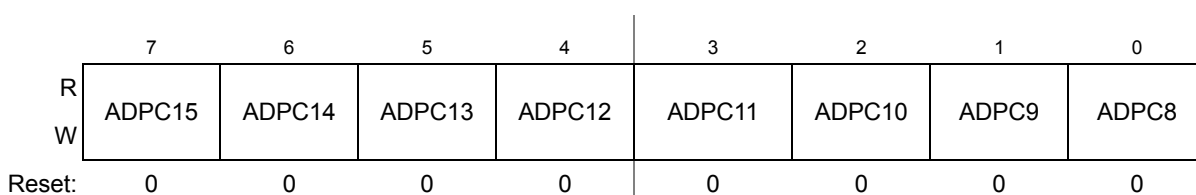


Figure 18-10. Pin Control 2 Register (APCTL2)

Table 18-11. APCTL2 Register Field Descriptions

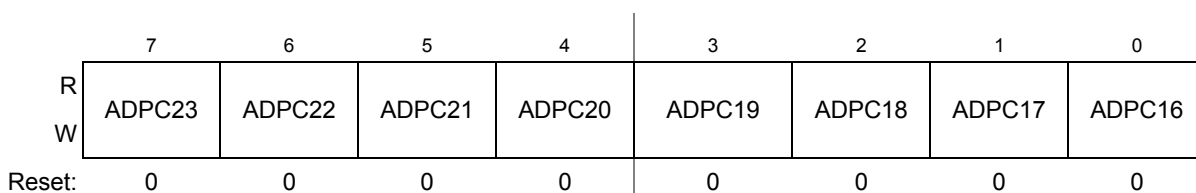
Field	Description
7 ADPC15	ADC Pin Control 15. ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	ADC Pin Control 14. ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control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

Table 18-11. APCTL2 Register Field Descriptions (continued)

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

18.3.10 Pin Control 3 Register (APCTL3)

APCTL3 controls channels 16–23 of the ADC module.


Figure 18-11. Pin Control 3 Register (APCTL3)
Table 18-12. APCTL3 Register Field Descriptions

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

Table 18-12. APCTL3 Register Field Descriptions (continued)

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

18.4 Functional Description

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

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

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

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

18.4.1 Clock Select and Divide Control

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

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

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

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

18.4.2 Input Select and Pin Control

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

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

18.4.3 Hardware Trigger

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

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

18.4.4 Conversion Control

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

18.4.4.1 Initiating Conversions

A conversion is initiated:

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

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

18.4.4.2 Completing Conversions

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

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

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

18.4.4.3 Aborting Conversions

Any conversion in progress is aborted when:

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

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

18.4.4.4 Power Control

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

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

18.4.4.5 Sample Time and Total Conversion Time

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

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

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

The maximum total conversion time for different conditions is summarized in [Table 18-13](#).

Table 18-13. Total Conversion Time vs. Control Conditions

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μ s + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 μ s + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μ s + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 μ s + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

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

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

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

NOTE

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

18.4.5 Automatic Compare Function

The compare function is enabled by the ACFE bit. The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the compare value (ADCCVH and ADCCVL) is subtracted from the conversion result. When comparing to an upper limit (ACFGT = 1), if the conversion result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

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

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers.

NOTE

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

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

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

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

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

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

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

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

```

%01 0111 1111 <= Complement of 10-bit value stored in ADCRH:ADCRL
+           %1
-----
%01 1000 0000 <= Unsigned value 0x180 is the absolute value of (Result - Compare Value)
    
```

18.4.6 MCU Wait Mode Operation

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

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

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

18.4.7 MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

18.4.7.1 Stop3 Mode With ADACK Disabled

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

18.4.7.2 Stop3 Mode With ADACK Enabled

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

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

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

NOTE

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the data transfer blocking mechanism (discussed in [Section 18.4.4.2, "Completing Conversions"](#)) is cleared when entering stop3 and continuing ADC conversions.

18.4.8 MCU Stop2 Mode Operation

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

18.5 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to [Table 18-7](#), [Table 18-8](#), and [Table 18-9](#) for information used in this example.

NOTE

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

18.5.1 ADC Module Initialization Example

18.5.1.1 Initialization Sequence

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

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

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

18.5.1.2 Pseudo-Code Example

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

ADCCFG = 0x98 (%10011000)

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

ADCSC2 = 0x00 (%00000000)

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

ADCSC1 = 0x41 (%01000001)

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

ADCRH/L = 0xxx

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

ADCCVH/L = 0xxx

Holds compare value when compare function enabled

APCTL1=0x02

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

APCTL2=0x00

All other AD pins remain general purpose I/O pins

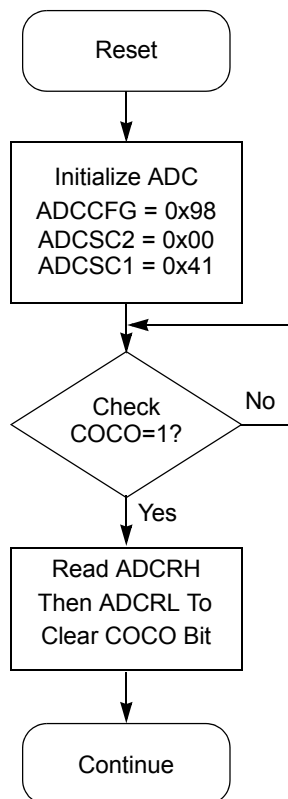


Figure 18-12. Initialization Flowchart for Example

18.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

18.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

18.6.1.1 Analog Supply Pins

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

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

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

18.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH} , which may be shared on the same pin as V_{DDA} on some devices. The low reference is V_{REFL} , which may be shared on the same pin as V_{SSA} on some devices.

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

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

18.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at V_{DD} or V_{SS} . Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

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

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL} . If the input is equal to or exceeds V_{REFH} , the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than V_{REFL} , the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling

capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

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

18.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

18.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately $7\text{k}\Omega$ and input capacitance of approximately 5.5 pF , sampling to within $1/4\text{LSB}$ (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below $2\text{ k}\Omega$.

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

18.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDA} / (2^N * I_{LEAK})$ for less than $1/4\text{LSB}$ leakage error ($N = 8$ in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

18.6.2.3 Noise-Induced Errors

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

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

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

- Place a 0.01 μF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSA} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

18.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1 \text{ lsb} = (V_{REFH} - V_{REFL}) / 2^N \quad \text{Eqn. 18-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be $\pm 1/2$ lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is -1 lsb to 0 lsb and the code width of each step is 1 lsb.

18.6.2.5 Linearity Errors

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

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.

- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

18.6.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

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

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around $\pm 1/2$ lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 18.6.2.3](#) reduces this error.

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

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

Chapter 19

Analog Comparator 3V

19.1 Introduction

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

Some MCUs may include more than one ACMP, so a text placeholder (x) is used to specify the ACMP, i.e., an MCU with two ACMPs would have ACMP1 and ACMP2.

19.1.1 Features

The ACMP has the following features:

- Full rail to rail supply operation.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Option to compare to fixed internal bandgap reference voltage.
- Option to allow comparator output to be visible on a pin, ACMPxO.

NOTE

Stop1 mode is not available in this device.

19.1.2 ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP1+ and/or ACMP2+, the user must enable the bandgap buffer by setting BGBE = 1 in SPMSC1.

19.1.3 ACMP/TPM Configuration Information

The ACMP modules can be configured to connect the output of the analog comparator to a TPM input capture channel 0 by setting the corresponding ACICx bit in SOPT2. With ACICx set, the TPMxCH0 pin is not available externally regardless of the configuration of the TPMx module.

The ACMP1 output can be connected to TPM1CH0; The ACMP2 output can be connected to TPM2CH0.

19.1.4 ACMP Clock Gating

The bus clock to both of the ACMPs can be gated on and off using the ACMP bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ACMP bit can be cleared to disable the clock to this module when not in use.

19.1.5 Interrupt Vectors

ACMP1 and ACMP2 share a single interrupt vector, located at address 0xFFD6:0xFFD7. When interrupts are enabled for both ACMPs, the ACF bit must be polled in the ACMP1SC and ACMP2SC registers to determine which ACMP caused the interrupt.

19.1.6 Modes of Operation

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

19.1.6.1 ACMP in Wait Mode

The ACMP continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the ACMP can be used to bring the MCU out of wait mode if the ACMP interrupt is enabled (ACIE is set). For lowest possible current consumption, the ACMP should be disabled by software if not required as an interrupt source during wait mode.

19.1.6.2 ACMP in Stop Modes

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

During stop2 mode, the ACMP module is fully powered down. Upon wake-up from stop2 mode, the ACMP module is in the reset state.

During stop3 mode, clocks to the ACMP module are halted. No registers are affected. In addition, the ACMP comparator circuit enters a low-power state. No compare operation occurs while in stop3.

If stop3 is exited with a reset, the ACMP is put into its reset state. If stop3 is exited with an interrupt, the ACMP continues from the state it was in when stop3 was entered.

19.1.6.3 ACMP in Active Background Mode

When the microcontroller is in active background mode, the ACMP continues to operate normally.

19.1.7 Block Diagram

The block diagram for the analog comparator module is shown [Figure 19-1](#).

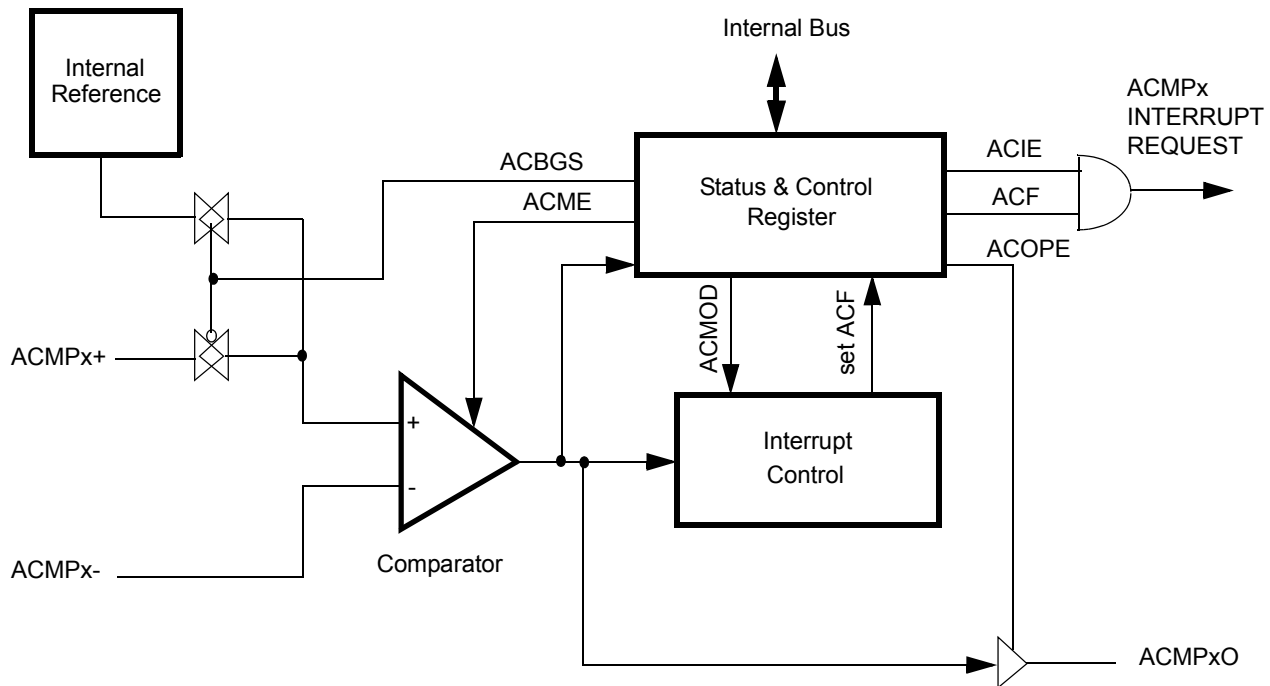


Figure 19-1. Analog Comparator (ACMP) Block Diagram

19.2 External Signal Description

The ACMP has two analog input pins, ACMPx+ and ACMPx– and one digital output pin ACMPxO. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. As shown in [Figure 19-1](#), the ACMPx– pin is connected to the inverting input of the comparator, and the ACMPx+ pin is connected to the comparator non-inverting input if ACBGS is a 0. As shown in [Figure 19-1](#), the ACMPxO pin can be enabled to drive an external pin.

The signal properties of ACMP are shown in [Table 19-1](#).

Table 19-1. Signal Properties

Signal	Function	I/O
ACMPx–	Inverting analog input to the ACMP. (Minus input)	I
ACMPx+	Non-inverting analog input to the ACMP. (Positive input)	I
ACMPxO	Digital output of the ACMP.	O

19.3 Memory Map/Register Definition

The ACMP includes one register:

- An 8-bit status and control register

Refer to the direct-page register summary in the memory section of this document for the absolute address assignments for the ACMP register. This section refers to register and control bits only by their names and relative address offsets.

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

Table 19-2. ACMP Register Summary

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

19.3.1 ACMPx Status and Control Register (ACMPxSC)

ACMPxSC contains the status flag and control bits used to enable and configure the ACMP.

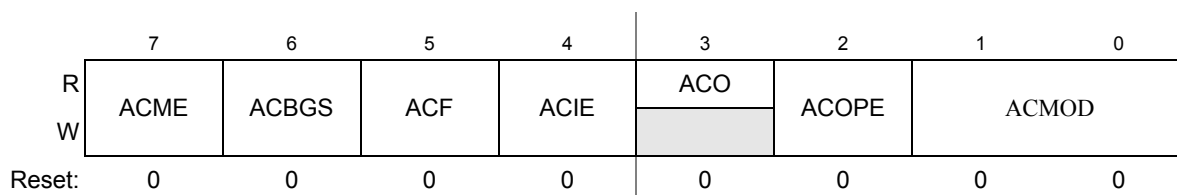


Figure 19-2. ACMPx Status and Control Register (ACMPxSC)

Table 19-3. ACMPxSC Field Descriptions

Field	Description
7 ACME	Analog Comparator Module Enable. Enables the ACMP module. 0 ACMP not enabled 1 ACMP is enabled
6 ACBGS	Analog Comparator Bandgap Select. Selects between the bandgap reference voltage or the ACMPx+ pin as the input to the non-inverting input of the analog comparator. 0 External pin ACMPx+ selected as non-inverting input to comparator 1 Internal reference select as non-inverting input to comparator
5 ACF	Analog Comparator Flag. ACF is set when a compare event occurs. Compare events are defined by ACMOD. ACF is cleared by writing a one to it. 0 Compare event has not occurred 1 Compare event has occurred
4 ACIE	Analog Comparator Interrupt Enable. Enables the interrupt from the ACMP. When ACIE is set, an interrupt is asserted when ACF is set. 0 Interrupt disabled 1 Interrupt enabled

Table 19-3. ACMPxSC Field Descriptions (continued)

Field	Description
3 ACO	Analog Comparator Output. Reading ACO returns the current value of the analog comparator output. ACO is reset to a 0 and reads as a 0 when the ACMP is disabled (ACME = 0).
2 ACOPE	Analog Comparator Output Pin Enable. Enables the comparator output to be placed onto the external pin, ACMPxO. 0 Analog comparator output not available on ACMPxO 1 Analog comparator output is driven out on ACMPxO
1–0 ACMOD	Analog Comparator Mode. ACMOD selects the type of compare event that sets ACF. 00 Encoding 0 — Comparator output falling edge 01 Encoding 1 — Comparator output rising edge 10 Encoding 2 — Comparator output falling edge 11 Encoding 3 — Comparator output rising or falling edge

19.4 Functional Description

The analog comparator can compare two analog input voltages applied to ACMPx+ and ACMPx–, or it can compare an analog input voltage applied to ACMPx– with an internal bandgap reference voltage. ACBGS selects between the bandgap reference voltage or the ACMPx+ pin as the input to the non-inverting input of the analog comparator. The comparator output is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. ACMOD selects the condition that causes ACF to be set. ACF can be set on a rising edge of the comparator output, a falling edge of the comparator output, or a rising or a falling edge (toggle). The comparator output can be read directly through ACO. The comparator output can be driven onto the ACMPxO pin using ACOPE.

Chapter 20

Development Support

20.1 Introduction

Development support systems in the QE32 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip FLASH and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the QE32 Family, address and data bus signals are not available on external pins. Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

20.1.1 Forcing Active Background

The method for forcing active background mode depends on the specific QE32 derivative. For the 9S08QE32 series, you can force active background after a power-on reset by holding the BKGD pin low as the device exits the reset condition. You can also force active background by driving BKGD low immediately after a serial background command that writes a one to the BDFR bit in the SBDFR register. Other causes of reset including an external pin reset or an internally generated error reset ignore the state of the BKGD pin and reset into normal user mode. If no debug pod is connected to the BKGD pin, the MCU will always reset into normal operating mode.

20.1.2 Module Configuration

The alternate BDC clock source is the ICSLCLK. This clock source is selected by clearing the CLKSW bit in the BDCSCR register. For details on ICSLCLK, see the “Functional Description” section of the ICS chapter.

20.1.3 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes

- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system 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:
 - Basic: A-only, A OR B
 - Sequence: A then B
 - Full: A AND B data, A AND NOT B data
 - Event (store data): Event-only B, A then event-only B
 - Range: Inside range ($A \leq \text{address} \leq B$), outside range ($\text{address} < A$ or $\text{address} > B$)

20.2 Background Debug Controller (BDC)

All MCUs in the QE32 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.
- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD} . An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use

power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

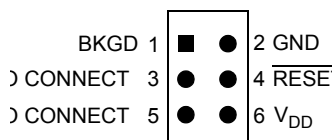


Figure 20-1. BDM Tool Connector

20.2.1 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 20.2.2, "Communication Details"](#).

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 20.2.2, "Communication Details"](#), for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the QE32 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

20.2.2 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 20-2 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target QE32 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.

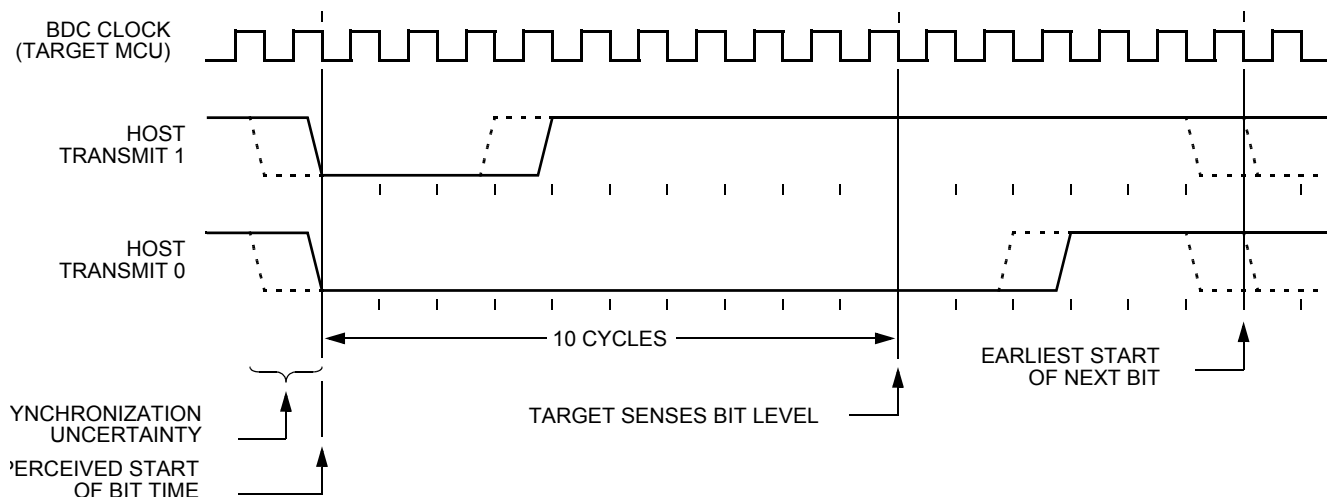


Figure 20-2. BDC Host-to-Target Serial Bit Timing

Figure 20-3 shows the host receiving a logic 1 from the target QE32 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

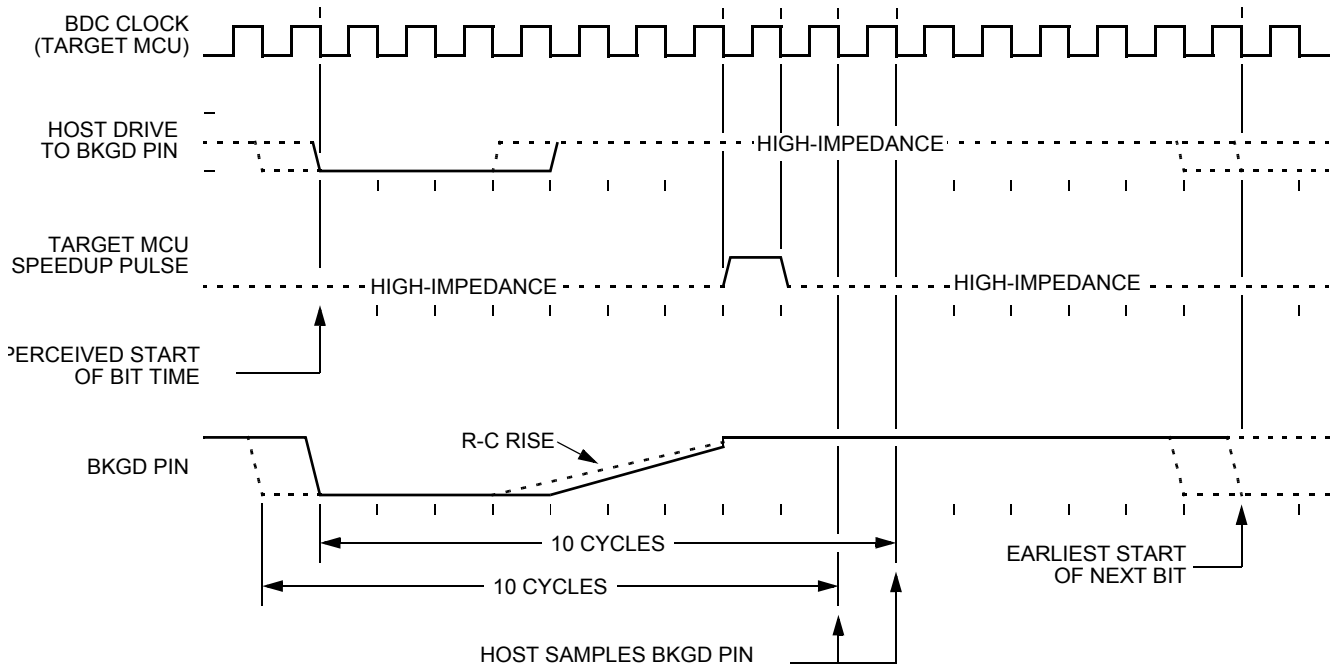


Figure 20-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 20-4 shows the host receiving a logic 0 from the target QE32 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 QE32 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.

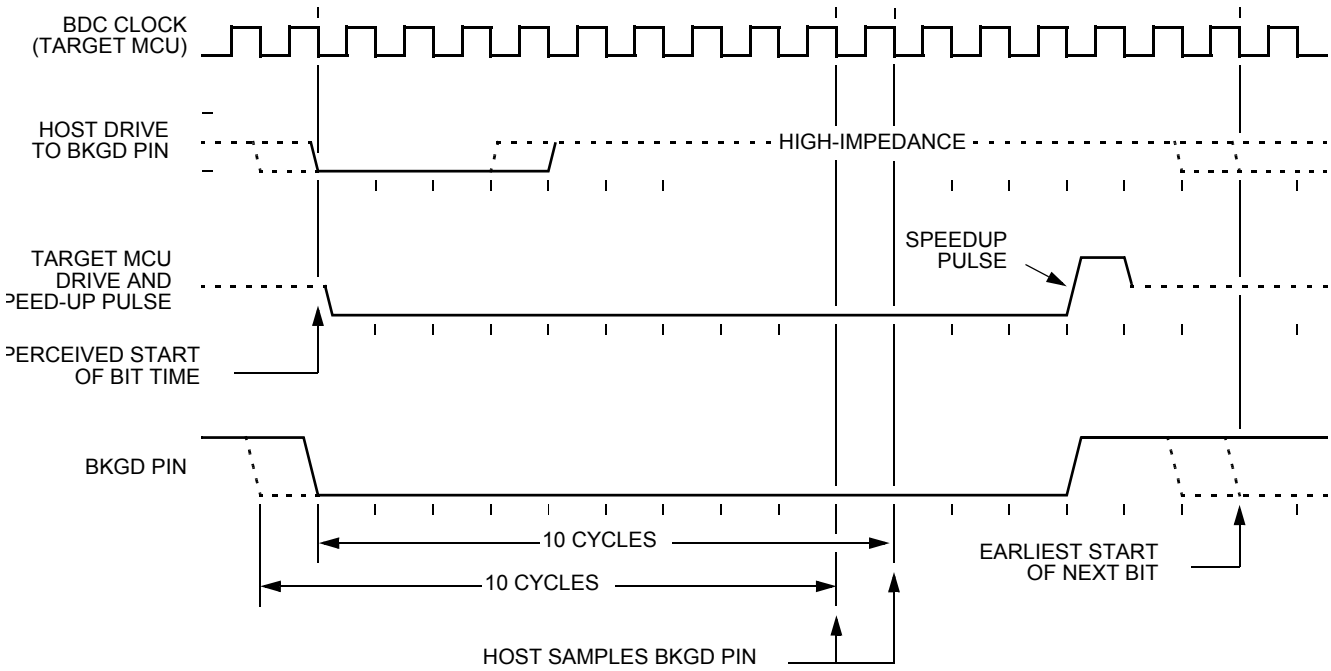


Figure 20-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

20.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target QE32 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 20-1 shows all QE32 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

Coding Structure Nomenclature

This nomenclature is used in Table 20-1 to describe the coding structure of the BDC commands.

	Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)
/	= separates parts of the command
d	= delay 16 target BDC clock cycles
AAAA	= a 16-bit address in the host-to-target direction
RD	= 8 bits of read data in the target-to-host direction
WD	= 8 bits of write data in the host-to-target direction
RD16	= 16 bits of read data in the target-to-host direction
WD16	= 16 bits of write data in the host-to-target direction
SS	= the contents of BDCSCR in the target-to-host direction (STATUS)
CC	= 8 bits of write data for BDCSCR in the host-to-target direction (CONTROL)
RBKP	= 16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register)
WBKP	= 16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register)

Table 20-1. BDC Command Summary

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a ¹	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 (QE32 devices have no external tagging pin)
READ_A	Active BDM	68/d/RD	Read accumulator (A)
READ_CCR	Active BDM	69/d/RD	Read condition code register (CCR)
READ_PC	Active BDM	6B/d/RD16	Read program counter (PC)
READ_HX	Active BDM	6C/d/RD16	Read H and X register pair (H:X)
READ_SP	Active BDM	6F/d/RD16	Read stack pointer (SP)
READ_NEXT	Active BDM	70/d/RD	Increment H:X by one then read memory byte located at H:X
READ_NEXT_WS	Active BDM	71/d/SS/RD	Increment H:X by one then read memory byte located at H:X. Report status and data.
WRITE_A	Active BDM	48/WD/d	Write accumulator (A)
WRITE_CCR	Active BDM	49/WD/d	Write condition code register (CCR)
WRITE_PC	Active BDM	4B/WD16/d	Write program counter (PC)
WRITE_HX	Active BDM	4C/WD16/d	Write H and X register pair (H:X)
WRITE_SP	Active BDM	4F/WD16/d	Write stack pointer (SP)
WRITE_NEXT	Active BDM	50/WD/d	Increment H:X by one, then write memory byte located at H:X
WRITE_NEXT_WS	Active BDM	51/WD/d/SS	Increment H:X by one, then write memory byte located at H:X. Also report status.

¹ 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.

20.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

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.

20.3 On-Chip Debug System (DBG)

Because QE32 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 discussed in greater detail in [Section 20.3.6, "Hardware Breakpoints"](#).

20.3.1 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 allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you 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)

20.3.2 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, you 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 $((8 - \text{CNT}) - 1)$ 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 20.3.5, “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 $\text{ARM} = 0$, 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.

20.3.3 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.

20.3.4 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 DBGTC 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.

20.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTC register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTC register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTC 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 DBGTC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGSR. 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 DBGGEN in DBGTC.

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 modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you 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 modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you 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.

20.3.6 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 20.3.5, “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 (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.

20.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet 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.

20.4.1 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.

20.4.1.1 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.

	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 20-5. BDC Status and Control Register (BDCSCR)

Table 20-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	Enable BDM (Permit Active Background Mode) — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	Background Mode Active Status — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	BDC Breakpoint Enable — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	Force/Tag Select — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	Select Source for BDC Communications Clock — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

Table 20-2. BDCSCR Register Field Descriptions (continued)

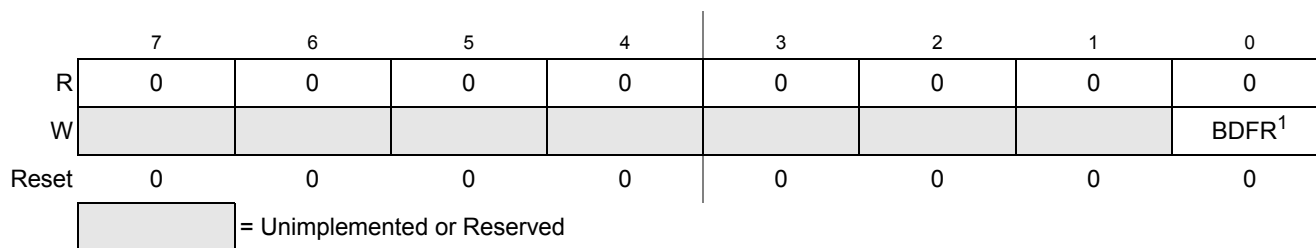
Field	Description
2 WS	<p>Wait or Stop Status — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.</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>Wait or Stop Failure Status — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.)</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>Data Valid Failure Status — This status bit is not used in the 9S08QE32 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>

20.4.1.2 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 20.2.4, “BDC Hardware Breakpoint”](#).

20.4.2 System Background Debug Force Reset Register (SBDFR)

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



¹ BDFR is writable only through serial background mode debug commands, not from user programs.

Figure 20-6. System Background Debug Force Reset Register (SBDFR)

Table 20-3. SBDFR Register Field Description

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

20.4.3 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.

20.4.3.1 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.

20.4.3.2 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.

20.4.3.3 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.

20.4.3.4 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.

20.4.3.5 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.

20.4.3.6 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 isn't 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.

20.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

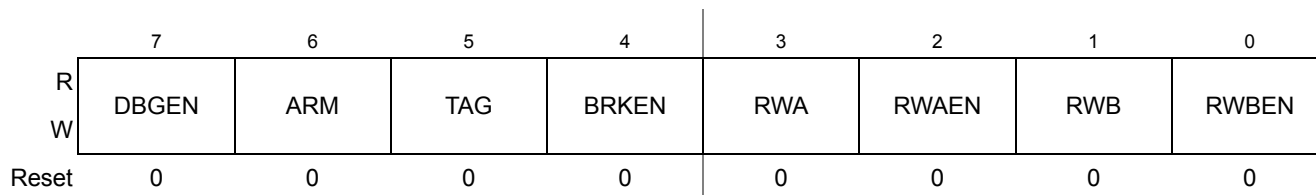


Figure 20-7. Debug Control Register (DBGC)

Table 20-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	Debug Module Enable — 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	Arm Control — 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	Tag/Force Select — 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	Break Enable — 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	R/W Comparison Value for Comparator A — 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	Enable R/W for Comparator A — 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	R/W Comparison Value for Comparator 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	Enable R/W for Comparator 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

20.4.3.8 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.

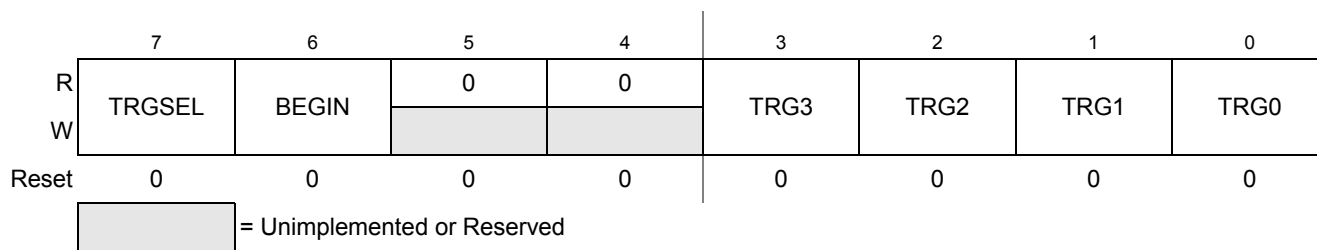


Figure 20-8. Debug Trigger Register (DBGT)

Table 20-5. DBGT Register Field Descriptions

Field	Description
7 TRGSEL	<p>Trigger Type — 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>Begin/End Trigger Select — 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>Select Trigger Mode — 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: $A \leq \text{address} \leq B$ 1000 Outside range: $\text{address} < A$ or $\text{address} > B$ 1001 – 1111 (No trigger)</p>

20.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

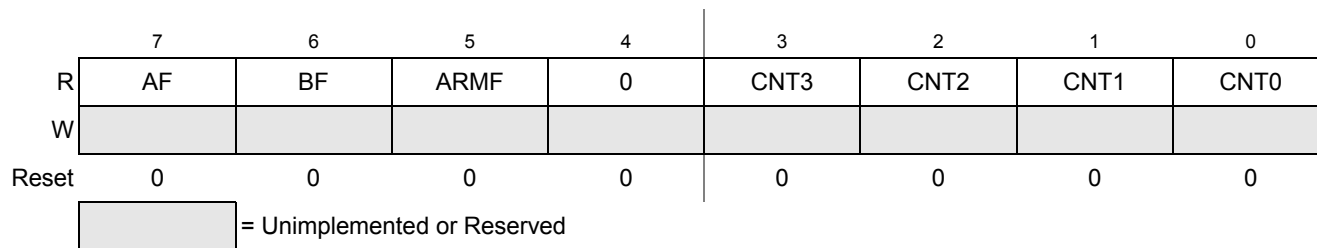


Figure 20-9. Debug Status Register (DBGS)

Table 20-6. DBGS Register Field Descriptions

Field	Description
7 AF	Trigger Match A Flag — 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	Trigger Match B Flag — 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	Arm Flag — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (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 DBGC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	FIFO Valid Count — 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

