# MC13234/MC13237

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

Document Number: MC13237RM
Rev. 1.8
9/2013

**How to Reach Us:**

**Home Page:**
freescale.com

**Web Support:**
freescale.com/support

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

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 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 does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+,Energy Efficient Solutions logo, PowerQUICC, QorIQ, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. CoreNet, Layerscape, QorIQ Qonverge, QUICC Engine, Tower, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

Document Number: MC13237RM
Rev. 1.8
9/2013

# Contents

## About This Book

## Chapter 1
## MC13237 Introduction

## Chapter 2
## Pins and Connections

## Chapter 3
## System Considerations

## Chapter 4
## Memory

## Chapter 5
## System Management and Control

## Chapter 6
## IEEE 802.15.4 Transceiver (including 802.15.4 Module)

## Chapter 7
## Advanced Security Module (ASM)

## Chapter 8
## CPU

## Chapter 9
## Parallel Input and Output

## Chapter 10
## Real-Time Counter (RTC)

## Chapter 11
## Modules (KBIx)

## Chapter 12
## Analog-to-Digital Converter (ADC)

## Chapter 13
## Inter-integrate Circuit (IIC)

## Chapter 14
## Serial Communications Interface (SCI)

## Chapter 15
## Serial Peripheral Interface (SPI) Module

## Chapter 16
## Timer/PWM (TPM Module)

## Chapter 17
## Carrier Modulator Timer (CMT) Module

## Chapter 18
## Development Support

## Appendix A
## IEEE 802.15.4 PHY Messaging Overview

## About This Book

This manual describes Freescale’s fourth-generation ZigBee™ platform (the MC13234/MC13237), which incorporates a low power 2.4 GHz radio frequency transceiver and an 8-bit microcontroller into a single 7x7x1 mm 48-pin LGA package. The MC13234/MC13237 solution can be used for wireless applications from simple proprietary point-to-point connectivity, to a complete ZigBee mesh network. The combination of the radio and a microcontroller in a small footprint package allows for a cost-effective solution.

## Audience

This manual is intended for system designers.

## Organization

This document is organized into 18 chapters and one appendix.

Chapter 1 **Introduction** — Briefly introduces the MC13234/MC13237. The MC13234/MC13237 is Freescale’s fourth-generation ZigBee™ device. It incorporates a low power 2.4 GHz radio frequency transceiver and an 8-bit microcontroller into a single 7x7x1 mm 48-pin LGA package.

Chapter 2 **Pins and Connections** — Describes device pinout and functionality.

Chapter 3 **System Considerations** — Describes system level considerations of the MC13234/MC13237 modem and MCU.

Chapter 4 **Memory** — Describes on-chip memory in the HCS08 series of MCUs and shows that it consists of RAM, flash program memory for non-volatile data storage, plus I/O and control/status registers.

Chapter 5 **System Management** — Describes the various elements that manage and control operation of the HCS08.

Chapter 6 **IEEE 802.15.4 2.4 GHz Transceiver —** Provides an overview and general description for the radio and the modem and details the sequence manager, timer resources, and control functions.

Chapter 7 **Advanced Security Module (ASM)** — Details how the ASM engine encrypts using the Advanced Encryption Standard (AES).

Chapter 8 **Central Processor Unit (CPU)** — Provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family. This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family.

Chapter 9 **Parallel Input/Output** — Explains software controls related to parallel input/output (I/O).This section explains software controls related to parallel input/output (I/O).

Chapter 10 **Real Time Counter (RTC) —** Details the 16-bit counter, a 16-bit comparator, several binary-based and decimal-based prescaler dividers, three clock sources, and a programmable periodic interrupt request.

Chapter 11 **Keyboard Interrupt (KBI)** — Describes the KBI module. Eight keyboard interrupt inputs are shared with port B pins.

Chapter 12 **Analog-to-Digital Converter (ADC) Module** — Describes the ADC module. The 12-bit ADC is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

Chapter 13 **Inter Integrated Circuit (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 14 **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 15 **Serial Peripheral Interface —** This chapter details the serial peripheral interface (SPI).

Chapter 16 **Timer Pulse Width Modulator (PWM**) — Describes how the MC13234/MC13237 uses its internal Event Timer block to manage system timing.

Chapter 17 **Carrier Modulator Timer (CMT)** — The CMT module is an IR LED driver. The module can transmit data to IRO pin either in baseband or in FSK mode.

Chapter 18 **Debug —** Describes the MC13234/MC13237 comprehensive debug/development capability for the HCS08 MCU.

Appendix A **IEEE 802.15.4 PHY Messaging Overview**— Provides a simple overview of the sequence manager.

## Revision History

The following table summarizes revisions to this document since the previous release (Rev 1.7).

**Revision History**

| Location | Revision |
|---|---|
| Chapter 5 | Added a new Section 5.7.6.6, “Stop3 and LPRun Mode Transition” |
| Throughout | Minor typos. |

# 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 |
| FIFO | First In, First Out |
| 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 MC9S08GB/GT60 Data Sheet
3. Freescale MC13237 Data Sheet

# Chapter 1
# MC13237 Introduction

## 1.1 Overview

The MC13234/MC13237 is Freescale's low-cost System-on-Chip (SoC) for the IEEE® 802.15.4 Standard that incorporates a complete, low power, 2.4 GHz radio frequency transceiver with TX/RX switch, an 8-bit HCS08 CPU, and a functional set of MCU peripherals into a 48-pin LGA package. This solution is targeted for wireless RF remote control and other cost-sensitive applications ranging from home TV and entertainment systems such as ZigBee BeeStack Consumer (RF4CE) to low cost, low power, IEEE 802.15.4 and ZigBee end nodes. The MC13234/MC13237 is a highly integrated solution, with very low power consumption.

The MC13234/MC13237 contains an RF transceiver which is an 802.15.4 Standard - 2006 compliant radio that operates in the 2.4 GHz ISM frequency band. The transceiver includes a low noise amplifier, 1 mW nominal output power amplifier (PA), internal voltage controlled oscillator (VCO), integrated transmit/receive switch, on-board power supply regulation, 12-bit Analog-to-Digital Converter (ADC) and full spread-spectrum encoding and decoding.

The on-chip CPU is based on the Freescale HCS08 family of Microcontroller Units (MCU) and has 128 kilobyte (KB) of flash memory and 8 KB of RAM. The onboard MCU peripheral set has been defined to support the targeted applications. A dedicated DMA block transfers packet data between RAM and the transceiver to off-load the CPU and allow higher efficiency and increased performance.

## 1.2 Block Diagram

Figure 1-1 shows a simplified block diagram of the MC13234/MC13237.

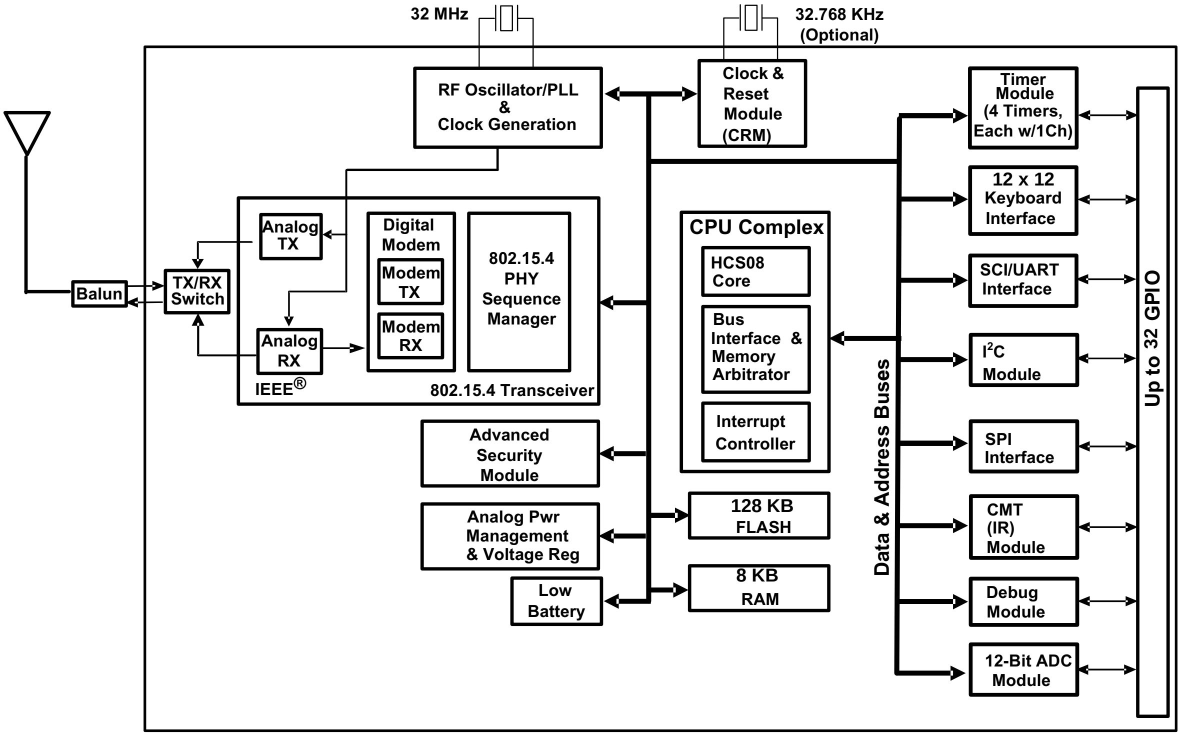


Figure 1-1. MC13234/MC13237 Simplified Block Diagram

## 1.3 Features Summary

- Fully compliant IEEE 802.15.4 Standard 2006 transceiver supports 250 kbps O-QPSK data in 5.0 MHz channels and full spread spectrum encode and decode
  — 2.4 GHz
  — Operates on one of 16 selectable channels per IEEE 802.15.4
  — Programmable output power with 0 dBm nominal output power, programmable from –30 dBm to +2 dBm typical
  — Receive sensitivity of –93 dBm (typical) at 1% PER, 20-byte packet, much better than the IEEE 802.15.4 Standard of –85 dBm
  — Partial Power Down “listen” mode (PPD_RX) available to reduce current while in receive mode and waiting for an incoming frame
- Small RF footprint
  — Integrated transmit/receive switch
  — Differential input/output port (typically used with a balun)
  — Low external component count

- Hardware acceleration for IEEE® 802.15.4 applications
  - DMA interface
  - AES-128 Security module
  - 16-Bit random number generator
  - 802.15.4 Auto-sequence support
  - 802.15.4 Receiver Frame filtering
- 32 MHz crystal reference oscillator; onboard load trim capability supplements external load capacitors
- Onboard 1 kHz oscillator for wake-up timing or an optional 32.768 kHz crystal for accurate low power timing.
- Transceiver Event Timer module has 4 timer comparators available to help manage the auto-sequencer and to supplement MCU TPM resources
- HCS08 8-bit, 32 MHz CPU
- Flash memory
  - $131072_{dec}$ Bytes organized as 128 segments by 1024 bytes
  - Programmable over the full power supply range of 1.8–3.6 V
  - Automated program and erase algorithms
  - Flexible protection scheme to prevent accidental program or erase
  - Security feature to prevent unauthorized access to the Flash
- RAM
  - 8 KBytes RAM
- Powerful In-circuit debug and flash programming available via on-chip module (BDM)
  - Two comparator and 9 trigger modes
  - Eight deep FIFO for storing change-of-flow addresses and event-only data
  - Tag and force breakpoints
  - In-circuit debugging with single breakpoint
- Multiple low power modes (less than 1 µA in Stop3)
- Keyboard interrupt (KBI) module
  - MC13234
    - Two keyboard control modules capable of supporting up to a 12 x 12 keyboard matrix
    - 12 dedicated KBI pins support a 6 x 6 matrix without impacting other IO resources
    - 12 KBI interrupts with selectable polarity
  - MC13237
    - One keyboard control module capable of supporting up to a 8 x 8 keyboard matrix
    - 8 dedicated KBI pins support a 4 x 4 matrix without impacting other IO resources
    - 8 KBI interrupts with selectable polarity
- Serial communication interface (SCI)
  - Full duplex non-return to zero (NRZ)

  - Baud rates as high as 1 Mbps can be supported
  - LIN master extended break generation
  - LIN slave extended break detection
  - Wake-up on active edge
- Serial peripheral interface (SPI)
  - Full-duplex or single-wire bidirectional
  - Double-buffered transmit and receive
  - Master or Slave mode; MSB-first or LSB-first shifting
- Inter-integrated circuit (IIC) interface
  - Up to 100 kbps baud rate with maximum bus loading
  - Baud rates as high as 800 kbps can be programmed
  - Multi-master operation
  - Programmable slave address
  - Interrupt driven byte-by-byte data transfer;
  - Supports broadcast mode and 10-bit addressing
- Four 16-bit timer/pulse width modulators (TMP[4:1]) — each TPM module has an assigned GPIO pin and provides
  - Single channel capability
  - Input capture
  - Output compare
  - Buffered edge-aligned or center-aligned PWM
- 8-Channel, 12-bit resolution ADC
  - 2.5 µs conversion time
  - Internal 1.7 mV/°C temperature sensor
  - Internal bandgap reference
  - Operation in Stop3
  - Fully functional from 1.8 V to 3.6 V
- Carrier Modulator Timer (CMT) — IR Remote carrier generator, modulator, and transmitter.
- Real-time counter (RTC)
  - 16-bit modulus counter with binary or decimal based prescaler;
  - External clock source for precise time base, time-of-day, calendar or task scheduling functions
  - Capable of greater than one day interrupt.
- System protection features
  - Programmable low voltage warning and interrupt (LVI)
  - Optional watchdog timer (COP)
  - Illegal opcode detection
- 1.8 V to 3.6 V operating voltage with on-chip voltage regulators.

- Up to 32 GPIO
  - MC13234: 32 GPIOs
  - MC13237: 28 GPIOs
  - Hysteresis and configurable pull up resistors on all input pins
  - Configurable slew rate and drive strength on all output pins
- –40°C to +85°C temperature range
- RoHS-compliant 7x7 mm 48-pin LGA package

**Table 1-1. MC13234 and MC13237 Comparison**

| Feature | MC13234 | MC13237 |
|---|---|---|
| Radio | IEEE 802.15.4 compliant | |
| CPU | 32 MHz HCS08 | |
| Flash memory | 128K | |
| RAM | 8K | |
| BDM | Yes | |
| Low power modes | Yes | |
| KBI | Two (12 interrupts) | One (8 interrupts) |
| SCI | Yes | |
| SPI | Yes | |
| IIC | Yes | |
| TPM | Yes | |
| CMT | Yes | |
| RTC | Yes | |
| LVD | Yes | |
| COP | Yes | |
| ADC | No | Yes |
| GPIO | 32 | 28 |

## 1.4 Software Solutions

Freescale provides a powerful software environment called the Freescale BeeKit Wireless Connectivity Toolkit. BeeKit is a comprehensive codebase of wireless networking libraries, application templates, and sample applications. The BeeKit Graphical User Interface (GUI), part of the BeeKit Wireless Connectivity Toolkit, allows users to create, modify, and update various wireless networking implementations. A wide range of software functionality is available to complement the MC13234/MC13237 and these are provided as codebases within BeeKit. The following sections describe the available tools.

### 1.4.1 Simple Media Access Controller (SMAC)

The Freescale Simple Media Access Controller (SMAC) is a simple ANSI C based code stack available as sample source code. The SMAC can be used for developing proprietary RF transceiver applications using the MC13234/MC13237.

- Supports point-to-point and star network configurations
- Proprietary networks
- Source code and application examples provided

### 1.4.2 IEEE 802.15.4 2006 Standard-Compliant MAC

The Freescale 802.15.4 Standard-Compliant MAC is a code stack available as object code. The 802.15.4 MAC can be used for developing MC13234/MC13237 networking applications based on the full IEEE® 802.15.4 Standard that use custom Network Layer and application software.

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

### 1.4.3 BeeStack Consumer

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

Some key characteristics of a BeeStack Consumer network are:

- An over the air data rate of 250 kbps in the 2.4 GHz band
- 3 independent communication channels in the 2.4 GHz band
- 2 network node types, controller node and target node
- Channel Agility mechanism
- Provides robustness and ease of use
- Includes essential functionality to build and support a CE network

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

### 1.4.4 ZigBee-Compliant Network Stack

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

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

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

The complete Freescale BeeStack protocol stack includes the following components:

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

### 1.4.5 SynkroRF Platform

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

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

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

## 1.5 Integrated IEEE 802.15.4 Transceiver (Radio and Modem)

The MC13234/MC13237 IEEE 802.15.4 is a fully-compliant transceiver. It provides a complete 2.4 GHz radio with 250 kbps Offset-Quadrature Phase Shift Keying (O-QPSK) data in 5.0 MHz channels with a full spread-spectrum encode and decode. The modem supports the full requirement of the IEEE 802.15.4 Standard functionality to transmit, receive, and do clear channel assessment (CCA), Energy Detect (ED), and Link Quality Indication (LQI).

- Programmable output power with 0 dBm nominal output power, programmable from –30 dBm to +2 dBm typical
- Receive sensitivity of –93 dBm (typical) at 1% PER, 20-byte packet
- Differential bidirectional RF input/output port
- Integrated transmit/receive switch
- Receive current can be reduced while waiting or "listening" for an incoming frame using partial power down (PPD) mode

### 1.5.1 RF Interface and Usage

The MC13234/MC13237 RF interface provides a bidirectional, differential port that connects directly to a balun. The balun connects directly to a single-ended antenna and converts that interface to a full differential, bidirectional, on-chip interface with transmit/receive switch, LNA, and complementary PA outputs.This combination allows for a small footprint and low cost RF solution to be realized.

### 1.5.2 Transceiver Register Interface and Operation

The transceiver is controlled by set of interface registers that are memory-mapped into the CPU address space. The transceiver is capable of independent operation to transmit, receive, or perform CCA/ED operations. Additional features of the transceiver include:

- DMA function moves data directly between RAM and transceiver buffers during transmit and receive on a cycle-steal basis. This off loads the data transfer from the CPU and provides higher performance, and can also allow lower power during TX/RX.
- Interrupt capability dependent on RX packet data availability. An interrupt can be generated based on a programmed count of RX data bytes that have been received and moved to RAM. This allows CPU filtering of RX data before completion of the packet reception to accelerate response to the packet.
- Four (4) transceiver Event Timer comparators are available to supplement MCU peripheral timer resources for PHY and MAC timing requirements.

### 1.5.3 IEEE 802.15.4 Acceleration Hardware

The MC13234/MC13237 802.15.4 transceiver has several hardware features that reduce the software stack size, off load the function from the CPU, and improve performance. A list of features supported is provided below:

- 2003 & 2006 versions of the IEEE® 802.15.4 standard is full supported
- Slotted and unslotted modes
- Beacon enabled and non-beacon enabled networks
- DMA data transfer between RAM and radio
- Separate AES-128 Security Module
- 16-bit Random Number Generator
- 802.15.4 Sequence support
  - RX (conditionally followed by TXAck)
  - TX
  - CCA (used for CCA and ED cycles)
  - TX/RX (TX followed by unconditional RX or RCACK)
  - Continuous CCA
- 802.15.4 Receiver Frame filtering.

### 1.5.4 Unique Partial Power Down (PPD_RX) or “Listen” Receive Mode

The MC13234/MC13237 provides a unique Partial Power Down receive (PPD_RX) mode. A summary of PPD_RX mode of operation when selected is described below:

- Whenever a receive cycle is initiated, the receiver is not turned fully on to save current until receive energy of a preset level is detected

- The receiver will turn fully on only when triggered by energy at a pre-determined preset level thus enabling reception of the expected frame. Afterwards, the receiver will begin operating in the full-on state that is considered to be the same as the standard receive state
- The preset level can be programmed for various RX input power levels

Use of the PPD_RX mode provides two distinct advantages:

- Reduced Listen mode current — The receive current is significantly reduced while waiting for a frame. If a node is a coordinator, router, or gateway and it spends a significant percentage of its RF-active time waiting for incoming frames from clients or other devices, the net power savings can be significant.
- Reduced sensitivity as a desired effect — The PPD_RX mode provides different levels of reduced sensitivity. If a node operates in a densely populated area, it may be desirable to de-sensitize the receiver such that the device does not respond to incoming frames with an energy level below the desired threshold. This could be useful for security, net efficiency, reduced noise triggering and many other purposes.

## 1.6 HCS08 8-Bit Central Processing Unit (CPU)

The onboard CPU is a 32 MHz 8-bit HCS08 CPU. It executes the HC08 instruction set with added BGND instruction and supports for up to 32 interrupt/reset sources. The HCS08 CPU is fully source and object code compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes are added to improve C compiler efficiency and to support a new background debug system. It has an 8-bit data bus, a 16-bit address bus and a 2-stage instruction pipe that facilitates the overlapping of instruction fetching and execution. There are 29 vectors for internal interrupt sources and one vector for an external interrupt pin. The debug or BDM module provides a serial one-wire interface for non-intrusive debugging of application programs.

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- 64-KB CPU address space with banked memory management unit for greater than 64 KB
- 16-bit stack pointer (any size stack anywhere in 64-KB CPU address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
  - Inherent — Operands in internal registers
  - Relative — 8-bit signed offset to branch destination
  - Immediate — Operand in next object code byte(s)
  - Direct — Operand in memory at 0x0000–0x00FF
  - Extended — Operand anywhere in 64-KB 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

## 1.7 System Clocks

The primary system reference frequency is a 32 MHz crystal oscillator. The crystal requirements for the oscillator and oscillator performance must support a +/-40 ppm frequency accuracy to meet the IEEE 802.15.4 Standard requirements. All system clocks are generated from this source. Features of the clock system include:

- The 32 MHz reference oscillator has onboard programmable capacitive loading that allows software tuning of frequency accuracy
- CPU clock as high as 32 MHz
- Bus clock (and peripheral clock) equals 1/2 CPU clock
- Clocks to individual peripherals can be independently disabled for best power management.
- CPU clock can be lowered to 500 kHz for lower power (250 kHz bus clock)

An optional 32.768 kHz crystal oscillator is available for accurate low power timing and the Real Time Clock (RTC). Also, an onboard, low accuracy 1 kHz oscillator is available for sleep timing wake-up.

## 1.8 Memory

The MC13234/MC13237 memory resources consist of RAM, flash program memory for nonvolatile data storage, and control/status registers for I/O, peripherals, management, and the transceiver. Features include:

- 128 Kbytes flash
- 8 Kbytes RAM
- Security circuitry to prevent unauthorized access to RAM and flash contents

## 1.9 System and Power Management

The MC13234/MC13237 is inherently a low power device, but it also has extensive system control and power management to maximize battery life and provide system protection.

### 1.9.1 Modes of Operation

The MC13234/MC13237 modes of operation include:

- 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 clock is set to 500 kHz and peripheral clocks (bus clock) to 250 kHz and the internal voltage regulator is in standby
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained
- LPWait mode — CPU shuts down to conserve power; peripheral clocks are restricted to 250 kHz and the internal voltage regulator is in standby
- Stop3 mode — System clocks are stopped and voltage regulator is in standby. All internal circuits are powered for fast recovery (32 MHz oscillator on-off optional)

### 1.9.2 Power Management

The MC13234/MC13237 power management is controlled through programming of the modes of operation. Different modes allow for different levels of power-down. Additional features include:

- The transceiver is powered as required
- The analog radio is only powered-up as required to do a TX, RX, or CCA/ED operation
- Peripheral control clock gating can be disabled on a module-by-module basis to provide lowest power
- Programmed mode manages
  - Degree of chip power down
  - Retention of programmed parameters
  - Clock management
- Power-down and wake-up (clocks and analog blocks) are gracefully controlled
- RTC can be used as wake-up timer
- Wake-up available through KBI and UART RX asynchronous interrupts
- Real-time counter (RTC) module
  - 16-bit modulus counter with binary or decimal based prescaler for precise time base, time-of-day, calendar or task scheduling functions.
  - Capable of greater than one day interrupt.
  - Can also be used for device wake-up.

### 1.9.3 System Protection

The MC13234/MC13237 provides several vehicles to maintain security or a high level of system robustness:

- Watchdog computer operating properly (COP) reset with option to run from dedicated internal clock source or bus clock
- Low-voltage warning and detection with reset or interrupt; selectable trip points
- Illegal opcode detection with reset
- Flash block protection

## 1.10 MCU Peripherals

The MC13234/MC13237 has a functional set of MCU peripherals focused for intended applications:

- Keyboard interrupt module (KBI) — MC13234 has two (2) KBI modules; KBI1 shares eight (8) port B pins and KBI2 shares four (4) port C pins. MC13237 has one (1) KBI module; KBI1 shares eight (8) port B pins. Any KBI pin can be enabled as a keyboard input that can act as an interrupt request. As a result, the total 12 KBI inputs (MC13234) allows as large as a 12x12 keyboard matrix. The total 8 KBI inputs (MC13237) allows as large as a 8x8 keyboard matrix with use of other GPIO pins as outputs to the matrix.
- Serial communication interface (SCI) / UART module
- Serial peripheral interface (SPI) module — Fully programmable for full-duplex or single-wire bidirectional operation, as well as, master or slave mode
- Inter-integrated circuit (IIC) interface — With up to 100 kbps bit-rate
- Four (4) timer/pulse width modulators (TMP[4:1]) — With input capture, output compare, and PWM modes.
- Carrier Modulator Timer (CMT) module — IR remote carrier generator, modulator, and transmitter.
- Real-time counter (RTC) module — 16-bit modulus counter with binary or decimal based prescaler for precise time base, time-of-day, calendar or task scheduling functions. Capable of greater than one day interrupt. Can also be used for device wake-up.
- Up to 32 GPIO — With hysteresis and a pullup device that can be configured on all input pins and slew rate and drive strength that can be configured on all output pins.

## 1.11 12-Bit Analog-to-Digital Conversion (ADC) Module

The MC13237 integrates a multi-channel (12 channel), 12-bit resolution Successive Approximation Register (SAR) analog-to-digital conversion (ADC) module. (The ADC module is not available on the MC13234.) The analog input channels are shared/multiplexed with standard GPIO pins.

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bits resolution
- Operational in Stop3 mode
- 2.5 µs conversion time
- Internal bandgap reference
- Operation over full VBATT voltage range
- Internal 1.7 mV/°C temperature sensor
- Output data can be formatted in 8-, 10-, or 12-bit justified format
- Single or continuous conversion
- Configurable sample time and conversion speed / power.
- Auto compare for less-than, greater than, or equal to programmable value
- Converter subsystem shut-down
- Real-time clock conversion trigger

## 1.12 Development Environment

Development support for the HCS08 on the MC13234/MC13237 includes the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the MCU that provides a convenient interface for programming the on-chip flash and other storage. 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.

Address and data bus signals are not available on external pins. Debug is done through commands fed into the 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. Features include:

- Single-wire background debug interface
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging (plus two more breakpoints in on-chip debug module)
- On-chip in-circuit emulator (ICE) debug module containing three comparators and nine trigger modes.
- Eight deep FIFO for storing change-of-flow addresses and event-only data. Debug module supports both tag and force breakpoints.

# Chapter 2
# Pins and Connections

## 2.1 Device Pin Assignment

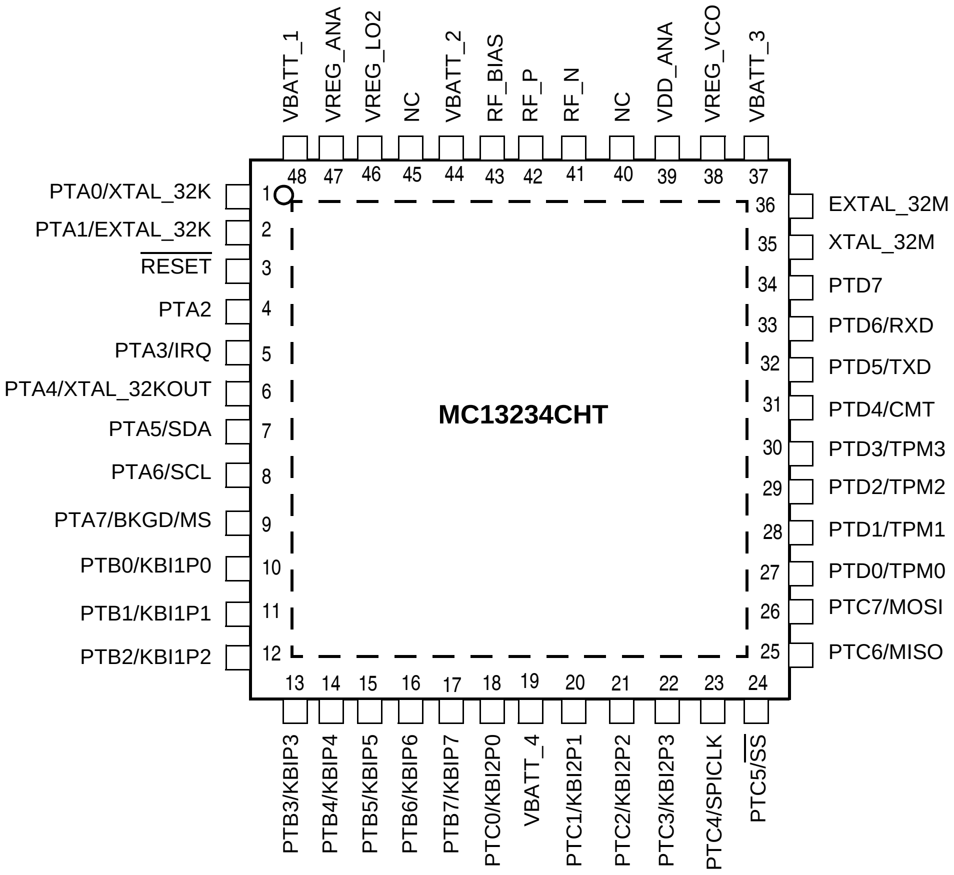


**Figure 2-1. MC13234 Pinout**

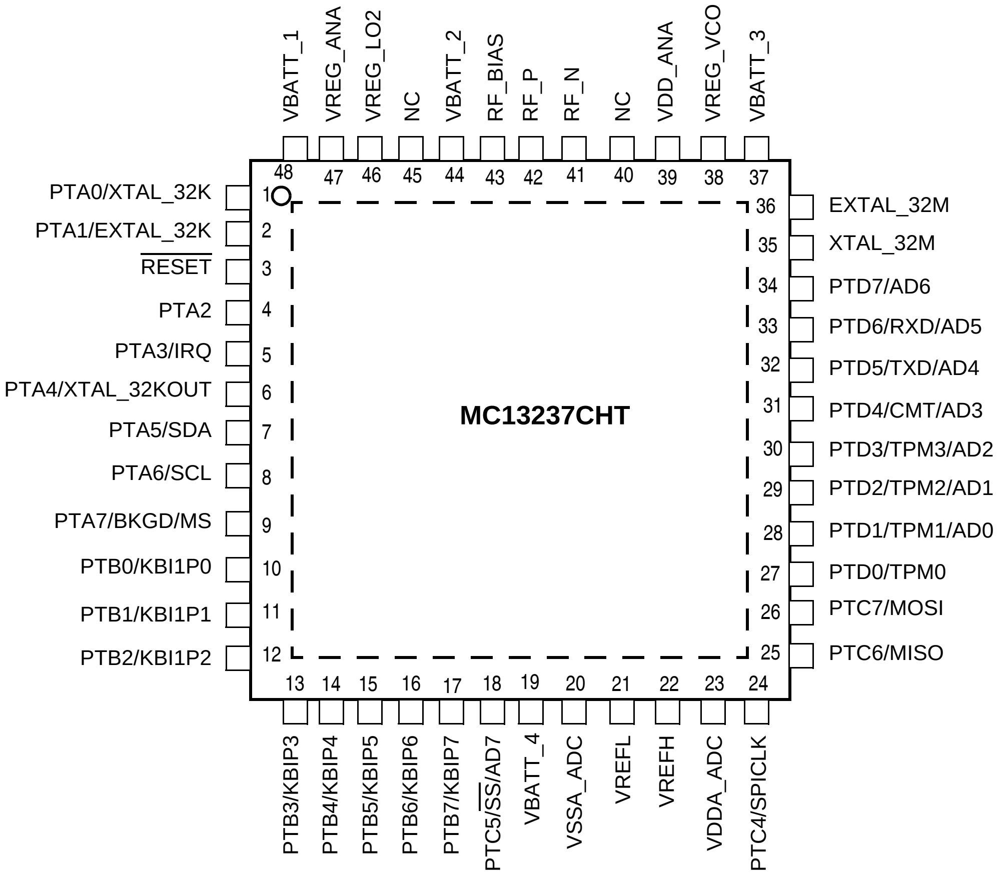


**Figure 2-2. MC13237 Pinout**

## 2.2 Pin Definitions

Table 2-1 details the MC13234 pinout and functionality.

**Table 2-1. MC13234 Pin Function Description**

| Pin # | Pin Name | Type | Description | Functionality |
|---|---|---|---|---|
| 1 | PTA0/XTAL_32K | Digital Input/Output | Port A Bit 0 / 32.768 kHz oscillator output | |
| 2 | PTA1/EXTAL_32K | Digital Input/Output | Port A Bit 1 / 32.768 kHz oscillator input | For normal use, 10 kOhm resistor to ground recommended |
| 3 | $\overline{\text{RESET}}$ | Digital Input/Output | Device asynchronous hardware reset. Active low. Onboard Pullup | Normally input; gets driven low for a period after a reset |
| 4 | PTA2 | Digital Output | Port A Bit 2 / Test Mode enable. | TM mode input. Must be biased low exiting POR for normal operation |
| 5 | PTA3/IRQ | Digital Input/Output | Port A Bit 3 / IRQ. | |
| 6 | PTA4/ XTAL_32KOUT | Digital Input/Output | Port A Bit 4 / Buffered 32.768 kHz clock output | Optional 32.768 kHz output clock for measuring 32 kHz oscillator accuracy (ppm) |
| 7 | PTA5/SDA | Digital Input/Output | Port A Bit 5 / IIC Bus data | Defaults to open drain for IIC |
| 8 | PTA6/SCL | Digital Input/Output | Port A Bit 6 / IIC Bus clock | Defaults to open drain for IIC |
| 9 | PTA7/BKGD/MS | Digital Input/Output | Port A Bit 7 / Background / Mode Select | Debug signal |
| 10 | PTB0/KBI1P0 | Digital Input/Output | Port B Bit 0 / KBI1 Input Bit 0 | Wake-up capability |
| 11 | PTB1/KBI1P1 | Digital Input/Output | Port B Bit 1 / KBI1 Input Bit 1 | Wake-up capability |
| 12 | PTB2/KBI1P2 | Digital Input/Output | Port B Bit 2 / KBI1 Input Bit 2 | Wake-up capability |
| 13 | PTB3/KBI1P3 | Digital Input/Output | Port B Bit 3 / KBI1 Input Bit 3 | Wake-up capability |
| 14 | PTB4/KBI1P4 | Digital Input/Output | Port B Bit 4 / KBI1 Input Bit 4 | Wake-up capability |
| 15 | PTB5/KBI1P5 | Digital Input/Output | Port B Bit 5 / KBI1 Input Bit 5 | Wake-up capability |
| 16 | PTB6/KBI1P6 | Digital Input/Output | Port B Bit 6 / KBI1 Input Bit 6 | Wake-up capability |
| 17 | PTB7/KBI1P7 | Digital Input/Output | Port B Bit 7 / KBI1 Input Bit 7 | Wake-up capability |
| 18 | PTC0/KBI2P0 | Digital Input/Output | Port C Bit 0 / KBI2 Input Bit 0 | |
| 19 | VBATT_4 | Power Input | VDD supply input [1] | Connect to system VDD supply |
| 20 | PTC1/KBI2P1 | Digital Input/Output | Pot C Bit 1 / KBI2 Input Bit 1 | |
| 21 | PTC2/KBI2P2 | Digital Input/Output | Pot C Bit 2 / KBI2 Input Bit 2 | |
| 22 | PTC3/KBI2P3 | Digital Input/Output | Pot C Bit 3 / KBI2 Input Bit 3 | |
| 23 | PTC4/SPICLK | Digital Input/Output | Port C Bit 4 / SPI clock | |
| 24 | PTC5/$\overline{\text{SS}}$ | Digital Input/Output | Port C Bit 5 / SPI slave select | |
| 25 | PTC6/MISO | Digital Input/Output | Port C Bit 6 / SPI MISO | |
| 26 | PTC7/MOSI | Digital Input/Output | Port C Bit 7 / SPI MOSI | |
| 27 | PTD0/TPM0 | Digital Input/Output | Port D Bit 0 / TPM0 signal | TPM0 timer output / gate input signal |

Table 2-1. MC13234 Pin Function Description (continued)

| Pin # | Pin Name | Type | Description | Functionality |
|---|---|---|---|---|
| 28 | PTD1/TPM1 | Digital Input/Output | Port D Bit 1/ TPM1 signal | TPM1 timer output / gate input signal. |
| 29 | PTD2/TPM2 | Digital Input/Output | Port D Bit 2 / TPM2 signal | TPM2 timer output / gate input signal. |
| 30 | PTD3/TPM3 | Digital Input/Output | Port D Bit 3 / TPM3 signal | TPM3 timer output / gate input signal. |
| 31 | PTD4/CMT | Digital Input/Output | Port D Bit 4 / CMT output | Hi drive output for IR diode. |
| 32 | PTD5/TXD | Digital Input/Output | Port D Bit 5 / UART TXD output | UART has no hardware flow control. |
| 33 | PTD6/RXD | Digital Input/Output | Port D Bit 6 / UART RXD input / AD5 signal | UART has no hardware flow control. |
| 34 | PTD7 | Digital Input/Output | Port D Bit 7 | |
| 35 | XTAL_32M | Analog Output | 32 MHz reference oscillator output | |
| 36 | EXTAL_32M | Analog input | 32 MHz reference oscillator input | |
| 37 | VBATT_3 | Power Input | VDD supply input[1] | Connect to system VDD supply |
| 38 | VREG_VCO | VCO Reg Out / in | VCO regulator output and input to VCO 1.8 Vdc VDD | Bypass to ground with 220 nF capacitor. |
| 39 | VDD_ANA | Analog Power Input | Analog 1.8 Vdc Input | Connect to VREG_ANA |
| 40 | NC | | No connection to device | May be left open or connect to ground |
| 41 | RF_N | RF Input/Output | Modem RF input/output negative | Bi-directional RF port for the internal LNA and PA |
| 42 | RF_P | RF Input/Output | Modem RF input/output positive | Bi-directional RF port for the internal LNA and PA |
| 43 | RF_BIAS | RF Voltage Output | Switched RF bias voltage (1.8 Vdc) | High for TX; low for RX |
| 44 | VBATT_2 | Power Input | VDD supply input[1] | Connect to system VDD supply |
| 45 | NC | | | May be left open or connect to ground |
| 46 | VREG_LO2 | LO2 Reg Out | LO2 regulator output @ 1.8 Vdc | Bypass to ground with 220 nF capacitor. |
| 47 | VREG_ANA | ANA Reg Out | Analog regulator output @ 1.8 Vdc | Bypass to ground with 220 nF capacitor. Connect to VDD_ANA |
| 48 | VBATT_1 | Power Input | VDD supply to Analog regulator[1] | Connect to system VDD supply |
| Flag | GND | Power Input | System ground | |

[1] VBATT_1, VBATT_2, VBATT_3 and VBATT_4 signals are not connected onboard MC13234/MC13237.

Table 2-2 details the MC13237 pinout and functionality.

**Table 2-2. MC13237 Pin Function Description**

| Pin # | Pin Name | Type | Description | Functionality |
|---|---|---|---|---|
| 1 | PTA0/XTAL_32K | Digital Input/Output | Port A Bit 0 / 32.768 kHz oscillator output | |
| 2 | PTA1/EXTAL_32K | Digital Input/Output | Port A Bit 1 / 32.768 kHz oscillator input | For normal use, 10 kOhm resistor to ground recommended |
| 3 | $\overline{\text{RESET}}$ | Digital Input/Output | Device asynchronous hardware reset. Active low. Onboard Pullup | Normally input; gets driven low for a period after a reset |
| 4 | PTA2 | Digital Output | Port A Bit 2 / Test Mode enable. | TM mode input. Must be biased low exiting POR for normal operation |
| 5 | PTA3/IRQ | Digital Input/Output | Port A Bit 3 / IRQ. | |
| 6 | PTA4/ XTAL_32KOUT | Digital Input/Output | Port A Bit 4 / Buffered 32.768 kHz clock output | Optional 32.768 kHz output clock for measuring 32 kHz oscillator accuracy (ppm) |
| 7 | PTA5/SDA | Digital Input/Output | Port A Bit 5 / IIC Bus data | Defaults to open drain for IIC |
| 8 | PTA6/SCL | Digital Input/Output | Port A Bit 6 / IIC Bus clock | Defaults to open drain for IIC |
| 9 | PTA7/BKGD/MS | Digital Input/Output | Port A Bit 7 / Background / Mode Select | Debug signal |
| 10 | PTB0/KBI1P0 | Digital Input/Output | Port B Bit 0 / KBI1 Input Bit 0 | Wake-up capability |
| 11 | PTB1/KBI1P1 | Digital Input/Output | Port B Bit 1 / KBI1 Input Bit 1 | Wake-up capability |
| 12 | PTB2/KBI1P2 | Digital Input/Output | Port B Bit 2 / KBI1 Input Bit 2 | Wake-up capability |
| 13 | PTB3/KBI1P3 | Digital Input/Output | Port B Bit 3 / KBI1 Input Bit 3 | Wake-up capability |
| 14 | PTB4/KBI1P4 | Digital Input/Output | Port B Bit 4 / KBI1 Input Bit 4 | Wake-up capability |
| 15 | PTB5/KBI1P5 | Digital Input/Output | Port B Bit 5 / KBI1 Input Bit 5 | Wake-up capability |
| 16 | PTB6/KBI1P6 | Digital Input/Output | Port B Bit 6 / KBI1 Input Bit 6 | Wake-up capability |
| 17 | PTB7/KBI1P7 | Digital Input/Output | Port B Bit 7 / KBI1 Input Bit 7 | Wake-up capability |
| 18 | PTC5/$\overline{\text{SS}}$/AD7 | Digital Input/Output | Port C Bit 5 / SPI Slave Select / AD7 Signal | |
| 19 | VBATT_4 | Power Input | VDD supply input [1] | Connect to system VDD supply |
| 20 | VSSA_ADC | Digital Input/Output | ADC analog ground | |
| 21 | VREFL | Digital Input/Output | ADC low reference voltage | |
| 22 | VREFH | Digital Input/Output | ADC high reference voltage | |
| 23 | VDDA_ADC | Digital Input/Output | ADC analog power supply | |
| 24 | PTC4/SPICLK | Digital Input/Output | Port C Bit 4 / SPI Clock | |
| 25 | PTC6/MISO | Digital Input/Output | Port C Bit 6 / SPI MISO | |
| 26 | PTC7/MOSI | Digital Input/Output | Port C Bit 7 / SPI MOSI | |
| 27 | PTD0/TPM0 | Digital Input/Output | Port D Bit 0 / TPM0 signal | TPM0 timer output / gate input signal |

Table 2-2. MC13237 Pin Function Description (continued)

| Pin # | Pin Name | Type | Description | Functionality |
|---|---|---|---|---|
| 28 | PTD1/TPM1/AD0 | Digital Input/Output | Port D Bit 1/ TPM1 signal /AD0 signal | TPM1 timer output / gate input signal. ADC input 0 |
| 29 | PTD2/TPM2/AD1 | Digital Input/Output | Port D Bit 2 / TPM2 signal /AD1 signal | TPM2 timer output / gate input signal. ADC input 1 |
| 30 | PTD3/TPM3/AD2 | Digital Input/Output | Port D Bit 3 / TPM3 signal /AD2 signal | TPM3 timer output / gate input signal. ADC input 2 |
| 31 | PTD4/CMT/AD3 | Digital Input/Output | Port D Bit 4/ CMT output / AD3 signal | Hi drive output for IR diode. ADC input 3 |
| 32 | PTD5/TXD/AD4 | Digital Input/Output | Port D Bit 5 / UART TXD output / AD4 signal | UART has no hardware flow control. ADC input 4 |
| 33 | PTD6/RXD/AD5 | Digital Input/Output | Port D Bit 6 / UART RXD input / AD5 signal | UART has no hardware flow control. ADC input 5 |
| 34 | PTD7/AD6 | Digital Input/Output | Port D Bit 7 / AD6 signal | ADC input 6 |
| 35 | XTAL_32M | Analog Output | 32 MHz reference oscillator output | |
| 36 | EXTAL_32M | Analog input | 32 MHz reference oscillator input | |
| 37 | VBATT_3 | Power Input | VDD supply input[1] | Connect to system VDD supply |
| 38 | VREG_VCO | VCO Reg Out / in | VCO regulator output and input to VCO 1.8 Vdc VDD | Bypass to ground with 220 nF capacitor. |
| 39 | VDD_ANA | Analog Power Input | Analog 1.8 Vdc Input | Connect to VREG_ANA |
| 40 | NC | | No connection to device | May be left open or connect to ground |
| 41 | RF_N | RF Input/Output | Modem RF input/output negative | Bi-directional RF port for the internal LNA and PA |
| 42 | RF_P | RF Input/Output | Modem RF input/output positive | Bi-directional RF port for the internal LNA and PA |
| 43 | RF_BIAS | RF Voltage Output | Switched RF bias voltage (1.8 Vdc) | High for TX; low for RX |
| 44 | VBATT_2 | Power Input | VDD supply input[1] | Connect to system VDD supply |
| 45 | NC | Input | No connection to device | May be left open or connect to ground |
| 46 | VREG_LO2 | LO2 Reg Out | LO2 regulator output @ 1.8 Vdc | Bypass to ground with 220 nF capacitor. |
| 47 | VREG_ANA | ANA Reg Out | Analog regulator output @ 1.8 Vdc | Bypass to ground with 220 nF capacitor. Connect to VDD_ANA |
| 48 | VBATT_1 | Power Input | VDD supply to Analog regulator[1] | Connect to system VDD supply |
| Flag | GND | Power Input | System ground | |

[1] VBATT_1, VBATT_2, VBATT_3 and VBATT_4 signals are not connected onboard MC13234/MC13237.

# Chapter 3
# System Considerations

## 3.1 Introduction

This chapter presents information on application and operation of the MC13234/MC13237 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 MC13234/MC13237 device and the following chapters present individual functions in detailed descriptions.

## 3.2 Power Connections

The MC13234/MC13237 power connections are listed in Table 3-1.

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

| Pin # | Pin Name | Type | Description | Functionality |
|---|---|---|---|---|
| 19 | VBATT_4 | Power Input | VDD supply input [1] | Connect to system VDD supply |
| 20 | VSSA_ADC[2] | Digital Input/Output | ADC analog ground | ADC ground |
| 23 | VDDA_ADC[2] | Power Input | ADC analog power supply | Connect to system VDD supply or dedicated supply for ADC block |
| 37 | VBATT_3 | Power Input | VDD supply input[1] | Connect to system VDD supply |
| 38 | VREG_VCO | VCO Reg Out / In | VCO regulator output and input to VCO 1.8 Vdc VDD | Bypass to ground with 220 nF capacitor. |
| 39 | VDD_ANA | Analog Power Input | Analog 1.8 Vdc Input | Connect to VREG_ANA |
| 43 | RF_BIAS | RF Voltage Output | Switched RF bias voltage (1.8 Vdc) | High for TX; low for RX |
| 44 | VBATT_2 | Power Input | VDD supply input[1] | Connect to system VDD supply |
| 46 | VREG_LO2 | LO2 Reg Out | LO2 regulator output @ 1.8 Vdc | Bypass to ground with 220 nF capacitor. |
| 47 | VREG_ANA | ANA Reg Out | Analog regulator output @ 1.8 Vdc | Bypass to ground with 220 nF capacitor. Connect to VDD_ANA |
| 48 | VBATT_1 | Power Input | VDD supply to Analog regulator[1] | Connect to system VDD supply |
| Flag | GND | Power Input | System ground | |

[1] VBATT_1, VBATT_2, VBATT_3 and VBATT_4 signals are not connected onboard MC13237.

[2] VSSA_ADC, VDDA_ADC are available only in the MC13237.

When designing power to the MC13234/MC13237 LGA, the following points need to be considered:

- The LGA package has a single common ground flag (VSS).
- The device has four primary power inputs that include VBATT_1, VBATT_2, VBATT_3 and VBATT_4 signals
  - — These are not connected onboard.
  - — These must be connected to a common source supply of 1.8–3.6 Vdc.

### NOTE

> Although the MC13234/MC13237 can operate at VDD supply as low as 1.8 Vdc, it is not recommended to be a regulated supply at 1.8 Vdc. The low-voltage detect (LVD) reset is enabled out of POR and the power supply voltage must rise and exceed the LVD rising-edge threshold or the device will be held in reset. After the rising edge threshold is exceeded, the reset is released.

- VREG_ANA must be tied to VDD_ANA and bypassed with a 220 nF capacitor.
- VREG_LO2 and VREG_VCO must each be bypassed by a 220 nF capacitor.
- The LGA FLAG is the device ground connection and must be tied to supply GND.
- RF_BIAS is a bias voltage for the RF balun that switches between 1.8 Vdc for transmit and ground for receive.

Power supply connections are shown in Figure 3-1.

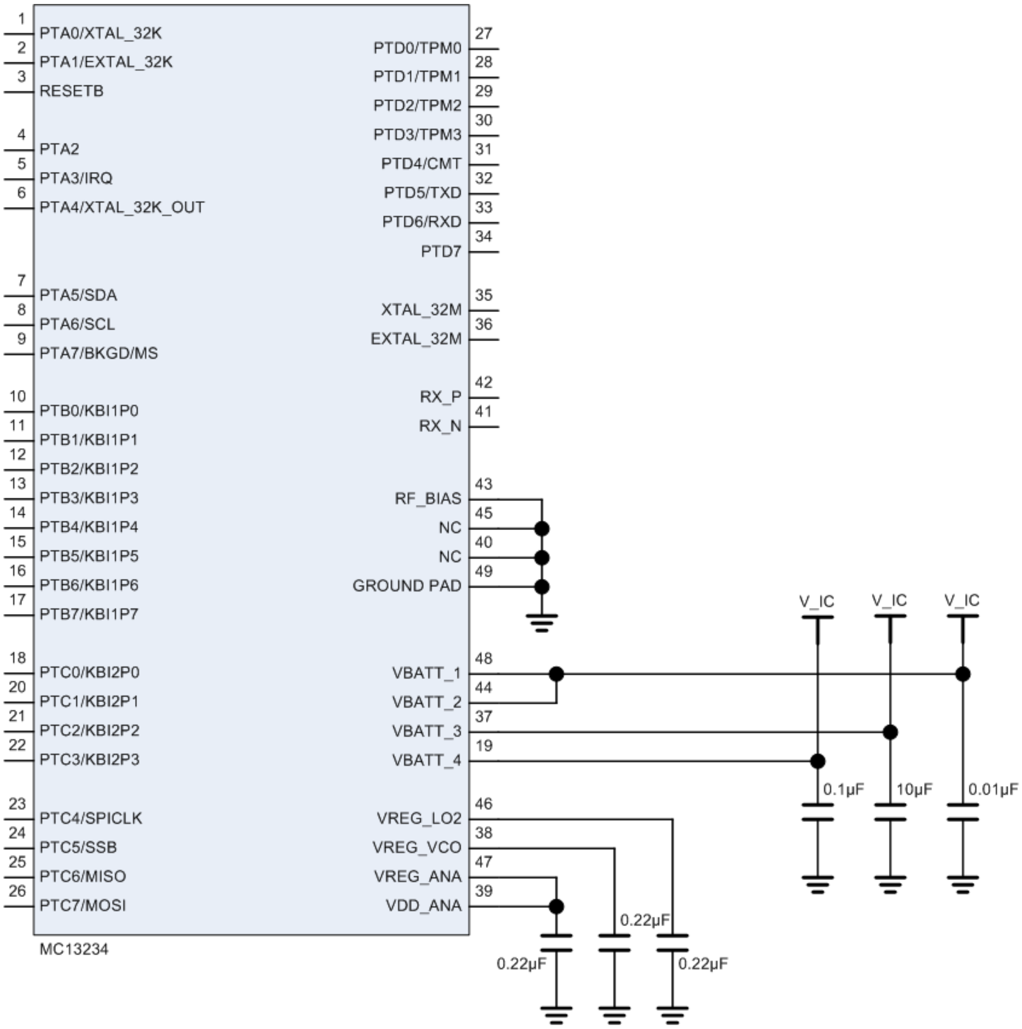


Figure 3-1. MC13234 Power Supply Connections

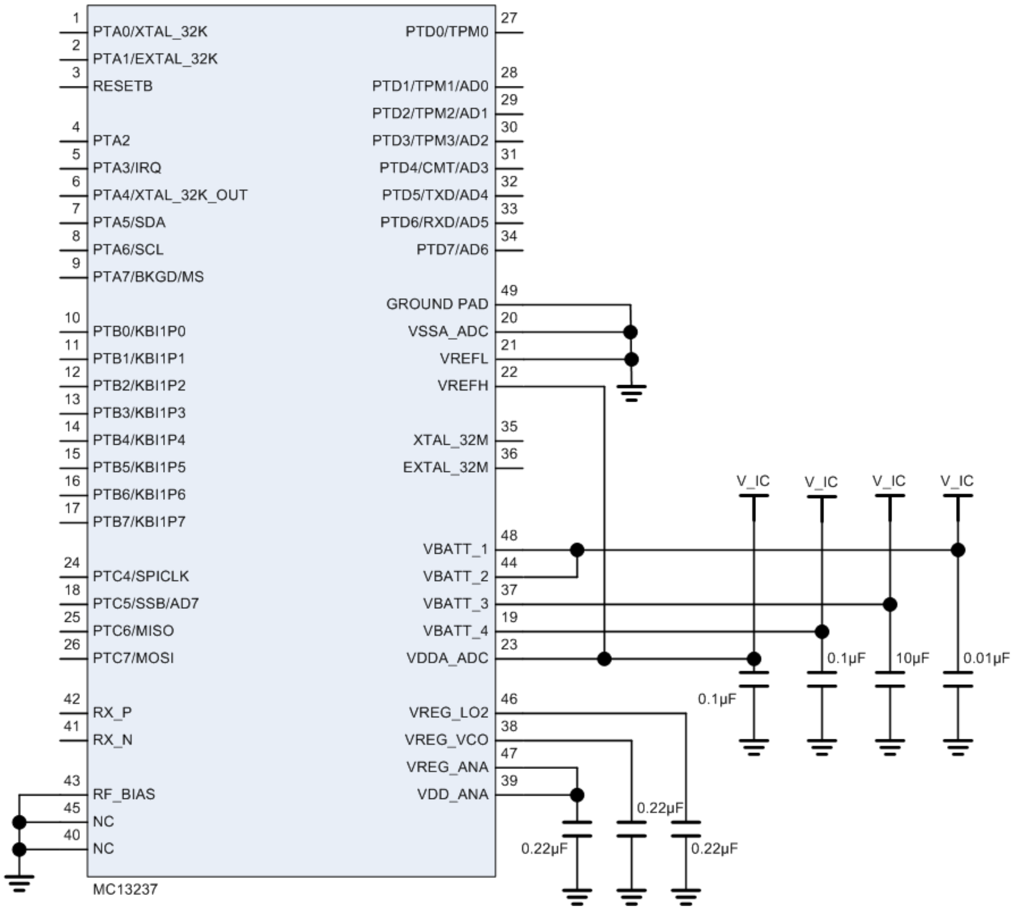


**Figure 3-2. MC13237 Power Supply Connections**

## NOTE

The 220 nF bypass capacitors are required by onboard series voltage regulators

## 3.3 Special Pin Usage

The following signal pins have special usage.

### 3.3.1 Hardware $\overline{\text{RESET}}$ Pin

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

A system reset can be initiated externally or internally:

- External reset — the $\overline{\text{RESET}}$ pin can be driven low from an external switch to ground or driven from a device output. If $\overline{\text{RESET}}$ is driven from an external IO (such as a master MCU), the IO should be open drain or buffered in such a manner as to not drive $\overline{\text{RESET}}$ active high at the same time that the onboard circuitry is driving the pin active low during an internally generated reset. A Schottky diode in series between a standard active high output and the $\overline{\text{RESET}}$ input can be used to block the output from driving high.

**NOTE**

> If $\overline{\text{RESET}}$ is held active for more than 69 bus clock cycles, the reference oscillator will be disabled. See Section 5.4.1, “Clock Sources”.

- Internal reset — if reset is initiated internally the $\overline{\text{RESET}}$ pin is driven low for about 34 cycles of the default bus clock (16 MHz); this equates to approximately 2 microseconds. The output low condition can be used to reset any external system components. As stated previously, an external driver must not hold the signal high to prevent this internally initiated low condition. The onboard output drive is open drain and relies on the pullup to pull the $\overline{\text{RESET}}$ pin high again. The reset pin is sampled approximately 38 cycles later and expects reset to return high. The reset circuitry decodes the onboard cause of reset and records it by setting a corresponding bit in the system control reset status register (SRS).

**NOTE**

> A reset initiated by the $\overline{\text{RESET}}$ pin or any other operation condition does not cycle the POR circuit. The POR reset is only cycled by transitions on the VDD supply.

Figure 3-3 illustrates $\overline{\text{RESET}}$ signal timing. If reset is driven externally, the active low time is set by the source and can be held indefinitely. Upon release, the rise time should be the RC time constant as determined by the total pin load capacitance and the pullup (the external driver should not actively drive the pin high). An external pullup is not required as an onboard pullup is present. If the reset is driven by the onboard driver, the active low time will be approximately 2 μs, and once released, the rise time is again determined by the RC time constant.

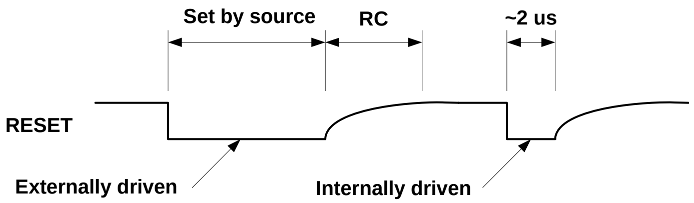


Figure 3-3. $\overline{\text{RESET}}$ Signal Timing

When $\overline{\text{RESET}}$ is active, the current is very low, but the MC13234/MC13237 is not totally disabled as in a programmed low power state.

## 3.3.2 Signal PTA2 (Factory Test Mode Enable)

Signal PTA2 is also an enable for a reserved factory test mode (see Section 9.3, “IO Port Pin Functional Description”). During a POR reset, signal PTA2 is an input and should not be driven high or test mode will be entered when POR is released. Good design practice is to tie PTA to ground through an external pulldown resistor.

Although I/O signal PTA2 is controlled as any other Port A signal it is suggested that PTA2 be limited to use as an output and be used only when required by applications needing the extra GPIO output.

### 3.3.3 MCU Background/Mode Select (PTA7/BKGD/MS)

The Background/Mode Select (BKGD/MS) shares its function with the MCU I/O port pin PTA7. While in reset, the pin functions as a mode select pin. Immediately after $\overline{\text{RESET}}$ rises, the pin functions as the background pin and can be used for background debug communication. While functioning as a Background/Mode Select pin, the pin includes an internal pullup device, input hysteresis, a standard output driver, and no output slew rate control. When used as an I/O port is becomes PTA7.

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

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

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

## 3.4 External Clock Connections

The MC13234/MC13237 provides the following clock sources:

- 32 MHz Reference Oscillator - all active clocks for the radio, CPU, and peripherals are derived from the reference oscillator. An external 32 MHz crystal is most commonly used with the reference oscillator, although an external clock source can be used.
- Onboard 1 kHz RC Oscillator - low power oscillator used for RTC and to wake-up from low power
- Optional 32.768 kHz Crystal Oscillator - used in place of the onboard 1 kHz oscillator for the RTC and to wake-up from low power.

The reference oscillator and the optional 32.768 kHz crystal oscillator require external connections which are shown in Figure 3-5.

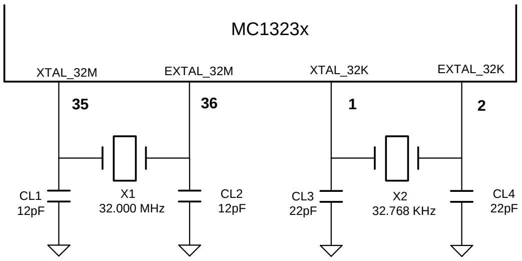


**Figure 3-4. MC13234/MC13237 External Clock Connections**

## 3.5 32 MHz Reference Oscillator

The reference oscillator must always be functional and can use an external crystal or an external source.

### 3.5.1 Description

The reference oscillator is shown in Figure 3-5. It has an amplifier block with Amplitude Level Control (ALC) to optimize power consumption and help startup. It uses an off-chip 32 MHz crystal and has internal programmable trim load capacitors. The primary load capacitors are external.

The external load capacitors CL1 and CL2 are typically in the 5 pF to 15 pF range and are determined by the requirements of the specific crystal. The manufacturer will specify the load capacitance and the external capacitors selected for CL1 and CL2 are used to match the stray circuit capacitance (the pin and PCB capacitance) to the required load capacitance. The onboard capacitors consist of two arrays on each pin, each trimmable by a 4-bit control field. The larger or coarse array can be from 0 - 4.215 pF with steps of 281 fF. The smaller or fine array can be from 0 - 300 fF with steps of 20 fF.

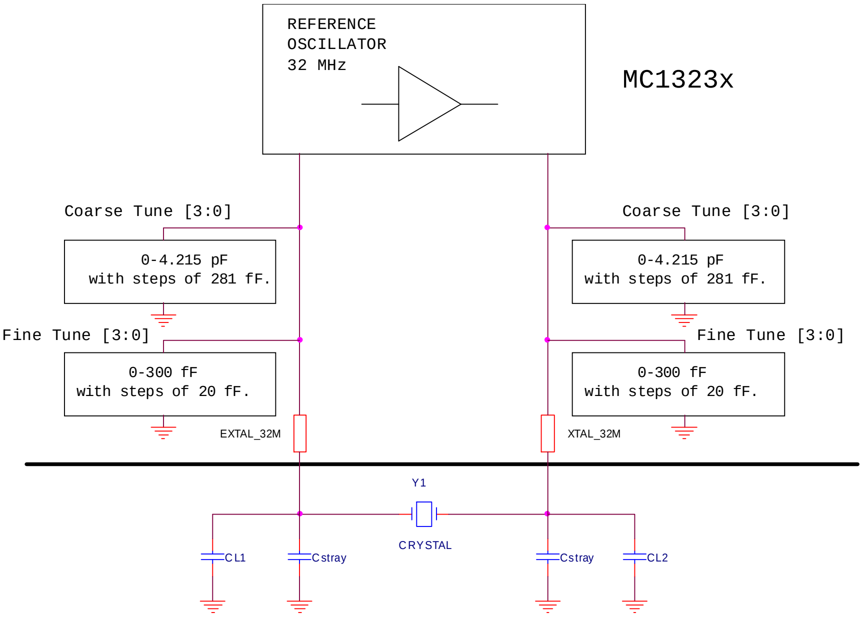


Figure 3-5. 32MHz Reference Oscillator Model

### 3.5.2 32 MHz Crystal Specifications

The IEEE 802.15.4 Standard requires a frequency tolerance less than or equal to +/- 40 ppm. With a suitable crystal (see Table 3-2), the device frequency tolerance can typically be trimmed to be less than +/- 30 ppm over all conditions.

Table 3-2. Recommended 32 MHz Crystal Specifications

| Parameter | Value | Unit | Condition |
|---|---|---|---|
| Frequency | 32.000000 | MHz | |
| Frequency tolerance (cut tolerance) | ± 10 | ppm | max at 25 °C |

Table 3-2. Recommended 32 MHz Crystal Specifications (continued)

| Parameter | Value | Unit | Condition |
|---|---|---|---|
| Frequency stability (temperature drift) | ± 16-18 | ppm | Over desired temperature range |
| Aging | ± 2 | ppm | max |
| Equivalent series resistance | 60 | Ω | max |
| Load capacitance | 9 | pF | max |
| Shunt capacitance | <2 | pF | max |
| Mode of oscillation | | | fundamental |

### 3.5.3 32 MHz Crystal Trimming

The MC13234/MC13237 reference oscillator frequency can be trimmed (via changing the load capacitance) through programming the System Oscillator Management and Control Register 2 (XTAL1_TRIM[7:0]) as described in Section 5.8.11, “System Oscillator Management and Control Register 2 (SOMC2)”. The XTAL1_TRIM[7:0] field has two control fields that set the trimming.

- XTAL1_TRIM[7:4] — selects course load capacitance; selects from 0–4.215 pF in 15 steps of approximately 281 fF
- XTAL_FTUNE[3:0] — selects fine tuning for load capacitance; selects from 0–300 fF in 15 steps of approximately 20 fF

The trimming procedure varies the frequency where as the trim value is increased, the frequency is decreased. This feature is useful to set the crystal frequency for the radio accuracy as required by the IEEE 802.15.4 specification.

Figure 3-6 and Figure 3-7 illustrate the variation is oscillator frequency vs. capacitive loading for a 9pF $C_L$ crystal.

- For Figure 3-6 the external load capacitors are present and the fine tune is centered while the course tune array is varied. The rate of frequency change is about 125 Hz per step.
- For Figure 3-7 the external load capacitors are present and the course tune is centered while the fine tune array is varied. The rate of frequency change is about 11 Hz per step.

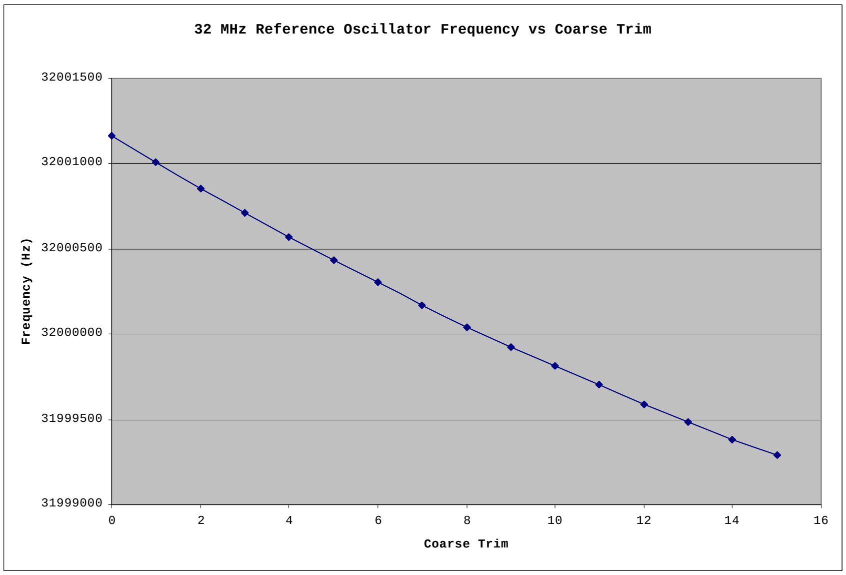


Figure 3-6. Typical MC13234/MC13237 Reference Oscillator Variation vs. XTAL1_TRIM[7:4] (XTAL1_TRIM[3:0] = 0x7 and external load capacitors are present)

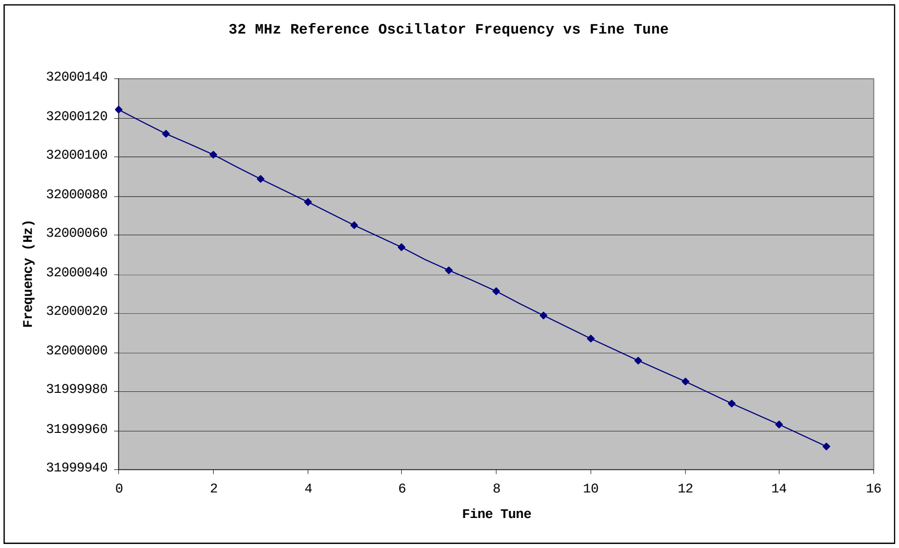


Figure 3-7. Typical MC13234/MC13237 Reference Oscillator Variation vs. XTAL1_TRIM[3:0] (XTAL1_TRIM[7:4] = 0x7 and external load capacitors are present)

A MC13234/MC13237 timer module (TPMx) output can be enabled by software to make measurements via a frequency counter to trim the reference oscillator.

## NOTE

- Good design practice dictates that external capacitors be selected to nominally center the reference frequency with the trim capacitance set for mid-range
- Good design practice also dictates that a user characterize their board to find nominal trim settings based on their crystal, layout, and pcb

characteristics. This should be done with a number of board examples to center the trim values across board variation

- With device-to-device and crystal variation, individual module trim at manufacture and test may be required.

## 3.6 32.768 kHz Crystal Oscillator (Optional)

The 32.768 kHz oscillator is strictly optional. The 32 kHz oscillator adds cost to a design, but is much more accurate than the onboard 1kHz oscillator.

The use of the 32.768 kHz oscillator versus the onboard 1 kHz oscillator is controlled via a register in the CRM (see Chapter 5, “System Management and Control”).

### 3.6.1 Description

The MC13234/MC13237 has a built-in 32 kHz crystal oscillator with an ALC feature. It uses an external 32.768 kHz crystal with a load capacitance of 12–16 pF. The load capacitors are external.

Figure 3-8 shows the user oscillator model. The model shows that there are external load capacitors for the oscillator crystal. On each leg of the oscillator (XTAL_32K and EXTAL_32K) there is:

- A fixed $C_L$ capacitor
- Stray capacitance - sum of pad, package, and trace capacitance (typically ~1.5pF)

The load capacitance $C_L$ to the crystal is symmetric and equals 1/2 ($C_L$ + $C_{stray}$). Target load capacitance is 12–16 pF and is determined by the crystal specification.

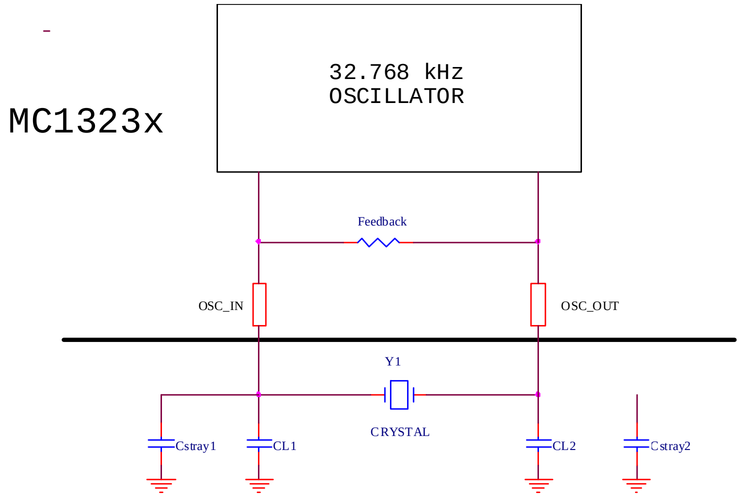


Figure 3-8. MC13234/MC13237 32.768 kHz Oscillator Model

### 3.6.2 32.768 kHz Crystal Specifications

Table 3-3 lists the recommended crystal specifications.

**Table 3-3. 32.768 kHz Recommended Crystal Specification**

| Parameter Name | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|
| Crystal frequency | | 32.768 | | kHz |
| Frequency tolerance @ 25 °C[1] | | ± 20 | | ppm |
| Frequency tolerance with temperature[1] | | | | ppm |
| Load capacitance | 12 | 12.5 | 16 | pF |
| Equivalent series resistance (ESR) | | | 35 | kΩ |
| Shunt capacitance | | | 1.35 | pF |
| Tolerated drive level | | | 1 | µW |

[1] Determined by customer desired frequency accuracy

## 3.7 Device Version ID

Bit[1,0] of SOPT2 register provide the device IDs for MC13234 and MC13237. See Section 5.8.5, “System Options Register 2 (SOPT2)” for details.

## 3.8 MC13234/MC13237 GPIO

The MC13234/MC13237 supports up to a total of 32 GPIO pins:

- MC13234: 32 GPIO pins; MC13237: 28 GPIO pins
- Pins have hysteresis inputs and configurable pullup device on all input pins
- Configurable slew rate and drive strength on all output pins.
- PTA2 is also a factory test mode enable.
- PTA3 is unavailable if used as IRQ
- If a 32.768 kHz crystal is used, PTA0/XTAL_32K and PTA1/EXTAL_32K pins cannot be used as GPIO.
- PTA4/XTAL_32KOUT is an optional 32 kHz clock output signal
- PTA7/BKGD/MS is not commonly used as GPIO because it is dedicated to the MCU debug port (BDM) and is an output only when used as GPIO

### 3.8.1 GPIO Characteristics

The GPIO hardware consists of four ports, A, B, C, and D. Each port consists of 8 signals (except for MC13237, port C only has 4 signals). With $\overline{\text{RESET}}$ active and immediately after exiting reset, all I/O pins default to high-impedance inputs with internal pullup devices disabled.

### NOTE

> To avoid extra current drain from floating input pins, the device 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. See Section Chapter 9, “Parallel Input and Output”).

Most of these pins share functionality with MCU peripheral functions. For information about controlling these pins as general-purpose I/O pins, see Section Chapter 9, “Parallel Input and Output”. For information about how and when on-chip peripheral systems use these pins, see the appropriate section from Table 3-4.

**Table 3-4. MC13234/MC13237 Pin Sharing References**

| Port Pins | Alternate Function | Reference[1] |
|---|---|---|
| PTA1–PTA0 | XTAL_32K–EXTAL_32K | Chapter 3, “System Considerations” |
| PTA3 | IRQ | Chapter 3, “System Considerations” |
| PTA4 | XTAL_32KOUT | Chapter 5, “System Management and Control” |
| PTA5–PTA6 | SDA–SCL | Chapter 13, “Inter-integrate Circuit (IIC)” |
| PTA7 | BKGD/MS | Chapter 18, “Development Support” |
| PTB0–PTB7 | KBI1P0–KBI1P7 | Chapter 11, “Modules (KBIx)” |
| PTC0–PTC3 | KBI2P0–KBI2P3 | MC13234 only |
| PTC4<br>PTC5<br>PTC6<br>PTC7 | SPICLK<br>SS_B<br>MISO<br>MOSI | Chapter 15, “Serial Peripheral Interface (SPI) Module” |
| PTD0–PTD3 | TPM0–TPM3 | Chapter 16, “Timer/PWM (TPM Module)” |
| PTD4 | CMT | Chapter 17, “Carrier Modulator Timer (CMT) Module” |
| PTD5–PTD6 | TXD-RXD | Chapter 14, “Serial Communications Interface (SCI)” |
| PTD1–PTD7,<br>PTC5 | AD0–AD7 | Chapter 12, “Analog-to-Digital Converter (ADC)” |

[1] See this section for information about modules that share these pins.

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from port data registers even though the peripheral module controls the pin direction by controlling the enable for the pin’s output buffer. See Chapter 9, “Parallel Input and Output” for details.

pullup and pulldown control consists of:

- 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 PTB7–PTB0, PTC3–PTC0 (MC13234 only), pins are controlled by the related KBI module and are configured for rising-edge/high-level sensitivity, the pullup enable control bits enable pulldown devices rather than pullup devices.

- When IRQ is configured as the IRQ input and is set to detect rising edges, the pullup enable control bit enables a pulldown device rather than a pullup device.

### 3.8.2 Signal Properties Summary

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

**Table 3-5. MC13234/MC13237 Digital Signal Properties**

| Pin Name | Dir | High Current Pin | Output Slew [1] | pullup[2] | Comments |
|---|---|---|---|---|---|
| **MCU SIGNALS** | | | | | |
| $\overline{\text{RESET}}$ | I | — | — | Y | Pin contains integrated pullup. |
| PTA0/XTAL_32K | I/O<br>Analog | N | SWC | SWC | Pullup and slew rate disabled when XTAL pin function. |
| PTA1/EXTAL_32K | I/O<br>Analog | N | SWC | SWC | Pullup and slew rate disabled when EXTAL pin function. |
| PTA2 | I/O | N | SWC | Y | **SPECIAL PIN - ENABLES TEST MODE**. Should be pulled low when exiting POR. |
| PTA3/IRQ | I/O | N | SWC | SWC | Enable bit IRQPE must be set to enable IRQ function. IRQ does not have a clamp diode to $V_{DD}$. IRQ should not be driven above $V_{DD}$.<br>Pullup/pulldown active when IRQ pin function enabled. Pullup forced on when IRQ enabled for falling edges; pulldown forced on when IRQ enabled for rising edges. Internal connection. |
| PTA4/ XTAL_32KOUT | I/O | N | SWC | SWC | |
| PTA5/SDA | I/O | N | SWC | SWC | |
| PTA6/SCL | I/O | N | SWC | SWC | |
| PTA7/BKGD/MS | O | N | SWC | SWC | Pullup enabled and slew rate disabled when BDM function enabled. |
| PTB0/KBI1P0 | I/O | N | SWC | SWC | Pullup/pulldown active when KBI pin function enabled. Pull-p forced on when KBI1Px enabled for falling edges; pulldown forced on when KBI1Px enabled for rising edges. |
| PTB1/KBI1P1 | I/O | N | SWC | SWC | |
| PTB2/KBI1P2 | I/O | N | SWC | SWC | |
| PTB3/KBI1P3 | I/O | N | SWC | SWC | |
| PTB4/KBI1P4 | I/O | N | SWC | SWC | |
| PTB5/KBI1P5 | I/O | N | SWC | SWC | |
| PTB6/KBI1P6 | I/O | N | SWC | SWC | |
| PTB7/KBI1P7 | I/O | N | SWC | SWC | |

Table 3-5. MC13234/MC13237 Digital Signal Properties (continued)

| Pin Name | Dir | High Current Pin | Output Slew [1] | pullup[2] | Comments |
|---|---|---|---|---|---|
| PTC0/KBI2P0 | I/O | N | SWC | SWC | MC13234 only. |
| PTC1/KBI2P1 | I/O | N | SWC | SWC | |
| PTC2/KBI2P2 | I/O | N | SWC | SWC | |
| PTC3/KBI2P3 | I/O | N | SWC | SWC | |
| PTC4/SPICLK | I/O | N | SWC | SWC | |
| PTC5/SS_B | I/O | N | SWC | SWC | |
| PTC6/MISO | I/O | N | SWC | SWC | |
| PTC7/MOSI | I/O | N | SWC | SWC | |
| PTD0/TPM0 | I/O | N | SWC | SWC | |
| PTD1/TPM1 | I/O | N | SWC | SWC | |
| PTD2/TPM2 | I/O | N | SWC | SWC | |
| PTD3/TPM3 | I/O | N | SWC | SWC | |
| PTD4/CMT | I/O | Y | SWC | SWC | 20 mA drive high or low |
| PTD5/TXD | I/O | N | SWC | SWC | |
| PTD6/RXD | I/O | N | SWC | SWC | |
| PTD7 | I/O | N | SWC | SWC | |

[1] SWC is software controlled slew rate, the register is associated with the respective port.

[2] SWC is software controlled pullup resistor, the register is associated with the respective port.

## 3.9 Transceiver RF Operation

The MC13234/MC13237 transceiver provides a simple RF interface with the following characteristics:

- Onboard Transmit/Receive (T/R) switch and LNA
- Two-pin bidirectional, differential RF port
- RF_Bias — switched 1.8 Vdc (VREG_ANA) bias reference for use with an external balun.

An internal integrated RX switch and TX power amplifier (PA) are used, and pins RF_P and RF_N are bidirectional and connect both for TX and RX. When receiving, the RX switch is enabled to the internal LNA and the TX PA is disabled. When transmitting, the RX switch is disabled (isolating the LNA) and the TX PA is enabled. RF_Bias pin provides a reference or bias voltage which is at 1.8 V for transmit and is at ground for receive. This signal can be used to provide the proper bias voltage to a balun that converts a single-ended antenna to the differential interface required by the RF device port.

Figure 3-9 shows a typical example with a balun. The RF_Bias is connected to the balun center-tap providing the proper DC bias voltage to the balun depending on RX or TX.

### NOTE

Figure 3-9 provides a simple example of an MC13234/MC13237 RF circuit. For detailed design information, Freescale recommends using provided reference designs as a starting point.

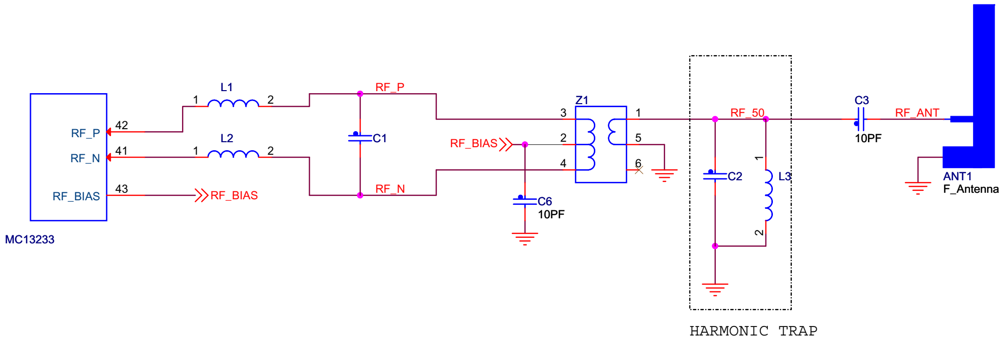


Figure 3-9. RF Operation with a Balun

## 3.10 Transceiver Indirect Access Register Application Functions

Operation of the indirect access register is described in Section 6.13.33, “Transceiver Indirect Access Registers (INDEX and DATA)”. These registers are generally reserved, but are used to provide three application functions:

1. 16-Bit Pseudo-Random Number Generator - see Section 6.11, “Hardware Pseudo-Random Number Generator”.
2. Set transmit power - the desired transmit power level of the transceiver power amplifier (PA) is set across three indirect access registers plus the direct access PA_PWR_CNTL Register. This is described in Section 6.13.25, “Power Amplifier (PA) Power Control Register (PA_PWR_CNTL)” and Freescale provides a software utility that is called to set desired output power.
3. Controlling the analog voltage regulator for low power operation - the MC13234/MC13237 has several onboard series voltage regulators, and the primary analog regulator for the radio must be controlled when entering and exiting lower power modes. Three indirect registers are involved:
   - Indirect Address 0x1D - disable default mode of analog regulator; set to value 0x00. This needs be done only once as part of initialization after a system reset of any kind.
   - Indirect Address 0x61 - enable direct control of analog regulator; set value to 0x01. This needs be done only once as part of initialization after a system reset of any kind.
   - Indirect Address 0x60 - provide direct control (on/off) of analog regulator. Set to value 0x01 (enable) to enable analog regulator before normal operation. Set to value 0x00 (disable) before entering low power operation.

## 3.11 General System Considerations for Low Power Operation

Low power is most important for battery operated applications such as 802.15.4 End Devices. Although the MC13234/MC13237 is inherently a low power device, both normal “run” as well as low power or “sleep” modes can be optimized for lowest net energy usage, i.e., longest battery life. The following descriptions give guidelines for optimizing lowest power.

**NOTE**

> This discussion is simply an overview of run and low power mode considerations. The user should become familiar with the referenced document material to understand detailed control and operation of the MC13234/MC13237 when running and entering/exiting low power modes.

### 3.11.1 Run Mode Considerations

Run mode current is dominated by the operating frequency of the MCU and its peripherals and the transceiver (radio) over-the-air RF activity (transmit, receive, and CCA/ED cycles). To minimize current usage:

1. Disable bus clocks to all unused peripherals - see Section 5.8.12, “System Clock Gating Control 1 Register (SCGC1)” and Section 5.8.13, “System Clock Gating Control 2 Register (SCGC2)”.
2. Consider running at a lower CPU/bus clock frequency - see Section 5.8.10, “System Oscillator Management and Control Register 1 (SOMC1)”.
   — Programming a lower run frequency impacts peripheral baud rates and speeds, as well as application performance. It is the user’s responsibility to account for these changes when operating at the lower frequency.
   — Flash can be erased and programmed only at full CPU/bus clock rate (32MHz/16MHz)
3. If possible, minimize transceiver “on” time - this is most likely to be receiver “on” time. Evaluate protocol usage to minimize receiver activity if possible.

### 3.11.2 Low Power Mode Considerations

**NOTE**

> This discussion is primarily an overview of low power mode considerations. The user is directed to Section 5.7.6, “Stop3 Mode” and Section 5.7.7, “Mode Selection” for additional detailed information.

Depending on use model, the low power mode can dominate overall current usage. Low power considerations include:

1. All GPIO must be configured for low power operation -
   — Configure all unused GPIO - see Section 3.8.1, “GPIO Characteristics”
   — Some GPIO may need to be re-configured when entering/exiting low power mode - this is to assure that there is no excess current usage.
2. Wake-up is possible from -
   — Push button operation via KBI interrupt request

- Timer operation using the Real Time Counter (RTC)
- UART receive activity via an RXD transition interrupt request

3. When using the RTC -
   - If the wake-up time period can vary by approximately +/-40% use the internal 1 kHz oscillator. This has lowest power and lowest cost
   - If the wake-up time period must be crystal accurate, use the external clock option for the RTC timer with a 32.768 kHz crystal. This has the advantage of crystal accuracy and the disadvantage of higher cost.
4. Stop3 mode - see Section 5.7.6, “Stop3 Mode”
   - Actual wake-up time from low power to run mode is less than 1 ms - see data sheet (MC13237.pdf)
   - The analog regulator must be disabled for lowest power in Stop3 - see Section 3.10, “Transceiver Indirect Access Register Application Functions”
5. Best practice is to disable the MCU low voltage detect (LVD) during low power mode. Transition between Low-Power Run mode and Stop3 mode needs extra steps. See Section 5.7.6.6, “Stop3 and LPRun Mode Transition”
   - LVD is only available for Stop3 mode
   - Use LVD only if required.
   - LVD current usage penalty is significant - see data sheet (MC13237.pdf)

### 3.11.3 Entering/Exiting Low Power Mode

The following flow summarizes application steps for entering low power from a run condition:

1. Exit any stack/application gracefully - save any required parameters, shut down any tasks that are affected.
2. As required configure any GPIO for low power operation -it is assumed that during system initialization, any unused GPIO are configured properly so as to not cause excess leakage current. At this point, re-configure any GPIO in use that may special handling during low power so as to not cause excess leakage.
3. Disable the transceiver analog regulator if required - see Section 3.10, “Transceiver Indirect Access Register Application Functions”.
4. Disable any active peripheral clocks.
5. Initiate any required wake-up sources / interrupts - choosing means and clocks as previously recommended.
6. Clear LVW/LVD as required.
7. Prepare “Stop” mode conditions in System Control Registers, see Section 5.8, “System Control Register Definitions”. This includes setting the Stop Enable bit.
8. Execute a STOP instruction.

Recovery from Stop3 is from a “suspended” state (registers and RAM were retained) and resumes as an interrupt service routine as triggered by the wake-up source. In this situation, recovery code typically does not use normal system initialization. The recovery code must restore any conditions (such as GPIO usage) to pre-sleep operation.

# Chapter 4
# Memory

## 4.1 MC13234/MC13237 Memory Map

As shown in Figure 4-1, MC13234/MC13237 on-chip memory consists of RAM, flash program memory for nonvolatile program and data storage, and I/O and control/status registers:

- 8064 Bytes of RAM, divided into two blocks (6016 Bytes and 2048 Bytes)
- 131072 Bytes of flash — organized as eight 16384-byte pages.
- Control/Status Registers — see Section 4.3, “Register Addresses and Bit Assignments”

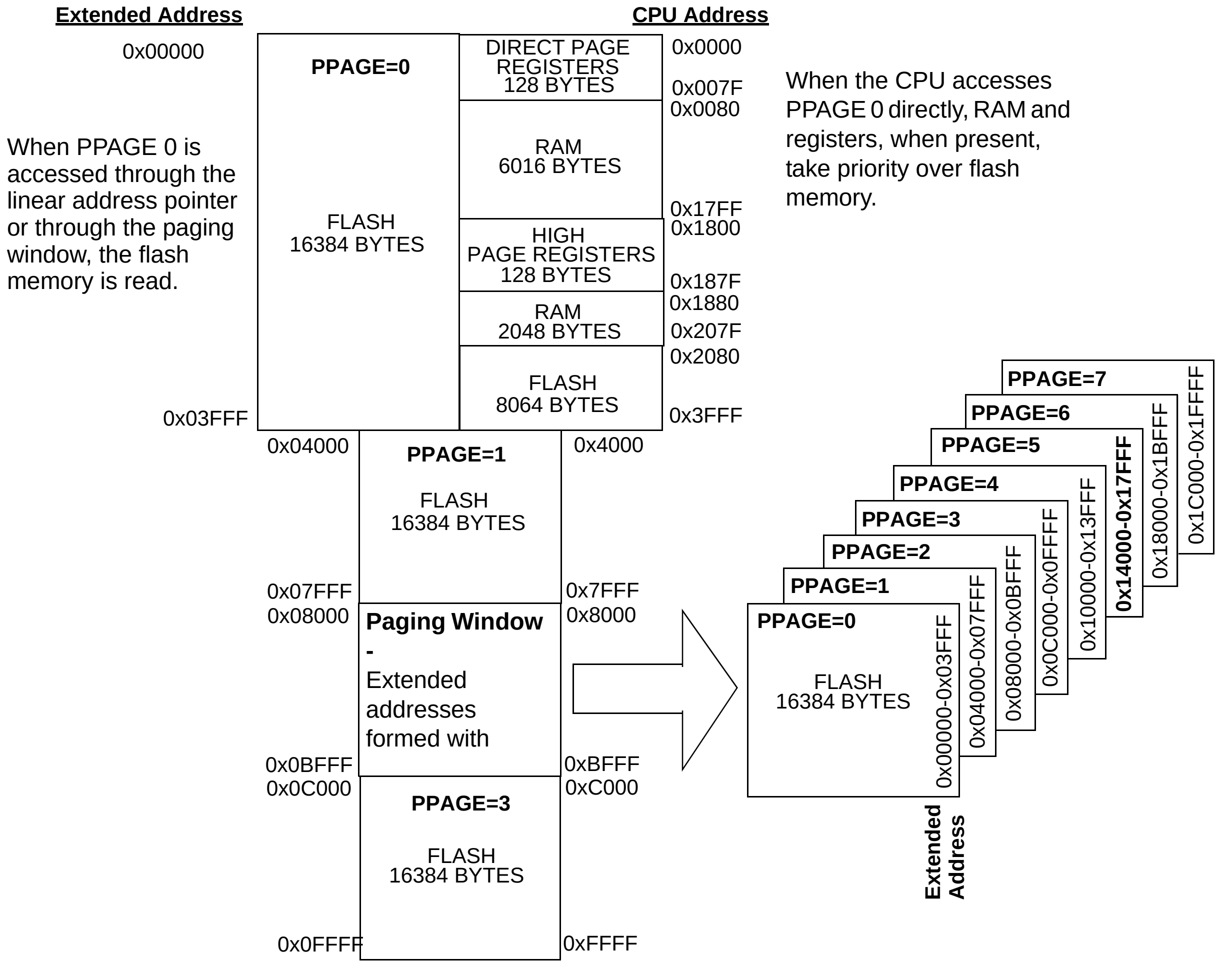


**Figure 4-1. MC13234/MC13237 Memory Map**

## 4.2 Reset and Interrupt Vector Assignments

Table 4-1 shows address assignments for reset and interrupt vectors. These vectors are located at the top of Page 3 of the flash because they are accessed at the top of the CPU direct address space.

Table 4-1. Reset and Interrupt Vectors

| | Address (High/Low) | Vector | Vector Name |
|---|---|---|---|
| 26 | 0xFFCA:0xFFCB | ADC (MC13237 only) | Vadc |
| 25 | 0xFFCC:0xFFCD | RTC | Vrtc |
| 24 | 0xFFCE:0xFFCF | KBI1 Interrupt | Vkeyboard1 |
| 23 | 0xFFD0:0xFFD1 | KBI2 Interrupt (MC13234 only) | Vkeyboard2 |
| 22 | 0xFFD2:0xFFD3 | IIC | Viicx |
| 21 | 0xFFD4:0xFFD5 | CMT | Vcmt |
| 20 | 0xFFD6:0xFFD7 | SCI Transmit | Vscitx |
| 19 | 0xFFD8:0xFFD9 | SCI Receive | Vscirx |
| 18 | 0xFFDA:0xFFDB | SCI Error | Vscierr |
| 17 | 0xFFDC:0xFFDD | SPI | Vspi |
| 16 | 0xFFDE:0xFFDF | TPM4 Overflow | Vtpm4ovf |
| 15 | 0xFFE0:0xFFE1 | TPM4 Channel 0 | Vtpm4ch0 |
| 14 | 0xFFE2:0xFFE3 | TPM3 Overflow | Vtpm3ovf |
| 13 | 0xFFE4:0xFFE5 | TPM3 Channel 0 | Vtpm3ch0 |
| 12 | 0xFFE6:0xFFE7 | TPM2 Overflow | Vtpm2ovf |
| 11 | 0xFFE8:0xFFE9 | TPM2 Channel 0 | Vtpm2ch0 |
| 10 | 0xFFEA:0xFFEB | TPM1 Overflow | Vtpm1ovf |
| 9 | 0xFFEC:0xFFED | TPM1 Channel 0 | Vtpm1ch0 |
| 8 | 0xFFEE:0xFFEF | AES 128 | Vaes128 |
| 7 | 0xFFF0:0xFFF1 | 802.15.4 RX watermark | Vrx_watermark_802_15_4 |
| 6 | 0xFFF2:0xFFF3 | 802.15.4 Timers | Vtimers_802_15_4 |
| 5 | 0xFFF4:0xFFF5 | 802.15.4 Transmit (SEQIRQ,LO1_UNLOCK_IRQ,TXIRQ) | Vtx_802_15_4 |
| 4 | 0xFFF6:0xFFF7 | 802.15.4 Receive (SEQIRQ,CCAIRQ,FILTERFAIL_IRQ,LO1_UNLOCK_IRQ,RXIRQ) | Vrx_802_15_4 |
| 3 | 0xFFF8:0xFFF9 | Low Voltage Detect or Low Voltage Warning | Vlvd |
| 2 | 0xFFFA:0xFFFB | IRQ | Virq |
| 1 | 0xFFFC:0xFFFD | SWI | Vswi |
| 0 | 0xFFFE:0xFFFF | Reset | Vreset |

## 4.3 Register Addresses and Bit Assignments

The various control and status registers in MC13234/MC13237 are divided into the following groups:

- Direct-page registers (Address 0x0000-0x007F) - are located in the first 128 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
  - Direct-page registers can be accessed with efficient direct addressing mode instructions
  - Bit manipulation instructions can be used to access any bit in any direct-page register
  - Table 4-2 shows a summary of all user-accessible direct-page registers and control bits.
- High-page registers (Address 0x1800-0x187F) - 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. Table 4-3 shows a summary of all high-page registers and control bits. The high-page registers have two use models:
  - Conventional direct access - these are the majority of registers
  - Indirect access - addresses 0x185B and 0x185C are used to access a block of transceiver register using an indirect access method. See Section 6.13.33, “Transceiver Indirect Access Registers (INDEX and DATA)” for details.
- Flash control registers (Address 0xFFB0–0xFFBF) - this nonvolatile register area consists of a block of 16 locations in flash memory and includes:
  - NVPROT and NVOPT are loaded into working registers at reset
  - An 8-byte back door comparison key that optionally allows a user to gain controlled access to secure memory

**NOTE**

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

Table 4-2 shows a summary of all user-accessible direct-page registers and control bits, Table 4-3 shows a summary of all high-page registers and control bits, and Table 4-4 shows reserved flash memory addresses. For these tables:

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

### Table 4-2. 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 | TPM1SC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x0009 | TPM1CNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x000A | TPM1CNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x000B | TPM1MODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x000C | TPM1MODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x000D | TPM1C0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x000E | TPM1C0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x000F | TPM1C0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0010 | TPM2SC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x0011 | TPM2CNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0012 | TPM2CNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0013 | TPM2MODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0014 | TPM2MODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0015 | TPM2C0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x0016 | TPM2C0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0017 | TPM2C0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0018 | TPM3SC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x0019 | TPM3CNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x001A | TPM3CNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001B | TPM3MODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x001C | TPM3MODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x001D | TPM3C0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x001E | TPM3C0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x001F | TPM3C0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0020 | 802.15.4 AES CONTROL1 | CLEAR | START | SELFTST | CTR | CBC | CLEAR | LOAD_ MAC | AES_MSK |
| 0x0021 | 802.15.4 AES CONTROL2 | DATA_REG_TYPE_SELECT | | | — | — | IRQ_FLAG | TSTPAS | — |
| 0x0022 | 802.15.4 ASM Data_0 | DATA[127:120] | | | | | | | |
| 0x0023 | 802.15.4 ASM Data_1 | DATA[119:112] | | | | | | | |
| 0x0024 | 802.15.4 ASM Data_2 | DATA[111:104] | | | | | | | |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0025 | 802.15.4 ASM Data_3 | DATA[103:96] | | | | | | | |
| 0x0026 | 802.15.4 ASM Data_4 | DATA[95:88] | | | | | | | |
| 0x0027 | 802.15.4 ASM Data_5 | DATA[87:80] | | | | | | | |
| 0x0028 | 802.15.4 ASM Data_6 | DATA[79:72] | | | | | | | |
| 0x0029 | 802.15.4 ASM Data_7 | DATA[71:64] | | | | | | | |
| 0x002A | 802.15.4 ASM Data_8 | DATA[63:56] | | | | | | | |
| 0x002B | 802.15.4 ASM Data_9 | DATA[55:48] | | | | | | | |
| 0x002C | 802.15.4 ASM Data_A | DATA[47:40] | | | | | | | |
| 0x002D | 802.15.4 ASM Data_B | DATA[39:32] | | | | | | | |
| 0x002E | 802.15.4 ASM Data_C | DATA[31:24] | | | | | | | |
| 0x002F | 802.15.4 ASM Data_D | DATA[23:16] | | | | | | | |
| 0x0030 | 802.15.4 ASM Data_E | DATA[15:8] | | | | | | | |
| 0x0031 | 802.15.4 ASM Data_F | DATA[7:0] | | | | | | | |
| 0x0032 | KBI1SC | 0 | 0 | 0 | 0 | KBF | KBACK | KBIE | KBIMOD |
| 0x0033 | KBI1PE | KBIPE7 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| 0x0034 | KBI1ES | KBEDG7 | KBEDG6 | KBEDG5 | KBEDG4 | KBEDG3 | KBEDG2 | KBEDG1 | KBEDG0 |
| 0x0035[1] | KBI2SC | 0 | 0 | 0 | 0 | KBF | KBACK | KBIE | KBIMOD |
| 0x0036 | KBI2PE | KBIPE7 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| 0x0037 | KBI2ES | KBEDG7 | KBEDG6 | KBEDG5 | KBEDG4 | KBEDG3 | KBEDG2 | KBEDG1 | KBEDG0 |
| 0x0038 | SCI1BDH | LBKDIE | RXEDGIE | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| 0x0039 | SCI1BDL | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| 0x003A | SCI1C1 | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| 0x003B | SCI1C2 | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| 0x003C | SCI1S1 | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| 0x003D | SCI1S2 | LBKDIF | RXEDGIF | 0 | RXINV | RWUID | BRK13 | LBKDE | RAF |
| 0x003E | SCI1C3 | R8 | T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE |
| 0x003F | SCI1D | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0040 | 802.15.4 PHY TXD_ADR_PNTR0 | — | — | — | TXD_ADR_PNTR[12:8] | | | | |
| 0x0041 | TXD_ADR_PNTR1 | TXD_ADR_PNTR[7:0] | | | | | | | |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0042 | **RXD_ADR_PNTR0** | — | — | — | RXD_ADR_PNTR[12:8] | | | | |
| 0x0043 | **RXD_ADR_PNTR1** | RXD_ADR_PNTR[7:0] | | | | | | | |
| 0x0044 | **CNTRL1** | TMRTRIGEN | SLOTTED | CCABFRTX | RXACKRQD | AUTOACK | XCVSEQ | | |
| 0x0045 | **CNTRL2** | TRCV_MSK | TC3TMOUT | PANCORDNTR | CCATYPE | | TMRLOAD | PROMISCUOUS | TC2’EN |
| 0x0046 | **CNTRL3** | CRC_MSK | LO1_UNLOCK_MSK | FILTERFAIL_MSK | RX_WMRK_MSK | CCAMSK | RXMSK | TXMSK | SEQMSK |
| 0x0047 | **CNTRL4** | TMR4MSK | TMR3MSK | TMR2MSK | TMR1MSK | TMR4CMP_EN | TMR3CMP_EN | TMR2CMP_EN | TMR1CMP_EN |
| 0x0048 | **SRC_CNTRL** | INDEX | | | | ACK_FRM_PND | SRCADDR_EN | INDEX_EN | INDEX_DISABLE |
| 0x0049 | **SRC_ADDRS_SUM_DATA0** | SRC_ADDRS_SUM_DATA[15:8] | | | | | | | |
| 0x004A | **SRC_ADDRS_SUM_DATA1** | SRC_ADDRS_SUM_DATA[7:0] | | | | | | | |
| 0x004B | **RX_WTR_MARK** | RX_WTR_MARK[7:0] | | | | | | | |
| 0x004C | **RX_BYTE_COUNT** | RX_BYTE_COUNT[7:0] | | | | | | | |
| 0x004D | **STATUS1** | CRCVALID | CCA | SRCADDR | PI | TMR4IRQ | TMR3IRQ | TMR2IRQ | TMR1IRQ |
| 0x004E | **STATUS2** | RX_FRM_PEND | LO1_UNLOCK_IRQ | FILTERFAIL_IRQ | RXWTRMRKIRQ | CCAIRQ | RXIRQ | TXIRQ | SEQIRQ |
| 0x004F | Reserved | | | | | | | | |
| 0x0050 | **CCAFNL** | CCAFNL[7:0] | | | | | | | |
| 0x0051 | **EVENT_TMR0** | EVENT_TMR[23:16] | | | | | | | |
| 0x0052 | **EVENT_TMR1** | EVENT_TMR[15:8] | | | | | | | |
| 0x0053 | **EVENT_TMR2** | EVENT_TMR[7:0] | | | | | | | |
| 0x0054 | **TIMESTAMP0** | TIMESTAMP[23:16] | | | | | | | |
| 0x0055 | **TIMESTAMP1** | TIMESTAMP[15:8] | | | | | | | |
| 0x0056 | **TIMESTAMP3** | TIMESTAMP[7:0] | | | | | | | |
| 0x0057 | **T3CMP0** | T3CMP[23:16] | | | | | | | |
| 0x0058 | **T3CMP1** | T3CMP[15:8] | | | | | | | |
| 0x0059 | **T3CMP2** | T3CMP[7:0] | | | | | | | |
| 0x005A | **TC2 PRIME0** | TC2’[15:8] | | | | | | | |
| 0x005B | **TC2 PRIME1** | TC2’[7:0] | | | | | | | |
| 0x005C | **MACSHORTADDRS0** | MACSHORTADDRS[15:8] | | | | | | | |
| 0x005D | **MACSHORTADDRS1** | MACSHORTADDRS[7:0] | | | | | | | |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x005E | MACPANID0 | MACPANID[15:8] | | | | | | | |
| 0x005F | MACPANID1 | MACPANID[7:0] | | | | | | | |
| 0x0060 | CMTCGH1 | PH7 | PH6 | PH5 | PH4 | PH3 | PH2 | PH1 | PH0 |
| 0x0061 | CMTCGL1 | PL7 | PL6 | PL5 | PL4 | PL3 | PL2 | PL1 | PL0 |
| 0x0062 | CMTCGH2 | SH7 | SH6 | SH5 | SH4 | SH3 | SH2 | SH1 | SH0 |
| 0x0063 | CMTCGL2 | SL7 | SL6 | SL5 | SL4 | SL3 | SL2 | SL1 | SL0 |
| 0x0064 | CMTOC | IROL | CMTPOL | IROPEN | CMTDIV2 | 0 | 0 | 0 | 0 |
| 0x0065 | CMTMSC | EOCF | CMTDIV1 | CMTDIV0 | EXSPC | BASE | FSK | EOCIE | MCGEN |
| 0x0066 | CMTCMD1 | MB15 | MB14 | MB13 | MB12 | MB11 | MB10 | MB9 | MB8 |
| 0x0067 | CMTCMD2 | MB7 | MB6 | MB5 | MB4 | MB3 | MB2 | MB1 | MB0 |
| 0x0068 | CMTCMD3 | SB15 | SB14 | SB13 | SB12 | SB11 | SB10 | SB9 | SB8 |
| 0x0069 | CMTCMD4 | SB7 | SB6 | SB5 | SB4 | SB3 | SB2 | SB1 | SB0 |
| 0x006A | IIC1A | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | 0 |
| 0x006B | IIC1F | MULT | | TAP2 TAP1 ICR | | | | | |
| 0x006C | IIC1C1 | IICEN | IICIE | MST | TX | TXAK | RSTA | 0 | 0 |
| 0x006D | IIC1S | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| 0x006E | IIC1D | DATA | | | | | | | |
| 0x006F | IIC1C2 | GCAEN | ADEXT | 0 | 0 | 0 | AD10 | AD9 | AD8 |
| 0x0070 | SPIC1 | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| 0x0071 | SPIC2 | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| 0x0072 | SPIBR | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| 0x0073 | SPIS | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| 0x0074 | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0075 | SPID | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0076 | IRQSC | 0 | IRQPDD | IRQEDG | IRQPE | IRQF | IRQACK | IRQIE | IRQMOD |
| 0x0077 | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0078 | PPAGE | 0 | 0 | 0 | 0 | 0 | XA16 | XA15 | XA14 |
| 0x0079 | LAP2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | LA16 |
| 0x007A | LAP1 | LA15 | LA14 | LA13 | LA12 | LA11 | LA10 | LA9 | LA8 |
| 0x007B | LAP0 | LA7 | LA6 | LA5 | LA4 | LA3 | LA2 | LA1 | LA0 |
| 0x007C | LWP | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 0x007D | LBP | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 0x007E | LB | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 0x007F | LAPAB | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

[1] Address 0x0035 to 0x0037, KBI2 registers are available in MC13234 only.

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

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1800 | **SRS** | POR | PIN | COP | ILOP | ILAP | 0 | LVD | 0 |
| 0x1801 | **SBDFR** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BDFR |
| 0x1802 | **SOPT1** | COPE | COPT | STOPE | ADCHT | 0 | 0 | BKGDPE | 0 |
| 0x1803 | **SOPT2** | COPCLKS | 0 | 0 | ADCCG[1] | 0 | 0 | PID | |
| 0x1804 | **SCIC4** | 0 | 0 | 0 | BRFA[4:0] | | | | |
| 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 | 0 |
| 0x1809 | **SPMSC2** | LPR | LPRS | LPWUI | 0 | — | 0 | PPDE | PPDC |
| 0x180A | Reserved | — | — | — | — | — | — | — | — |
| 0x180B | **SPMSC3** | LVWF | LVWACK | LVDV | LVWV | LVWIE | 0 | 0 | 0 |
| 0x180C | **SOMC1** | RDIV | | | XTAL0EN | XTAL0-STEN | XTAL1-STEN | — | XTAL0_TST_OUT |
| 0x180D | **SOMC2** | XTAL1_TRIM[7:0] | | | | | | | |
| 0x180E | **SCGC1** | 1 | TPM4 | TPM3 | TPM2 | TPM1 | CMT | IIC | SCI |
| 0x180F | **SCGC2** | — | DBG | FLS | IRQ | KBI1 | — | RTC | 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 |
| 0x1817 | **DBGFL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1818 | **DBGCAX** | RWAEN | RWA | PAGSEL | 0 | 0 | 0 | 0 | Bit 16 |
| 0x1819 | **DBGCBX** | RWBEN | RWB | PAGSEL | 0 | 0 | 0 | 0 | Bit 16 |
| 0x181A | **DBGCCX** | RWCEN | RWC | PAGSEL | 0 | 0 | 0 | 0 | Bit 16 |
| 0x181B | **DBGFX** | PPACC | 0 | 0 | 0 | 0 | 0 | 0 | Bit 16 |
| 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 | Reserved | — | — | — | — | — | — | — | — |
| 0x1821 | **FOPT** | KEYEN | | 0 | 0 | 0 | 0 | SEC | |
| 0x1822 | Reserved | — | — | — | — | — | — | — | — |
| 0x1823 | **FCNFG** | 0 | 0 | KEYACC | 0 | 0 | 0 | 0 | 0 |
| 0x1824 | **FPROT** | FPS | | | | | | | FPOPEN |
| 0x1825 | **FSTAT** | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| 0x1826 | **FCMD** | 0 | FCMD | | | | | | |
| 0x1827 | Reserved | — | — | — | — | — | — | — | — |
| 0x1828 | **RTCSC** | RTIF | RTCLKS | | RTIE | RTCPS | | | |
| 0x1829 | **RTCCNTL** | RTCCNTL[7:0] | | | | | | | |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x182A | **RTCMODL** | RTCMODL[7:0] | | | | | | | |
| 0x182B | **RTCCNTH** | RTCCNTH[7:0] | | | | | | | |
| 0x182C | **RTCMODH** | RTCMODH[7:0] | | | | | | | |
| 0x182D | Reserved | | | | | | | | |
| 0x182E | Reserved | | | | | | | | |
| 0x182F | Reserved | | | | | | | | |
| 0x1830 | **PTAPE** | PTAPE7 | PTAPE6 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| 0x1831 | **PTASE** | PTASE7 | PTASE6 | PTASE5 | PTASE4 | PTASE3 | PTASE2 | PTASE1 | PTASE0 |
| 0x1832 | **PTADS** | PTADS7 | PTADS6 | PTADS5 | PTADS4 | PTADS3 | PTADS2 | PTADS1 | PTADS0 |
| 0x1833 | Reserved | — | — | — | — | — | — | — | — |
| 0x1834 | **PTBPE** | PTBPE7 | PTBPE6 | PTBPE5 | PTBPE4 | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| 0x1835 | **PTBSE** | PTBSE7 | PTBSE6 | PTBSE5 | PTBSE4 | PTBSE3 | PTBSE2 | PTBSE1 | PTBSE0 |
| 0x1836 | **PTBDS** | PTBDS7 | PTBDS6 | PTBDS5 | PTBDS4 | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| 0x1837 | Reserved | — | — | — | — | — | — | — | — |
| 0x1838 | **PTCPE** | PTCPE7 | PTCPE6 | PTCPE5 | — | — | — | — | PTCPE0 |
| 0x1839 | **PTCSE** | PTCSE7 | PTCSE6 | PTCSE5 | — | — | — | — | PTCSE0 |
| 0x183A | **PTCDS** | PTCDS7 | PTCDS6 | PTCDS5 | — | — | — | — | PTCDS0 |
| 0x183B | Reserved | — | — | — | — | — | — | — | — |
| 0x183C | **PTDPE** | PTDPE7 | PTDPE6 | PTDPE5 | PTDPE4 | PTDPE3 | PTDPE2 | PTDPE1 | PTDPE0 |
| 0x183D | **PTDSE** | PTDSE7 | PTDSE6 | PTDSE5 | PTDSE4 | PTDSE3 | PTDSE2 | PTDSE1 | PTDSE0 |
| 0x183E | **PTDDS** | PTDDS7 | PTDDS6 | PTDDS5 | PTDDS4 | PTDDS3 | PTDDS2 | PTDDS1 | PTDDS0 |
| 0x183F | Reserved | | | | | | | | |
| 0x1840 | **T1CMP0** | T1CMP[23:16] | | | | | | | |
| 0x1841 | **T1CMP1** | T1CMP[15:8] | | | | | | | |
| 0x1842 | **T1CMP2** | T1CMP[7:0] | | | | | | | |
| 0x1843 | **T2CMP0** | T2CMP[23:16] | | | | | | | |
| 0x1844 | **T2CMP1** | T2CMP[15:8] | | | | | | | |
| 0x1845 | **T2CMP2** | T2CMP[7:0] | | | | | | | |
| 0x1846 | **T4CMP0** | T4CMP[23:16] | | | | | | | |
| 0x1847 | **T4CMP1** | T4CMP[15:8] | | | | | | | |
| 0x1848 | **T4CMP2** | T4CMP[7:0] | | | | | | | |
| 0x1849 | **LO1_FRAC0** | LO1_FRAC[15:8] | | | | | | | |
| 0x184A | **LO1_FRAC1** | LO1_FRAC[7:0] | | | | | | | |
| 0x184B | **LO1_INT** | LO1_INT[7:0] | | | | | | | |
| 0x184C | **PA PWR CNTL** | Reserved[7:0] | | | | | | | |
| 0x184D | **MACLONG ADDRS0** | MACLONGADDRS[63:56] | | | | | | | |
| 0x184E | **MACLONG ADDRS1** | MACLONGADDRS[55:48] | | | | | | | |
| 0x184F | **MACLONG ADDRS2** | MACLONGADDRS[47:40] | | | | | | | |
| 0x1850 | **MACLONG ADDRS3** | MACLONGADDRS[39:32] | | | | | | | |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1851 | **MACLONG ADDRS4** | MACLONGADDRS[31:24] | | | | | | | |
| 0x1852 | **MACLONG ADDRS5** | MACLONGADDRS[23:16] | | | | | | | |
| 0x1853 | **MACLONG ADDRS6** | MACLONGADDRS[15:8] | | | | | | | |
| 0x1854 | **MACLONG ADDRS7** | MACLONGADDRS[7:0] | | | | | | | |
| 0x1855 | **MAXFRAME LENGTH** | MAXFRAMELENGTH[7:0] | | | | | | | |
| 0x1856 | **RX FRAME FILTER** | — | — | — | NS_FT | CMD_FT | ACK_FT | DATA_FT | BEACON_ FT |
| 0x1857 | **MAC FRAME VER**<br>**FMR REV&TMR** | TMR_PRESCALE[2:0] | | | — | — | — | FRM_VER | |
| 0x1858 | **CCA_ THRESHOLD** | CCA_THRESHOLD[7:0] | | | | | | | |
| 0x1859 | **CCA_OFFSET_ CMP** | CCA_OFFSET_CMP[7:0] | | | | | | | |
| 0x185A | **FSM** | -- | -- | FSM[5:0] | | | | | |
| 0x185B | **INDIRECT ACCESS INDEX** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x185C | **INDIRECT ACCESS DATA** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x185D | Reserved | — | — | — | — | — | — | — | — |
| 0x185E | Reserved | — | — | — | — | — | — | — | — |
| 0x185F | Reserved | — | — | — | — | — | — | — | — |
| 0x1860 | **TPM4SC** | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x1861 | **TPM4CNTH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1862 | **TPM4CNTL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1863 | **TPM4MODH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1864 | **TPM4MODL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1865 | **TPM4C0SC** | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x1866 | **TPM4C0VH** | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1867 | **TPM4C0VL** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1868 | Reserved | — | — | — | — | — | — | — | — |
| 0x1869 | Reserved | — | — | — | — | — | — | — | — |
| 0x186A | Reserved | — | — | — | — | — | — | — | — |
| 0x186B | Reserved | — | — | — | — | — | — | — | — |
| 0x186C | Reserved | — | — | — | — | — | — | — | — |
| 0x186D | Reserved | — | — | — | — | — | — | — | — |
| 0x186E | Reserved | — | — | — | — | — | — | — | — |
| 0x186F | Reserved | — | — | — | — | — | — | — | — |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1870[2] | **ADCSC1** | COCO | AIEN | ADCO | ADCH | | | | |
| 0x1871 | **ADCSC2** | ADACT | ADTRG | ACFE | ACFGT | 0 | 0 | REFSEL | |
| 0x1872 | **ADCRH** | 0 | 0 | 0 | 0 | ADR11 | ADR10 | ADR9 | ADR8 |
| 0x1873 | **ADCRL** | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |
| 0x1874 | **ADCCVH** | 0 | 0 | 0 | 0 | — | — | ADCV9 | ADCV8 |
| 0x1875 | **ADCCVL** | ADCV7 | ADCV6 | ADCV5 | ADCV4 | ADCV3 | ADCV2 | ADCV1 | ADCV0 |
| 0x1876 | **ADCCFG** | ADLPC | ADIV | | ADLSMP | MODE | | ADICLK | |
| 0x1877 | **APCTL1** | ADPC7 | ADPC6 | ADPC5 | ADPC4 | ADPC3 | ADPC2 | ADPC1 | ADPC0 |
| 0x1878 | Reserved | — | — | — | — | — | — | — | — |
| 0x1879 | Reserved | — | — | — | — | — | — | — | — |
| 0x187A | Reserved | — | — | — | — | — | — | — | — |
| 0x187B | Reserved | — | — | — | — | — | — | — | — |
| 0x187C | Reserved | — | — | — | — | — | — | — | — |
| 0x187D | Reserved | — | — | — | — | — | — | — | — |
| 0x187E | Reserved | — | — | — | — | — | — | — | — |
| 0x187F | Reserved | — | — | — | — | — | — | — | — |

[1] ADCCG is available in MC13237 only.

[2] Address 0x1870 to 0x1877, ADC registers are available in MC13237 only.

Several reserved flash memory locations, shown in Table 4-4, are used for storing values used by several registers. These registers include an 8-byte back door 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.

**Table 4-4. Reserved Flash Memory Addresses**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0xFFAE | Reserved | — | — | — | — | — | — | — | — |
| 0xFFAF | Reserved | — | — | — | — | — | — | — | — |
| **0xFFB0 – 0xFFB7** | **NVBACKKEY** | 8-Byte Comparison Key | | | | | | | |
| **0xFFB8 – 0xFFBC** | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| **0xFFBD** | **NVPROT** | FPS | | | | | | | FPOPEN |
| **0xFFBE** | Reserved | — | — | — | — | — | — | — | — |
| **0xFFBF** | **NVOPT** | KEYEN | | 0 | 0 | 0 | 0 | SEC | |

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

## 4.4 Memory Management Unit

The memory management unit (MMU) allows the program and data space for the CPU to be extended beyond the 64K byte CPU addressable memory map. The MMU uses a paging scheme where the extended flash memory is accessed within a 16-kilobyte window located between addresses 0x8000 and 0xBFFF of the standard 64-kilobyte address range. The window can access the entire extended address range of the 128-kilobyte flash memory block. The extended memory when used for data can also be accessed linearly using a linear address pointer and data access registers.

### 4.4.1 Features

Key features of the MMU module are:

- Memory Management Unit extends the HCS08 memory space
  - up to 4M bytes for program and data space
- Extended program space using paging scheme
  - PPAGE register used for page selection
  - fixed 16K byte memory window
  - architecture supports up to 256, 16K pages
- Extended data space using linear address pointer
  - up to 22-bit linear address pointer
  - linear address pointer and data register provided in direct page allows access of complete flash memory map using direct page instructions
  - optional auto increment of pointer when data accessed
  - supports an 2s complement addition/subtraction to address pointer without using any math instructions or memory resources
  - supports word accesses to any address specified by the linear address pointer when using LDHX, STHX instructions

### 4.4.2 Register Definition

The following figure shows a summary of MMU registers.

**Table 4-5. MMU Register Summary**

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| PPAGE | R | 0 | 0 | 0 | 0 | 0 | XA16 | XA15 | XA14 |
| | W | | | | | | | | |

Table 4-5. MMU Register Summary (continued)

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| LAP2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | LA16 |
| | W | | | | | | | | |
| LAP1 | R | LA15 | LA14 | LA13 | LA12 | LA11 | LA10 | LA9 | LA8 |
| | W | | | | | | | | |
| LAP0 | R | LA7 | LA6 | LA5 | LA4 | LA3 | LA2 | LA1 | LA0 |
| | W | | | | | | | | |
| LWP | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | W | | | | | | | | |
| LBP | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | W | | | | | | | | |
| LB | R | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| | W | | | | | | | | |
| LAPAB | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

### 4.4.2.1 Program Page Register (PPAGE)

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

PPAGE 0x0078

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

Figure 4-2. Program Page Register (PPAGE)

Table 4-6. Program Page Register Field Descriptions

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

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

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

**LAP2:LAP0** **0x0079-0x007B**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | LA16 |
| W | | | | | | | | |
| R<br>W | LA15 | LA14 | LA13 | LA12 | LA11 | LA10 | LA9 | LA8 |
| R<br>W | LA7 | LA6 | LA5 | LA4 | LA3 | LA2 | LA1 | LA0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-3. Linear Address Pointer Registers 2:0 (LAP2:LAP0)**

**Table 4-7. Linear Address Pointer Registers 2:0 Field Descriptions**

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

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

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

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

**LWP** **0x007C**

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

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

**Table 4-8. Linear Word Post Increment Register Field Descriptions**

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

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

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

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

**LBP** **0x007D**

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

**Figure 4-5. Linear Byte Post Increment Register (LBP)**

**Table 4-9. Linear Byte Post Increment Register Field Descriptions**

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

### 4.4.2.5 Linear Byte Register (LB)

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

**LB** **0x007E**

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

**Table 4-10. Linear Data Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>D7:D0 | **Linear Byte** — Reads of this register returns the data value pointed to by the linear address pointer, LAP2:LAP0. Writes to this register will write the data value to the memory location specified by the linear address pointer. Writes to this register are most commonly used when writing to the flash block(s) during programming. |

### 4.4.2.6 Linear Address Pointer Add Byte Register (LAPAB)

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

**LAPAB** **0x007F**

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

**Figure 4-6. Linear Address Pointer Add Byte Register (LAPAB)**

**Table 4-11. Linear Address Pointer Add Byte Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>D7:D0 | **Linear Address Pointer Add Byte** — The 2s complement value written to LAPAB will be added to contents of the linear address pointer register, LAP2:LAP0. Writing a value of 0x7f to LAPAB will increase LAP by 127, a value of 0xff will decrease LAP by 1, and a value of 0x80 will decrease LAP by 128. |

## 4.4.3 Functional Description

### 4.4.3.1 Memory Expansion

The HCS08 Core architecture limits the CPU addressable space available to 64K bytes. The Program Page (PPAGE) allows for integrating up to 4M byte of flash into the system by selecting one of the 16K byte blocks to be accessed through the Paging Window located at 0x8000-0xBFFF. The MMU module also provides a linear address pointer that allows extension of data access up to 4M bytes.

#### 4.4.3.1.1 Program Space

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

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

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

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

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

During the execution of a CALL instruction, the CPU:

- Stacks the return address.
- Pushes the current PPAGE value onto the stack.
- Writes the new instruction-supplied PPAGE value into the PPAGE register.
- Transfers control to the subroutine of the new instruction-supplied address.

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

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

RTC is similar to an RTS instruction.

The RTC instruction terminates subroutines invoked by a CALL instruction.

During the execution of an RTC instruction, the CPU:

- Pulls the old PPAGE value from the stack and loads it into the PPAGE register

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

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

#### 4.4.3.1.3 Data Space

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

To access data using the linear address pointer, the user would first setup the extended address in the 17-bit address pointer, LAP2:LAP0. Accessing one of the three linear data registers LB, LBP and LWP will access the extended memory location specified by LAP2:LAP0. The three linear data registers access the memory locations in the same way, however the LBP and LWP will also increment LAP2:LAP0. Accessing either the LBP or LWP registers allows a user program to read successive memory locations without re-writing the linear address pointer. Accessing LBP or LWP does the exact same function. However, because of the address mapping of the registers with LBP following LWP, a user can do word accesses in the extended address space using the LDHX or STHX instructions to access location LWP.

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

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

See Figure 4-1, on how the program PPAGE memory pages and the Linear Address Pointer are mapped to extended address space.

## 4.5 RAM

The HCS08 includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

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

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC13234/MC13237, 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 the reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale Semiconductor-provided equate file).

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

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See Section 4.6.5, “Flash Module Security”, for a detailed description of the security feature.

## 4.6 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.

The flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The flash module includes a memory controller that executes commands to modify flash memory contents.

For flash memory, an erased bit reads 1 and a programmed bit reads 0. It is not possible to read from a flash block while any command is executing on that specific flash block.

**CAUTION**

- A flash block address must be in the erased state before being programmed. Cumulative programming of bits within a flash block address is not allowed except for status field updates required in EEPROM emulation applications.
- The flash can be written and erased only with the CPU/Bus clocks programmed for maximum frequency, i.e., 32 MHz and 16 MHz respectively.

For more details about in-circuit and in-application programming, see the *HCS08 Family Reference Manual, Volume I*, Freescale Semiconductor document order number HCS08RMv1/D.

### 4.6.1 Features

Features of the flash memory include:

- Flash size - 131072 bytes (128 sectors of 1024 bytes each)
- Single power supply program and erase
- Automated program and erase algorithm
- Fast program and erase operation
- Burst program command for faster flash array program times
- Up to 10,000 program/erase cycles at typical voltage and temperature
- Flexible protection scheme to prevent accidental program or erase
- Security feature to prevent unauthorized access to the flash and RAM
- Auto power-down for low-frequency read accesses

### 4.6.2 Register Descriptions

The flash module contains a set of 16 control and status registers. Detailed descriptions of each register bit are provided in the following sections.

#### 4.6.2.1 Flash Options Register (FOPT and NVOPT)

The FSEC register holds all bits associated with the security of the MCU and flash module.

| | FOPT | | | | 0x1821 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | KEYEN | | 0 | 0 | 0 | 0 | SEC | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | F | F | 0 | 0 | 0 | 0 | F | F |

▨ = Unimplemented or Reserved

**Figure 4-7. Flash Options Register (FOPT)**

All bits in the FSEC register are readable but are not writable. To change the value in this register, erase and reprogram the NVSEC location in flash memory as usual and then issue an MCU reset.

The FSEC register is loaded from the flash location, NVSEC, during the reset sequence, indicated by F in Table 4-13.

**Table 4-12. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>KEYEN[1:0] | **Back Door Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of back door key access to the flash module as shown in Table 4-13. |
| 1:0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 4-13, Table 4-14 and Table 4-15. If the flash module is unsecured using back door key access, the SEC[1:0] bits are forced to the unsecured state. |

**Table 4-13. Flash KEYEN States**

| KEYEN[1:0] | Status of Back door Key Access |
|---|---|
| 00 | DISABLED |
| 01[1] | DISABLED |
| 10 | ENABLED |
| 11 | DISABLED |

[1] Preferred KEYEN state to disable Back door Key Access.

Table 4-14. Flash Security States for S08FTSR

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01[1] | SECURED |
| 10 | UNSECURED |
| 11 | SECURED |

[1] Preferred SEC state to set MCU to secured state.

Table 4-15. Flash Security States

| SEC[1:0] | Status of Security |
|---|---|
| 00 | SECURED |
| 01[1] | SECURED |
| 10 | UNSECURED |
| 11 | SECURED |

[1] Preferred SEC state to set MCU to secured state.

The security feature in the flash module is described in Section 4.6.5, “Flash Module Security””.

### 4.6.2.2 Flash Configuration Register (FCNFG)

The FCNFG register enables the flash interrupts and gates the security back door writes.

| | FCNFG | | | | 0x1823 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | KEYACC | 0 | 0 | 0 | 0 | 0 |
| W | ▨ | ▨ | | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

Figure 4-8. Flash Configuration Register (FCNFG)

NVM User Mode: KEYACC bit is readable and writable while all remaining bits read 0 and are not writable. KEYACC is writable only if KEYEN is set to the enabled state (see Section 4.6.2.1, “Flash Options Register (FOPT and NVOPT)”.

Table 4-16. FCNFG Field Descriptions

| Field | Description |
|---|---|
| 5<br>KEYACC | **Enable Security Key Writing**<br>0 Writes to the flash block are interpreted as the start of a command write sequence.<br>1 Writes to the flash block are interpreted as keys to open the back door. |

**NOTE**

Flash array reads are allowed while KEYACC is set. However, the MCU core will be held off for one bus clock cycle during key fetch from the flash array.

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

The FPROT register defines which flash sectors are protected against program or erase operations.

| FPROT | | | | 0x1824 | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | FPS | | | | | | | FPOPEN |
| Reset | F | F | F | F | F | F | F | F |

**Figure 4-9. Flash Protection Register (FPROT)**

The FPROT bits are readable and writable as long as the size of the protected flash memory is being increased. Any write to FPROT that attempts to decrease the size of the protected flash memory will be ignored and the PVIOL flag in the FSTAT register will not be set (see Section 4.6.2.4).

During the reset sequence, the FPROT register is loaded from the flash protection byte, NVPROT. To change the flash protection that will be loaded during the reset sequence, the flash sector containing NVPROT must be unprotected and erased (FPS=0x7F and FPOP=0 for S08FTSR, FPHS=0x1F and FPLS!=0x0 for S12SFTSR, FPS=0x7F and FPOP=0 for CFV1FTSR), then NVPROT can be reprogrammed.

Trying to alter data in any protected area in the flash memory will result in a protection violation error and the FPVIOL flag will be set in the FSTAT register. The mass erase of the flash array is not possible in NVM user mode if any of the flash sectors contained in the flash array are protected.

**Table 4-17. FPROT Field Descriptions for MC13234/MC13237**

| Field | Description |
|---|---|
| 7:1<br>FPS[6:0] | **Flash Protection Size** — With FPOPEN set, the FPS bits determine the size of the protected flash address range as shown in Table 4-18 |
| 0<br>FPOPEN | **Flash Protection Open**<br>0 Flash array fully protected.<br>1 Flash array protected address range determined by FPS bits. |

### Table 4-18. Flash Protection Address Range for MC13234/MC13237

| FPS[6:0] | FPOPEN | Protected Address Range Relative to Flash Array Base | | Protected Size |
|---|---|---|---|---|
| | | Flash Array 0 | Flash Array 1 | |
| - | 0 | 0x0_0000–0x0_FFFF | 0x1_0000–0x1_FFFF | 128 Kbytes |
| 0x00 | 1 | 0x0_0000–0x0_FFFF | 0x1_0400–0x1_FFFF | 127 Kbytes |
| 0x01 | | 0x0_0000–0x0_FFFF | 0x1_0800–0x1_FFFF | 126 Kbytes |
| 0x02 | | 0x0_0000–0x0_FFFF | 0x1_0C00–0x1_FFFF | 125 Kbytes |
| 0x03 | | 0x0_0000–0x0_FFFF | 0x1_1000–0x1_FFFF | 124 Kbytes |
| 0x04 | | 0x0_0000–0x0_FFFF | 0x1_1400–0x1_FFFF | 123 Kbytes |
| 0x05 | | 0x0_0000–0x0_FFFF | 0x1_1800–0x1_FFFF | 122 Kbytes |
| 0x06 | | 0x0_0000–0x0_FFFF | 0x1_1C00–0x1_FFFF | 121 Kbytes |
| ... | | ... | ... | ... |
| 0x37 | | 0x0_0000–0x0_FFFF | 0x1_E000–0x1_FFFF | 72 Kbytes |
| 0x38 | | 0x0_0000–0x0_FFFF | 0x1_E400–0x1_FFFF | 71 Kbytes |
| 0x39 | | 0x0_0000–0x0_FFFF | 0x1_E800–0x1_FFFF | 70 Kbytes |
| 0x3A | | 0x0_0000–0x0_FFFF | 0x1_EC00–0x1_FFFF | 69 Kbytes |
| 0x3B | | 0x0_0000–0x0_FFFF | 0x1_F000–0x1_FFFF | 68 Kbytes |
| 0x3C | | 0x0_0000–0x0_FFFF | 0x1_F400–0x1_FFFF | 67 Kbytes |
| 0x3D | | 0x0_0000–0x0_FFFF | 0x1_F800–0x1_FFFF | 66 Kbytes |
| 0x3E | | 0x0_0000–0x0_FFFF | 0x1_FC00–0x1_FFFF | 65 Kbytes |
| 0x3F | | 0x0_0000–0x0_FFFF | | 64 Kbytes |
| 0x40 | | 0x0_0400–0x0_FFFF | | 63 Kbytes |
| 0x41 | | 0x0_0800–0x0_FFFF | | 62 Kbytes |
| 0x42 | | 0x0_0C00–0x0_FFFF | | 61 Kbytes |
| 0x43 | | 0x0_1000–0x0_FFFF | | 60 Kbytes |
| 0x44 | | 0x0_1400–0x0_FFFF | | 59 Kbytes |
| 0x45 | | 0x0_1800–0x0_FFFF | | 58 Kbytes |
| 0x46 | | 0x0_1C00–0x0_FFFF | | 57 Kbytes |
| ... | | ... | | ... |
| 0x77 | | 0x0_E000–0x0_FFFF | | 8 Kbytes |
| 0x78 | | 0x0_E400–0x0_FFFF | | 7 Kbytes |
| 0x79 | | 0x0_E800–0x0_FFFF | | 6 Kbytes |
| 0x7A | | 0x0_EC00–0x0_FFFF | | 5 Kbytes |
| 0x7B | | 0x0_F000–0x0_FFFF | | 4 Kbytes |
| 0x7C | | 0x0_F400–0x0_FFFF | | 3 Kbytes |
| 0x7D | | 0x0_F800–0x0_FFFF | | 2 Kbytes |
| 0x7E | | 0x0_FC00–0x0_FFFF | | 1 Kbyte |
| 0x7F | | No Protection | | 0 Kbytes |

### 4.6.2.4 Flash Status Register (FSTAT)

The FSTAT register defines the operational status of the flash module. NVM User Mode: In normal mode, FCCF, FPVIOL, and FACCERR are readable and writable, FCCF and FBLANK are readable and not writable, remaining bits read 0 and are not writable.

**FSTAT (User Mode)** **0x1825**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| W | | ▨ | | | ▨ | ▨ | ▨ | ▨ |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 4-10. Flash Status Register (FSTAT)**

**Table 4-19. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FCBEF | **Flash Command Buffer Empty Flag** — The FCBEF flag indicates that the command buffer is empty so that a new command write sequence can be started when performing burst programming. Writing a 0 to the FCBEF flag has no effect on FCBEF. Writing a 0 to FCBEF after writing an aligned address to the flash array memory, but before FCBEF is cleared, will abort a command write sequence and cause the FACCERR flag to be set. Writing a 0 to FCBEF outside of a command write sequence will not set the FACCERR flag. The FCBEF flag is cleared by writing a 1 to FCBEF. The CBEIF flag is used together with the CBEIE bit in the FCNFG register to generate an interrupt request (see Figure 4-2).<br>0 Command buffers are full.<br>1 Command buffers are ready to accept a new command. |
| 6<br>FCCF | **Flash Command Complete Interrupt Flag** — The FCCF flag indicates that there are no more commands pending. The FCCF flag is cleared when FCBEF is cleared and sets automatically upon completion of all active and pending commands. The FCCF flag does not set when an active program command completes and a pending burst program command is fetched from the command buffer. Writing to the FCCF flag has no effect on FCCF. The CCIF flag is used together with the CCIE bit in the FCNFG register to generate an interrupt request (see Figure 4-2).<br>0 Command in progress.<br>1 All commands are completed. |
| 5<br>FPVIOL | **Flash Protection Violation Flag** —The FPVIOL flag indicates an attempt was made to program or erase an address in a protected area of the flash memory or flash IFR during a command write sequence. Writing a 0 to the FPVIOL flag has no effect on FPVIOL. The FPVIOL flag is cleared by writing a 1 to FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence.<br>0 No protection violation detected.<br>1 Protection violation has occurred. |
| 4<br>FACCERR | **Flash Access Error Flag** — The FACCERR flag indicates an illegal access has occurred to the flash memory or flash IFR caused by either a violation of the command write sequence (see Section 4.6.3.1.2, “Command Write Sequence”), issuing an illegal flash command (see Table 4-21), or the execution of a CPU STOP instruction while a command is executing (FCCF = 0). Writing a 0 to the FACCERR flag has no effect on FACCERR. The FACCERR flag is cleared by writing a 1 to FACCERR.While FACCERR is set, it is not possible to launch a command or start a command write sequence.<br>0 No access error detected.<br>1 Access error has occurred. |

Table 4-19. FSTAT Field Descriptions

| Field | Description |
|---|---|
| 2<br>FBLANK | **Flash Flag Indicating the Erase Verify Operation Status** — When the FCCF flag is set after completion of an erase verify command, the FBLANK flag indicates the result of the erase verify operation. The FBLANK flag is cleared by the flash module when FCBEF is cleared as part of a new valid command write sequence. Writing to the FBLANK flag has no effect on FBLANK.<br>0 flash block verified as not erased.<br>1 flash block verified as erased. |
| 1<br>FAIL | **Flag Indicating a Failed Flash Operation** — The FAIL flag will set if the erase verify operation fails (flash block verified as not erased) or other read operation fails, if an attempt to send an invalid command is made, or if the flash pulse register is improperly loaded. Writing a 0 to the FAIL flag has no effect on FAIL. The FAIL flag is cleared by writing a 1 to FAIL.<br>0 Flash operation completed without error or the flash module is not in NVM Special/Test mode.<br>1 Flash operation failed. |

## 4.6.2.5 Flash Command Register (FCMD)

The FCMD register is the flash command register.

FCMD 0x1826

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

▨ = Unimplemented or Reserved

Figure 4-11. Flash Command Register (FCMD)

All FCMD bits are readable and writable during a command write sequence while bit 7 reads 0 and is not writable.

Table 4-20. FCMD Field Descriptions

| Field | Description |
|---|---|
| 6:0<br>FCMD[6:0] | **Flash Command** — Valid flash commands in NVM user/normal mode are shown in Table 4-21. Writing any command other than those listed in Table 4-21 in NVM user/normal mode sets the FACCERR flag in the FSTAT register. flash commands available in NVM test mode are listed in Table 4-11. |

Table 4-21. Valid Flash Command List (NVM User Mode)

| FCMD[6:0] | NVM Command |
|---|---|
| 0x05 | Erase Verify |
| 0x20 | Program |
| 0x25 | Burst Program |
| 0x40 | Sector Erase |
| 0x41 | Mass Erase |

## 4.6.3 Functional Description

The following describes operation of the flash block.

### 4.6.3.1 Flash Command Operations

Flash command operations are used to execute program, erase, and erase verify algorithms described in this section. The program and erase algorithms are controlled by the flash memory controller whose time base, FCLK, is derived from the bus clock via a programmable divider.

The next sections describe:

1. Command write sequences to program, erase, and erase verify operations on the flash memory
2. Valid flash commands
3. Effects resulting from illegal flash command write sequences or aborting flash operations

#### 4.6.3.1.1 Flash Program and Erase Timing

The flash can be programmed on a byte or burst basis, and can be erased on a sector or mass (total flash) basis as controlled by the command mode. All program or erase operations must be done at the default or max CPU clock (32 MHz) / bus clock (16 MHz) rate.

**Table 4-22. Flash Program and Erase Times**

| Parameter | Cycles of 32 MHz Clock | Time |
|---|---|---|
| Byte program | 1280 | 40 µs |
| Byte program (burst) | 640 | 20 µs / byte[1] |
| Sector erase | 640320 | 20 ms |
| Mass erase | 643360 | 20.1 ms |

[1] Excluding start/end overhead

#### 4.6.3.1.2 Command Write Sequence

The flash command controller is used to supervise the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the FACCERR and FPVIOL flags in the FSTAT register must be clear and the FCBEF flag must be set (see Section 4.6.2.4, “Flash Status Register (FSTAT)”).

A command write sequence consists of three steps which must be strictly adhered to with writes to the flash module not permitted between the steps. However, flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the flash array memory.
2. Write a valid command to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the command.

After a command is launched, the completion of the command operation is indicated by the setting of the FCCF flag in the FSTAT register with an interrupt generated, if enabled. The FCCF flag will set upon completion of all active and buffered burst program commands.

A command write sequence can be aborted prior to clearing the CBEIF flag by writing a 0 to the CBEIF flag and will result in the ACCERR flag in the FSTAT register being set. The ACCERR flag in the FSTAT register must be cleared prior to starting a new command write sequence.

### 4.6.3.2 Flash Commands

Table 4-23 summarizes the valid flash commands along with the effects of the commands on the flash block.

**Table 4-23. Flash Command Description**

| FCMDB | NVM Command | Function on Flash Memory |
|---|---|---|
| 0x05 | Erase Verify | Verify all memory bytes in the flash array memory are erased.<br>If the flash array memory is erased, the FBLANK flag in the FSTAT register will set upon command completion. |
| 0x20 | Program | Program an address in the flash array. |
| 0x25 | Burst Program | Program an address in the flash array with the internal address incrementing after the program operation. |
| 0x40 | Sector Erase | Erase all memory bytes in a sector of the flash array. |
| 0x41 | Mass Erase | Erase all memory bytes in the flash array.<br>A mass erase of the full flash array is only possible when no protection is enabled prior to launching the command. |

**CAUTION**

A flash block address must be in the erased state before being programmed. Cumulative programming of bits within a flash block address is not allowed except for status field updates required in EEPROM emulation applications.

#### 4.6.3.2.1 Erase Verify Command

The erase verify operation will verify that a flash block is erased.

An example flow to execute the erase verify operation is shown in Figure 4-12. The erase verify command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the erase verify command. The address and data written will be ignored.
2. Write the erase verify command, 0x05, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the erase verify command.

After launching the erase verify command, the FCCF flag in the FSTAT register will set after the operation has completed. The number of bus cycles required to execute the erase verify operation is equal to the number of addresses in the flash array memory plus several bus cycles as measured from the time the

FCBEF flag is cleared until the FCCF flag is set. Upon completion of the erase verify operation, the FBLANK flag in the FSTAT register will be set if all addresses in the flash array memory are verified to be erased. If any address in the flash array memory is not erased, the erase verify operation will terminate and the FBLANK flag in the FSTAT register will remain clear.

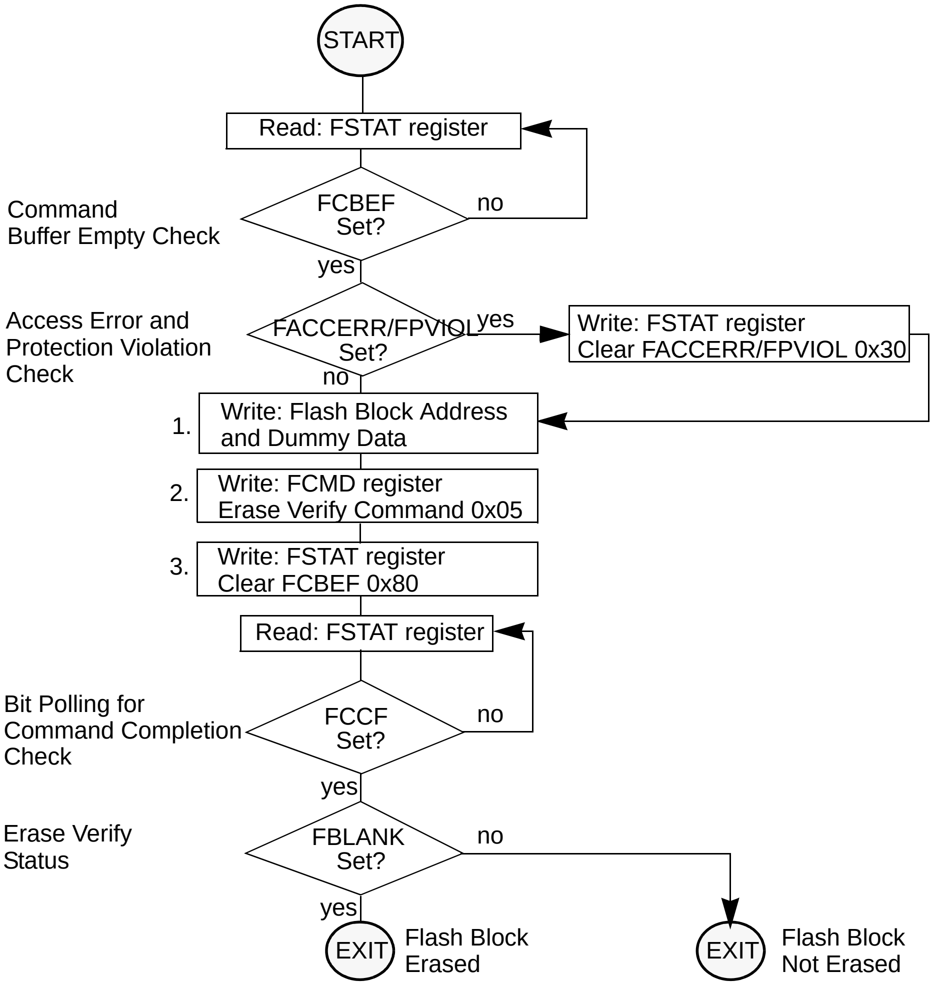


**Figure 4-12. Example Erase Verify Command Flow**

### 4.6.3.2.2 Program Command

The program operation will program a previously erased address in the flash memory using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 4-13. The program command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the program command. The data written will be programmed to the address written.
2. Write the program command, 0x20, to the FCMD register.

3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the program command.

If an address to be programmed is in a protected area of the flash block, the FPVIOL flag in the FSTAT register will set and the program command will not launch. After the program command has successfully launched, the FCCF flag in the FSTAT register will set after the program operation has completed.

Figure 4-13 shows the timing diagram for programming a single flash address using the program command.

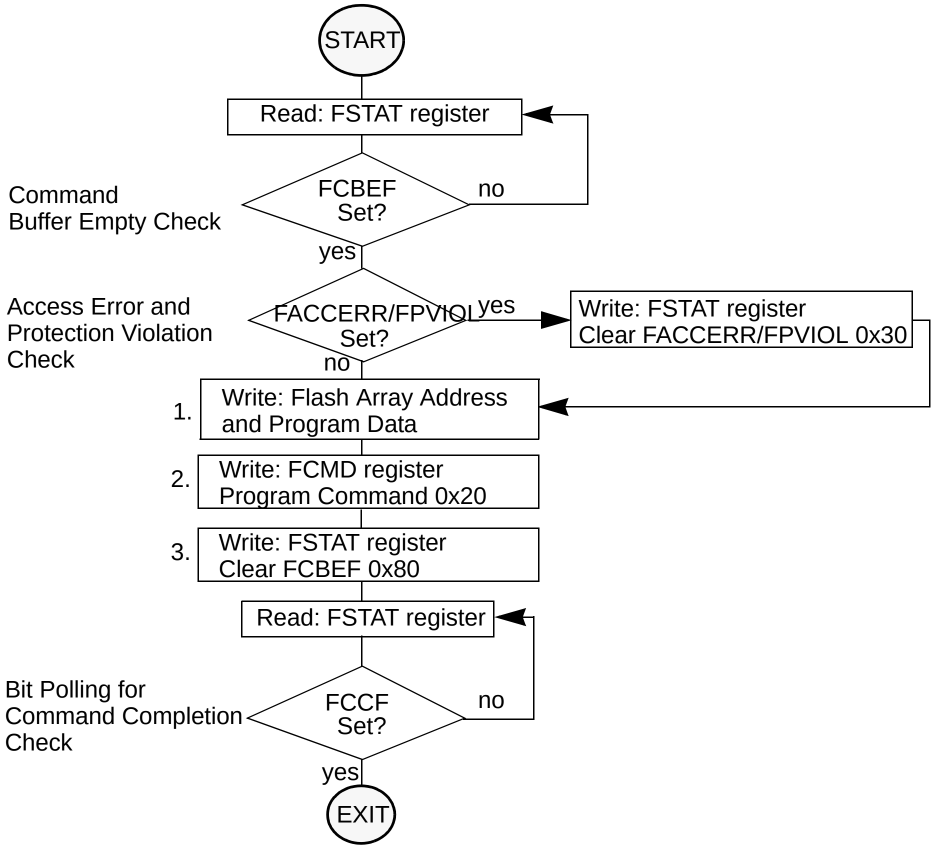


**Figure 4-13. Example Program Command Flow**

#### 4.6.3.2.3 Burst Program Command

The burst program operation will program previously erased data in the flash memory using an embedded algorithm.

While burst programming, two internal data registers operate as a buffer and a register (2-stage FIFO) so that a second burst programming command along with the necessary data can be stored to the buffers while the first burst programming command is still in progress. This pipe lined operation allows a time optimization when programming more than one consecutive address on a specific row in the flash array as the high voltage generation can be kept active in between two programming commands.

An example flow to execute the burst program operation is shown in Figure 4-14. The burst program command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the burst program command. The data written will be programmed to the address written.
2. Write the program burst command, 0x25, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the program burst command.
4. After the FCBEF flag in the FSTAT register returns to a 1 (interrupt generated, if enabled), repeat steps 1 through 3. The address written is ignored but increments internally.

The burst program procedure can be used to program the entire flash memory even while crossing row boundaries within the flash array. If data to be burst programmed falls within a protected area of the flash array, the FPVIOL flag in the FSTAT register will set and the burst program command will not launch. After the burst program command has successfully launched, the FCCF flag in the FSTAT register will set after the burst program operation has completed unless a new burst program command write sequence has been buffered. By executing a new burst program command write sequence on sequential addresses after the FCBEF flag in the FSTAT register has been set, greater than 50% faster programming time for the entire flash array can be effectively achieved when compared to using the basic program command.

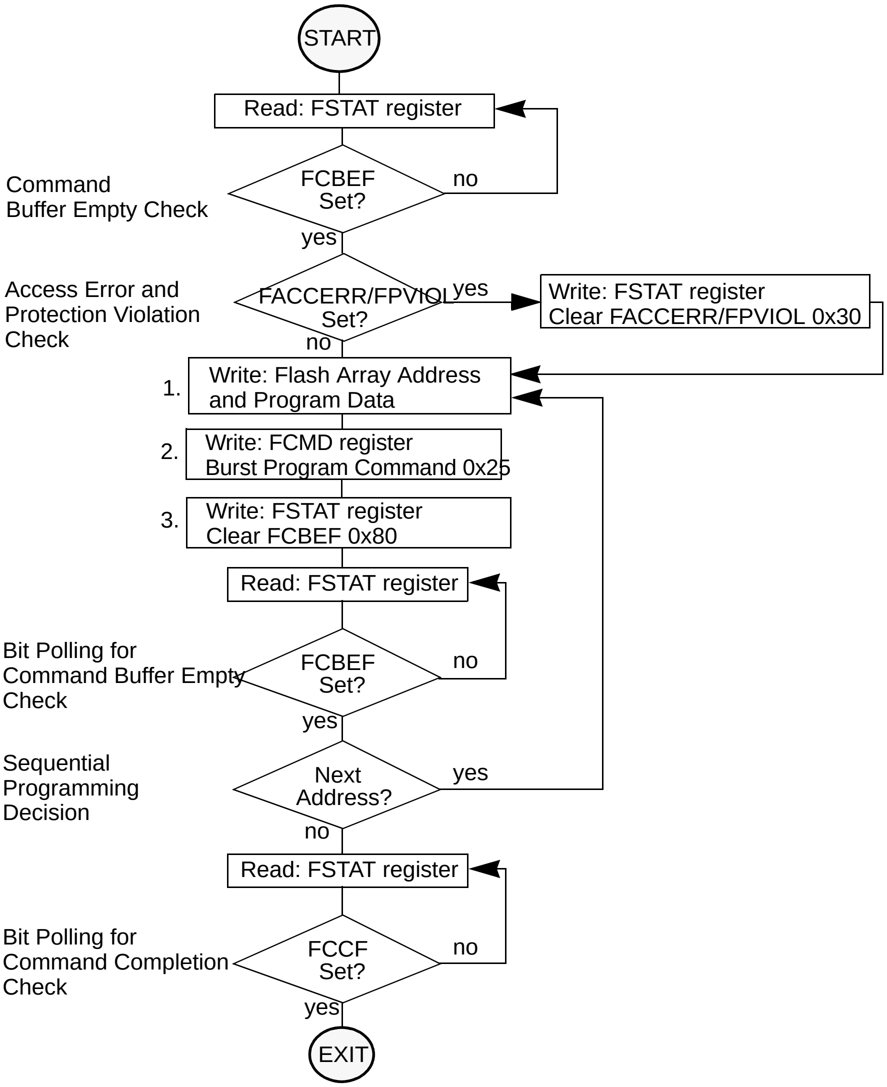


**Figure 4-14. Example Burst Program Command Flow**

### 4.6.3.2.4 Sector Erase Command

The sector erase operation will erase all addresses in a 1 Kbyte sector of flash memory using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 4-15. The sector erase command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the sector erase command. The flash address written determines the sector to be erased while global address bits [8:0] and the data written are ignored.
2. Write the sector erase command, 0x40, to the FCMD register.

3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the sector erase command.

If a flash sector to be erased is in a protected area of the flash block, the FPVIOL flag in the FSTAT register will set and the sector erase command will not launch. After the sector erase command has successfully launched, the FCCF flag in the FSTAT register will set after the sector erase operation has completed.

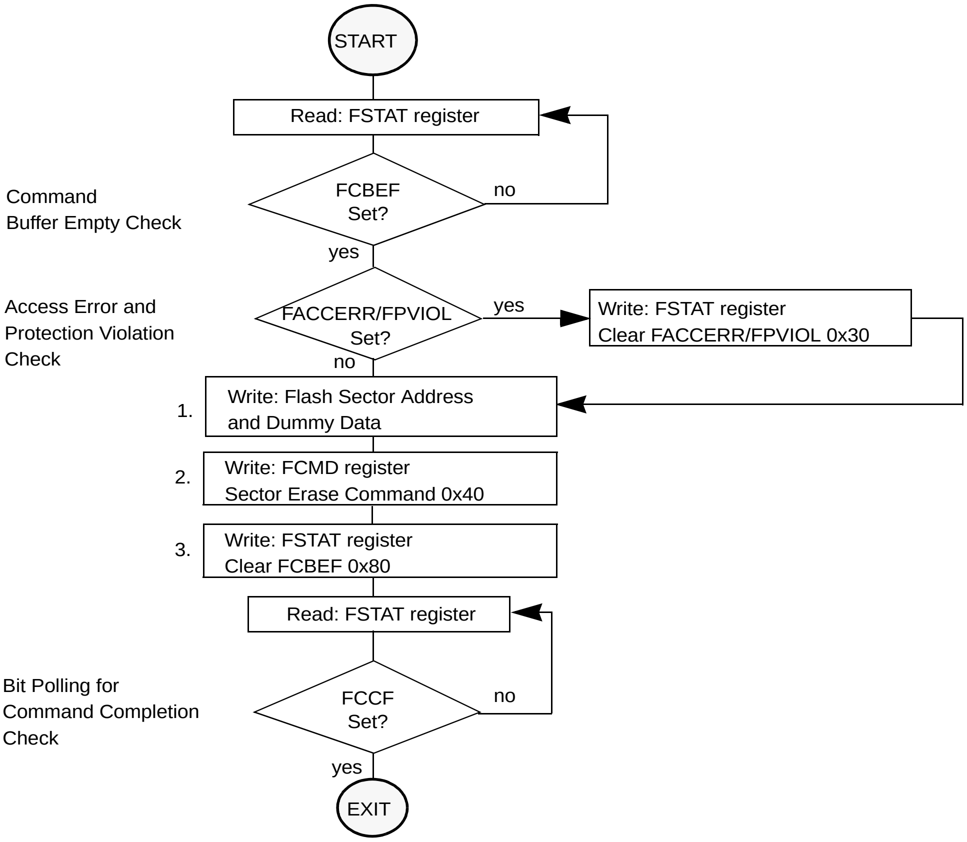


**Figure 4-15. Example Sector Erase Command Flow**

### 4.6.3.3 Illegal Flash Operations

#### 4.6.3.3.1 Flash Access Violations

The FACCERR flag will be set during the command write sequence if any of the following illegal steps are performed, causing the command write sequence to immediately abort:

1. Writing a byte or misaligned word to a valid flash address for S12SFTSR.
2. Writing a byte, word, or misaligned double word to a valid flash address for CFV1FTSR.
3. Writing to any flash register other than FCMD after writing to a flash address.
4. Writing to a second flash address in the same command write sequence.
5. Writing an invalid command to the FCMD register unless the address written was in a protected area of the flash array.
6. Writing a command other than burst program while FCBEF is set and FCCF is clear.

7. When security is enabled, writing a command other than mass erase to the FCMD register when the write originates from a non-secure memory location or from the background debug mode.
8. Writing to a flash address after writing to the FCMD register.
9. Writing to any flash register other than FSTAT (to clear FCBEF) after writing to the FCMD register.
10. Writing a 0 to the FCBEF flag in the FSTAT register to abort a command write sequence.

The FACCERR flag will also be set if the MCU enters stop mode while a program or erase operation is active. The operation is aborted immediately and, if burst programming, any pending burst program command is purged (see Section 4.6.4.2, “Stop Mode”).

The FACCERR flag will not be set if any flash register is read during a valid command write sequence.

If the flash memory is read during execution of an algorithm (FCCF = 0), the read operation will return invalid data in NVM user mode and the FACCERR flag will not be set.

If the FACCERR flag is set in the FSTAT register, the user must clear the FACCERR flag before starting another command write sequence (see Section 4.6.2.4, “Flash Status Register (FSTAT)”).

#### 4.6.3.3.2 Flash Protection Violations

The FPVIOL flag will be set after the command is written to the FCMD register during a command write sequence if any of the following illegal operations are attempted, causing the command write sequence to immediately abort:

1. Writing the program command if the address written in the command write sequence was in a protected area of the flash array.
2. Writing the sector erase command if the address written in the command write sequence was in a protected area of the flash array.
3. Writing the mass erase command while any flash protection is enabled.
4. Writing an invalid command if the address written in the command write sequence was in a protected area of the flash array. In NVM user and test modes, unimplemented commands started through array writes, with the given address pointing to a protected area, will always return a protection violation instead of a FAIL flag. For valid commands, protection is checked prior to command execution.

If the FPVIOL flag is set in the FSTAT register, the user must clear the FPVIOL flag before starting another command write sequence (see Section 4.6.2.4, “Flash Status Register (FSTAT)”).

## 4.6.4 Operating Modes

### 4.6.4.1 Wait Mode

If a command is active (FCCF = 0) when the MCU enters wait mode, the active command and any buffered command will be completed.

The flash module can recover the MCU from wait mode if the CBEIF and CCIF interrupts are enabled.

### 4.6.4.2 Stop Mode

If a command is active (FCCF = 0) when the MCU enters stop mode, the operation will be aborted and, if the operation is program or erase, the flash array data being programmed or erased may be corrupted and the FCCF and FACCERR flags will be set. If active, the high voltage circuitry to the flash array will immediately be switched off when entering stop mode. Upon exit from stop mode, the FCBEF flag is set and any buffered command will not be launched. The FACCERR flag must be cleared before starting a command write sequence (see Section 4.6.3.1.2, “Command Write Sequence”).

**NOTE**

As active commands are immediately aborted when the MCU enters stop mode, it is strongly recommended that the user does not use the STOP instruction during program or erase operations.

### 4.6.4.3 Background Debug Mode

In background debug mode (BDM), the FPROT register is writable. If the MCU is unsecured, then all flash commands listed in Table 4-23 can be executed. If the MCU is secured and is in special mode, only the erase verify and mass erase commands can be executed.

## 4.6.5 Flash Module Security

The HCS08 includes circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, flash and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

The flash module provides the necessary security information to the MCU. During each reset sequence, the flash module determines the security state of the MCU as defined in Section 4.6.2.1, “Flash Options Register (FOPT and NVOPT)””.

The contents of the flash security byte in NVSEC must be changed directly by programming the NVSEC location when the MCU is unsecured and the sector containing NVSEC is unprotected. If NVSEC is left in a secured state, any reset will cause the MCU to initialize into a secure operating mode.

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.

### 4.6.5.1 Unsecuring the MCU using Back Door Key Access

The MCU may be unsecured by using the back door key access feature which requires knowledge of the contents of the back door keys (NVBACKKEY through NVBACKKEY+7, see Table 4-4 for specific addresses). If the KEYEN[1:0] bits are in the enabled state (see Section 4.6.2.1, “Flash Options Register (FOPT and NVOPT)”) and the KEYACC bit is set, a write to a back door key address in the flash memory triggers a comparison between the written data and the back door key data stored in the flash memory. If all back door keys are written to the correct addresses in the correct order and the data matches the back door keys stored in the flash memory, the MCU will be unsecured. The data must be written to the back door keys sequentially. Values 0x0000 and 0xFFFF are not permitted as back door keys. While the KEYACC bit is set, reads of the flash memory will return invalid data.

The user code stored in the flash memory must have a method of receiving the back door keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 4.6.2.1, "Flash Options Register (FOPT and NVOPT)"), the MCU can be unsecured by the back door key access sequence described below:

1. Set the KEYACC bit in the flash configuration register (FCNFG).
2. Sequentially write the correct eight 8-bit bytes to the flash addresses containing the back door keys for S08FTSR.
3. Sequentially write the correct four 16-bit words to the flash addresses containing the back door keys for S12SFTSR.
4. Sequentially write the correct two 32-bit double words to the flash addresses containing the back door keys for CFV1FTSR.
5. Clear the KEYACC bit. Depending on the user code used to write the back door keys, a wait cycle (NOP) may be required before clearing the KEYACC bit.
6. If all data written match the back door keys, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 1:0.

The back door key access sequence is monitored by an internal security state machine. An illegal operation during the back door key access sequence will cause the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU will cause the security state machine to exit the lock state and allow a new back door key access sequence to be attempted. The following operations during the back door key access sequence will lock the security state machine:

1. If any of the keys written does not match the back door keys programmed in the flash array.
2. If the keys are written in the wrong sequence.
3. If more keys than are required are written.
4. If any of the keys written are all 0's or all 1's.
5. If the KEYACC bit does not remain set while the keys are written.
6. If any of the keys are written on successive MCU clock cycles.
7. Executing a STOP instruction while the KEYACC bit is set.

After the back door keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the back door keys by programming the associated addresses in NVBACKKEY through NVBACKKEY+7.

The security as defined in the flash security byte is not changed by using the back door key access sequence to unsecure. The stored back door keys are unaffected by the back door key access sequence. After the next reset of the MCU, the security state of the flash module is determined by the flash security byte. The back door key access sequence has no effect on the program and erase protections defined in the flash protection register (FPROT).

It is not possible to unsecure the MCU in special mode by using the back door key access sequence in background debug mode (BDM) as the BDM will not allow flash array writes to the flash module.

### 4.6.6 Resets

If a reset occurs while any flash command is in progress, that command will be immediately aborted. The state of the flash array address being programmed or the sector/block being erased is not guaranteed.

# Chapter 5
# System Management and Control

## 5.1 Introduction

This chapter brings together and describes the various elements that manage and control operation of the MC13234/MC13237. Topics include reset, clock resources and control, control timer resources, interrupts, modes of operation, power management, and control register descriptions.

## 5.2 Management Features

- System reset and available software initiated system reset
- Power management of internal regulated voltage sources
- Real-Time Clock (RTC)
- Computer Operating Properly (COP) timer
- Interrupt control and management
- Mode management
  - Active background mode for code development
  - Run mode — CPU running at full speed and the internal supply is fully regulated, radio optionally active. CPU clock can be run at divided values of: 1, 2, 4, 8, 16, and 64.
  - LPRun mode — CPU clock is set to 500 kHz and peripheral clock set (bus clock) to 250 kHz 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, radio optionally active.
  - LPWait mode — CPU shuts down to conserve power; peripheral clocks are restricted to 250 kHz and the internal voltage regulator is in standby.
  - Stop3 mode — System clocks are stopped and voltage regulator is in standby. All internal circuits are powered for fast recovery, (full regulation set before entering stop).
- Wake-up mode management
  - Chip is gracefully powered up.
  - Clocks are automatically turned on.
  - Wake-up available by programmable RTC timer.
  - Wake-up available by external interrupts from KBI pads, IRQ, or UART RX.
- Management of 1 kHz ring oscillator and optional 32.768 kHz oscillator
- Management of 32 MHz reference oscillator
- Management of MCU core and peripheral clocks

## 5.3 System Reset

The MC13234/MC13237 provides complete management of system reset and recovery/initialization from 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
- The program counter is loaded from the reset vector (0xFFFE:0xFFFF) so that at release of reset the CPU starts executing instructions from this address
- On-chip peripheral modules are disabled
- I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled.
- The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to 0x00FF at reset.

The sources of reset for the MC13234/MC13237 can be loosely divided into the power-on reset circuit (POR) and all others:

- Power-on reset
  - The POR is only activated or cycled by the main VDD supply. If a reset is initiated by any other source than a power cycle or the low-voltage detect, the POR is not involved.
  - Factory test mode is enabled via POR and is not desired for normal operation. Signal PTA2 must be held low during release of POR to allow the device to enter normal operation.
  - The low-voltage detect reset is enabled out of POR and the power supply voltage must rise and exceed the LVD rising-edge threshold or the device will be held in reset. After the rising edge threshold is exceeded, the reset is released.
- Other reset sources
  - External asynchronous $\overline{\text{RESET}}$ pin - holding RESET asserted for an extended period affects the reference oscillator, see Section 5.4.1, “Clock Sources”.
  - Computer operating properly (COP) timer
  - Illegal opcode detect (ILOP)
  - Illegal Address detect (ILAP)
  - 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).

Reset or a partial reset is also affected by exit from low power modes.

## 5.4 System Clocks

This section provides a brief overview of the system clocks.

### 5.4.1 Clock Sources

The HCS08 has 3 possible clock sources, i.e., the 32 MHz reference oscillator, an on board self-contained 1 kHz ring oscillator, and an optional 32.768 kHz crystal oscillator.

- Reference oscillator - is the primary clock source for the device
  - — 32 MHz is the required frequency.
  - — A crystal is used with the onboard amplifier.
  - — Frequency accuracy of +/-40 ppm or better for IEEE 802.15.4 compatibility must be maintained over all conditions.
  - — See Chapter 3, “System Considerations” for more information about this oscillator.
  - — Trim the crystal oscillator frequency accuracy by changing the on board load capacitance.
  - — The reference oscillator can be disabled when the hardware $\overline{\text{RESET}}$ pin is enabled (active low) if $\overline{\text{RESET}}$ is held active for more than 69 bus clock cycles then the oscillator will be disabled.
- 1 kHz ring oscillator - is the default oscillator source for the Real Time Clock (RTC) which can be used as a wake-up source and for the COP timer. The 1 kHz oscillator is disabled only if not selected for use by the RTC or the COP.
- Optional 32.768 kHz crystal oscillator - is available with the addition of an external 32.768 kHz crystal (see Chapter 3, “System Considerations”). This oscillator typically replaces the use of the ring oscillator if a low power, high accuracy source is desired for wake-up and the RTC.

### 5.4.2 Main System Clock Distribution

Figure 5-1 shows a simplified diagram of the HCS08 clock distribution.

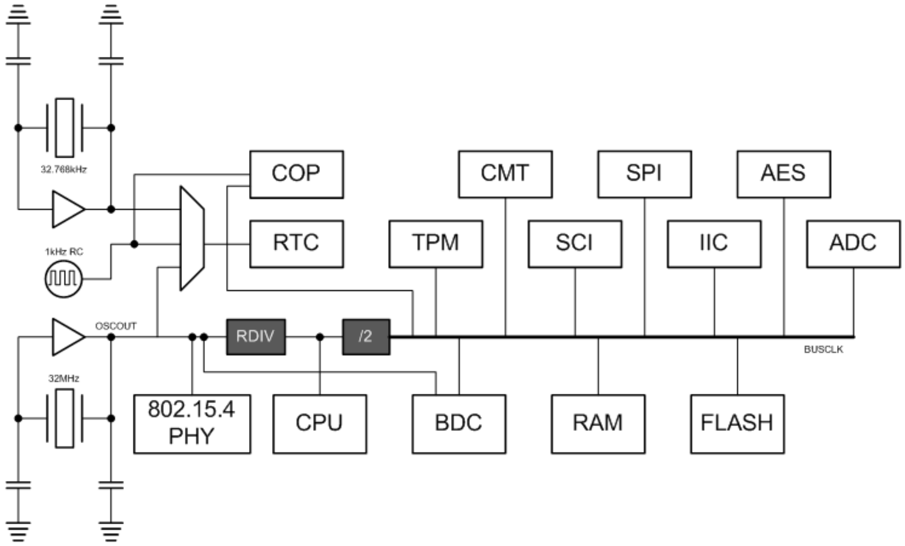


**Figure 5-1. MC13234/MC13237 Simplified Clock Distribution**

### 5.4.3 Clock Distribution Network Features

- The primary reference oscillator is 32 MHz
- The 32 MHz reference drives the transceiver 802.15.4 PHY block directly.
- The CPU and bus clocks are all derived from the reference oscillator and a programmable divider (prescaler)
  - The CPU clock is determined by the prescaler divide ratio as set by the RDIV[2:0] field of the System Oscillator Management and Control Register 1 (SOMC1 )- divide ratios are programmable from 1 to 64 (default = 1). Lower core clock rates reduce power, but also reduce performance.
  - The bus clock is always the CPU clock divided-by-2. All MCU peripherals are driven by the bus clock.
- The Real Time Clock (RTC) source clock is selected from the reference oscillator, the onboard 1 kHz RC oscillator or the optional 32.768 kHz crystal oscillator
- The COP is optionally driven from the onboard 1 kHz RC oscillator or the bus clock.

### 5.4.4 Managing Clock Resources

The HCS08 provides the following for managing the clock resources:

1. System Oscillator Management and Control Register 1, see Section 5.8.10, “System Oscillator Management and Control Register 1 (SOMC1)”- this register sets the CPU clock prescale integer, enables the 32.768 kHz oscillator, controls the Stop mode oscillator, and enables the optional clock out pin for the 32.768 kHz oscillator.
2. System Oscillator Management and Control Register 2, see Section 5.8.11, “System Oscillator Management and Control Register 2 (SOMC2)”- this register controls the trim capacitance that adjusts the reference oscillator frequency.
3. System Clock Gating Control Registers, see Section 5.8.12, “System Clock Gating Control 1 Register (SCGC1)” and Section 5.8.13, “System Clock Gating Control 2 Register (SCGC2)” - these registers provide control bits to disable the bus clock to the various peripheral modules. For power sensitive applications, the current can be minimized by disabling the bus clock to non-used circuitry.
4. The 1 kHz RC oscillator is disabled if not selected as a source for either the RTC or the COP.

## 5.5 Management of Timer Resources

To help control operational modes and provide wake-up services to the HCS08, there are the Real-Time Counter (RTC) and COP blocks.

### 5.5.1 RTC Timer Resource

The RTC provides a resource for low power wake-up and time “ticks” applicable for software use. The RTC is described in a separate Chapter 10, “Real-Time Counter (RTC)”.

## 5.5.2 Computer Operating Properly (COP) Timer

The COP is a watchdog timer for the MCU and is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset caused by 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.

**NOTE**

> The COP is enabled by default when exiting reset. If COP operation is undesired, it should be disabled as part of the device initialization.

Features of the COP include:

- Clock sources include bus clock and the 1 kHz RC clock
- Two time-out periods are available for each clock source
- The COP is controlled the following bits -
  - COPE - COP enable bit, Register SOPT1
  - COPT - COP time-out bit, Register SOPT1
  - COPCLKS - COP clock source select bit, Register SOPT2
- The SRS Register (see Section 5.8.1, “Interrupt Pin Request Status and Control Register (IRQSC)”) also relates to COP operation
  - Writing the SRS register address resets the COP timer
  - If the COP times-out causing a system reset, the status is reported in the SPS COP status bit.

The COPE bit is the primary enable for the COP (see Section 5.8.4, “System Options Register 1 (SOPT1)”):

- After any reset, the COPE bit becomes set in SOPT1 enabling the COP watchdog.
- If the COP watchdog is not used in an application, it can be disabled by clearing COPE.

The COPCLKS bit in SOPT2 (see Section 5.8.5, “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 5-1 summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz clock source and the associated long time-out ($2^8$ cycles).

Table 5-1. COP Configuration Options

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

### NOTE

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

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.

### NOTE

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.

With regard to operation during Stop mode:

- 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. The COP counter begins from zero after the MCU exits stop mode.

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

## 5.6 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 status flag will become set. The CPU will not respond unless the local interrupt enable is set to enable the interrupt and the I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which prevents all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

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

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

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

**NOTE**

In order for the ISR to be available in the memory map regardless of the PPAGE value, ISRs should be located in Pages 0, 1, or 3.

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

**NOTE**

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

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

### 5.6.1 Interrupt Stack Frame

Figure 5-2 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.

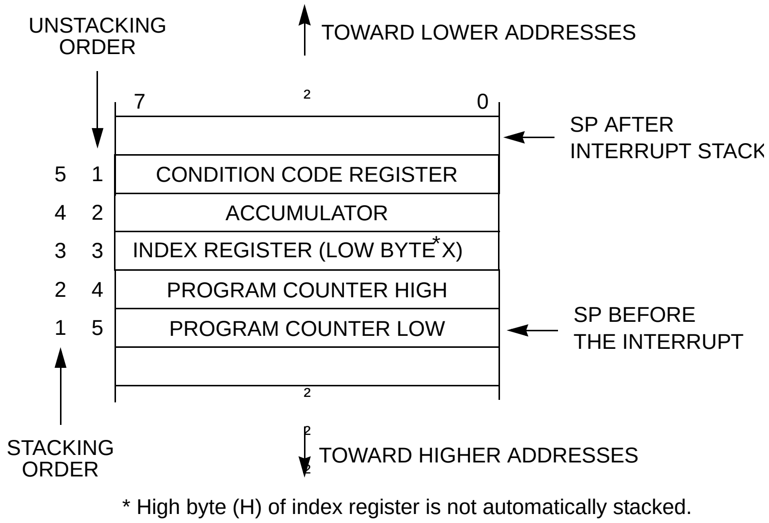


**Figure 5-2. Interrupt Stack Frame**

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

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

## 5.6.2 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. The external IRQ pin is shared with I/O PTA3.

### 5.6.2.1 Pin Configuration Options

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

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

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

**NOTE**

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

**NOTE**

The voltage measured on the internally pulled up $\overline{\text{RESET}}$ pin will not be pulled to $V_{DD}$. The internal gates connected to this pin are pulled to $V_{DD}$. The $\overline{\text{RESET}}$ pullup should not be used to pullup components external to the MCU.

#### 5.6.2.2 Edge and Level Sensitivity

The IRQMOD control bit configures 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 de-asserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

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

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

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

**Table 5-2. Vector Summary**

| Vector Priority | Vector No | Address (High/Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|---|
| Lowest | 26[1] | 0xFFCA/0xFFCB | Vadc | ADC | COCO | AIEN | Analog-to-digital converter |
| | 25 | 0xFFCC/0xFFCD | Vrtc | RTC | RTIF | RTIE | Real-time clock |
| | 24 | 0xFFCE/0xFFCF | Vkeyboard1 | KBI1 | KBF1 | KBIE1 | Keyboard1 x pins |
| | 23[2] | 0xFFD0:0xFFD1 | Vkeyboard2 | KBI2 | KBF2 | KBIE2 | Keyboard2 x pins |
| | 22 | 0xFFD2/0xFFD3 | Viicx | IIC | IICIS | IICIE | IICx control |
| | 21 | 0xFFD4/0xFFD5 | Vcmt | CMT | EOCF | EOCIE | CMT |
| | 20 | 0xFFD6/0xFFD7 | Vscitx | SCI | TDRE,TC | TIE,TCIE | SCI transmit |

**Table 5-2. Vector Summary**

| Vector Priority | Vector No | Address (High/Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|---|
| | 19 | 0xFFD8/0xFFD9 | Vscirx | SCI | IDLE,<br>LBKDIF,<br>RDRF,<br>RXEDGIF | ILIE, LBKDIE, RIE,<br>RXEDGIE | SCI receive |
| | 18 | 0xFFDA/0xFFDB | Vscierr | SCI | OR, NF,<br>FE, PF | ORIE, NFIE,<br>FEIE, PFIE | SCI error |
| | 17 | 0xFFDC/0xFFDD | Vspi | SPI | SPIF, MODF,<br>SPTEF | SPIE SPTIE | SPI |
| | 16 | 0xFFDE/0xFFDF | Vtpm4ovf | TPM4 | TOF | TOIE | TPM4 overflow |
| | 15 | 0xFFE0/0xFFE1 | Vtpm4ch0 | TPM4 | CH0F | CH0IE | TPM4 channel 0 |
| | 14 | 0xFFE2/0xFFE3 | Vtpm3ovf | TPM3 | TOF | TOIE | TPM3 overflow |
| | 13 | 0xFFE4/0xFFE5 | Vtpm3ch0 | TPM3 | CH0F | CH0IE | TPM3 channel 0 |
| | 12 | 0xFFE6/0xFFE7 | Vtpm2ovf | TPM2 | TOF | TOIE | TPM2 overflow |
| | 11 | 0xFFE8/0xFFE9 | Vtpm2ch0 | TPM2 | CH0F | CH0IE | TPM2 channel 0 |
| | 10 | 0xFFEA/0xFFEB | Vtpm1ovf | TPM1 | TOF | TOIE | TPM1 overflow |
| | 9 | 0xFFEC/0xFFED | Vtpm1ch0 | TPM1 | CH0F | CH0IE | TPM1 channel 0 |
| | 8 | 0xFFEE/0xFFEF | | AES128 | AES | AES_MASK | AES 128 Cipher |
| | 7 | 0xFFF0/0xFFF1 | | 802.15.4 | 802.15.4<br>RX DMA | RX_WMRK_MSK | 802.15.4 PHY<br>RX WTR MRK |
| | 6 | 0xFFF2/0xFFF3 | | 802.15.4 | 802.15.4<br>Event<br>Timers | TMR4MSK<br>TMR3MSK<br>TMR2MSK<br>TMR1MSK | 802.15.4 PHY TIMERS |
| | 5 | 0xFFF4/0xFFF5 | | 802.15.4 | 802.15.4<br>TX | TXMSK<br>LO1_UNLOCK_MSK<br>SEQMSK | 802.15.4 PHY TX |
| | 4 | 0xFFF6/0xFFF7 | | 802.15.4 | 802.15.4<br>RX/CCA | RX_RCVD_MSK<br>LO1_UNLOCK_MSK<br>CCAMSK<br>FLITERFAIL_MSK<br>SEQMSK | 802.15.4 PHY RX |
| | 3 | 0xFFF8/0xFFF9 | Vlvd | System<br>control | LVDF, LVWF | LVDIE, LVWIE | Low-voltage detect,<br>Low-voltage warning |
| | 2 | 0xFFFA/0xFFFB | Virq | IRQ | IRQF | IRQIE | IRQ pin |

Table 5-2. Vector Summary

| Vector Priority | Vector No | Address (High/Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|---|
| | 1 | 0xFFFC/0xFFFD | Vswi | Core | SWI Instruction | — | Software interrupt |
| Highest | 0 | 0xFFFE/0xFFFF | Vreset | System control | COP,<br>LVD,<br>$\overline{\text{RESET}}$ pin,<br>Illegal opcode, | COPE<br>LVDRE<br>—<br>— | Watchdog timer<br>Low-voltage detect<br>External pin<br>Illegal opcode |

[1] ADC interrupt (vector 26) is available only in MC13237.

[2] KBI2 interrupt (vector 23) is available only in MC13234.

## 5.7 Modes of Operation

The HCS08 provides the following modes of operation:

- Active background mode for code development
- Run mode — CPU running at full speed and the internal supply is fully regulated, radio optionally active.
- LPRun mode — CPU clock is set to 500 kHz and peripheral clocks (bus clock) to 250 kHz and the internal voltage regulator is in standby, radio optionally active.
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained, radio optionally active.
- LPWait mode — CPU shuts down to conserve power; peripheral clocks are set to 250 kHz and the internal voltage regulator is in standby, radio optionally active.
- Stop3 mode — System clocks are stopped and voltage regulator is in standby. All internal circuits are powered for fast recovery, radio optionally active, (full regulation set before entering stop).

### 5.7.1 Run Mode

This is the normal operating mode for the HCS08. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFE–0xFFFF after reset. The 802.15.4 transceiver is available for any transmit or receive operations. The clock CPU frequency is controlled through the RDIV register to yield: 32, 16, 8, 4, 2, 1, and 0.5 MHz.

### 5.7.2 Low Power Run Mode (LPRun)

In the low power run mode, the bus clock is modified such that the 32MHz reference clock is divided by 64. In this state, the 802.15.4 transceiver is available for any transmit or receive operations. 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. The digital regulator is held in standby mode.

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

- The LVDE or LVDSE bit in SPMSC1 register must be clear. LVD and LVW will automatically be disabled.
- Flash programming/erasing is not allowed.
- The MCU cannot be in active background mode.

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

To exit LPRun and 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 when the regulator has returned to full regulation mode.
- When LPRS is cleared to ‘0’, the regulator is in full regulation mode and the MCU has automatically returned to full CPU clock speed (RDIV = 0x0). If the application was using a lower CPU clock rate before entering LPRun, this condition must be re-initiated by writing RDIV field.

#### 5.7.2.1 Interrupts in Low Power Run Mode

Low power run mode provides the option to return to full regulation if any interrupt occurs. This is done by setting the LPWUI bit in the SPMSC2 register. The bus clock returns to full speed immediately in the interrupt service routine.

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

#### 5.7.2.2 Resets in Low Power Run Mode

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

### 5.7.3 Active Background Mode

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

Active background mode is entered in any of six ways:

- When the BKGD/MS pin is low during POR
- When the BKGD/MS pin is low immediately after issuing a background debug force reset (see Section 5.8.3, “System Background Debug Force Reset Register (SBDFR)”)
- When a BACKGROUND command is received through the BKGD/MS pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

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

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD 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 HCS08 is shipped from the Freescale Semiconductor factory, the flash program memory is erased by default unless specifically noted, so there is no program that could be executed in run mode until the flash memory is initially programmed. The active background mode can also be used to erase and reprogram the flash memory after it has been previously programmed.

For additional information about the active background mode, see Chapter 18, “Development Support”.

### 5.7.4 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 802.15.4 transceiver can be active during this mode; completing a transmit or receive operation initiated during Run mode.The I bit in CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

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

### 5.7.5 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. The 802.15.4 transceiver can be active during this mode; completing a transmit or receive operation initiated during LPRun mode. In the low power wait mode, the on-chip voltage regulator remains in its active state as in the low power run mode and the bus clock is 250 kHz. 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.

### 5.7.5.1 Interrupts in Low Power Wait Mode

If the LPWUI bit is set, then the voltage regulator will return to full regulation. LPWait mode is exited to Run mode. The LPR and LPRS bits will be cleared and the MCU will be clock from the XOSC2 oscillator (32MHz).

If the LPWUI bit is clear an interrupt will return the device to low power run (LPRun) mode.

### 5.7.5.2 Resets in Low Power Wait Mode

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

## 5.7.6 Stop3 Mode

Stop3 is entered upon execution of a STOP instruction when the STOPE bit in the System Option 1 Register (SOPT1) is set. In Stop3 mode, the CPU and bus clocks are halted. The regulator can be configured for standby or active mode. The 32 kHz and 32 MHz crystal oscillators can be configured as active or in standby in Stop3 mode.

If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU will not enter Stop3 mode and an illegal opcode reset is forced. Stop3 mode is selected by setting the appropriate bits in the Section 5.8.8, “System Power Management Status & Control 2 (SPMSC2)”.

Table 5-3 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 5-3. 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 to 0 | | 0 | Stop3 |

[1] ENBDM is located in the BDCSCR which is only accessible through BDC commands.

[2] When in Stop3 mode with BDM enabled, The $S_{IDD}$ will be near $R_{IDD}$ levels because internal clocks are enabled.

### 5.7.6.1 Description

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in Table 5-3. 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, IRQ, SCI or the KBI.

If Stop3 is exited by means of the $\overline{\text{RESET}}$ pin, then the MCU is reset and operation will resume after taking the reset vector. Exit by means of one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

The 802.15.4 transceiver will be off in this mode. In Stop3 mode the digital regulator will be in standby mode.

### 5.7.6.2 Active BDM Enabled in Stop3 Mode

The advantage of Stop3 mode is the low current consumption of the device. While MC13234/MC13237 is in Stop3 mode, circuit blocks are turned off to reduce consumption. BDM (Background Debug Module) is also disabled. For debug purpose, BDM can be enabled to let the external BDM module communicate to MC13234/MC13237. To enable BDM in Stop3 mode, the following bits should be set prior the STOP command is issued:

SOPT1 bit 1: BKGDPE

SCGC2 bit 6: DBG

### 5.7.6.3 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in Stop3 mode (LVDE and LVDSE bits in SPMSC1 both set), the voltage regulator remains active during stop mode.

### 5.7.6.4 Stop3 Mode in Low Power Run Mode

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

### 5.7.6.5 Wake-up Time from Stop3 Mode

The dominating factor for Stop3 wake-up time is the reference oscillator start-up time. In Stop3 mode, the reference oscillator can optionally be running. If the reference oscillator is off in Stop3 mode, the startup time is specified at 800 µs typical. This will determine the wakeup time from Stop3 mode. If the reference oscillator is running in Stop3 mode, the wake up time will be significantly less.

When the reference oscillator must start, it is specified at 800 µs typical startup time and this will determine the wake-up. It the reference oscillator is already running, the wake-up time will be significantly less.

### 5.7.6.6 Stop3 and LPRun Mode Transition

In both MC13234 and MC13237, transition between Stop3 and LPRUN mode needs extra steps when the LPWUI bit is used. In Run mode, going to LPRun mode (LPR = 1), if LPWUI is set before entering Stop3. The device will exit Stop3 with full regulation (Run mode) when an interrupt is received. However, it will not enter LPRun mode even LPR is set. Until the device enters Stop3 mode again. Then it will enter LPRun again with LPR = 1. The following diagram illustrates this transition.

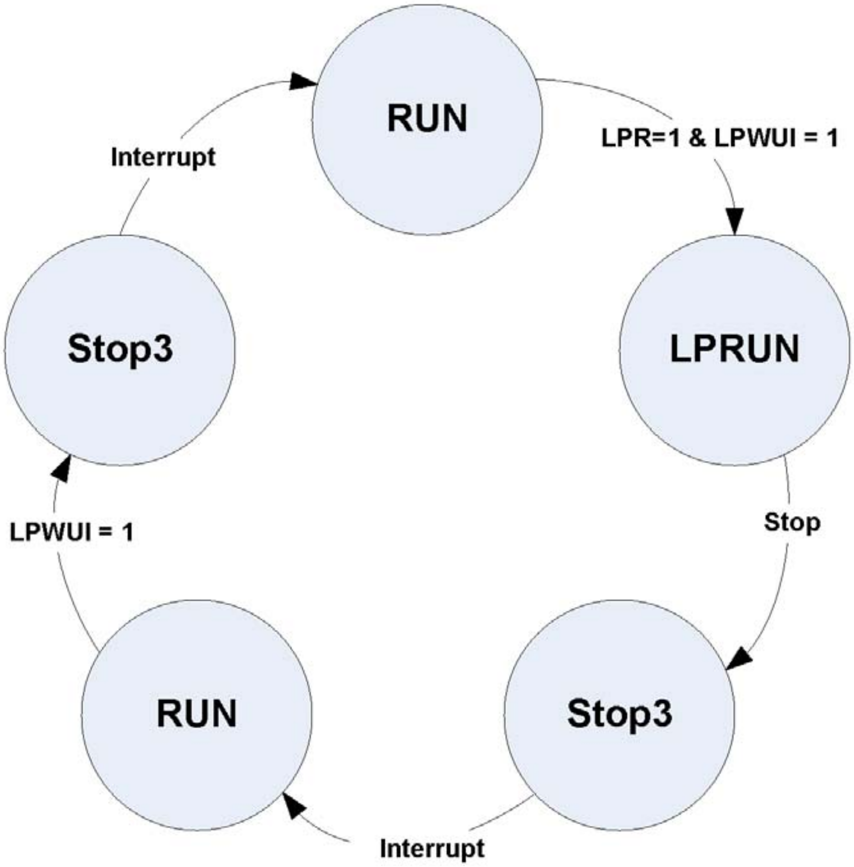


Figure 5-3. LPRun-to-Stop3 Diagram

## 5.7.7 Mode Selection

Several control signals are used to determine the current operating mode of the device. Table 5-4 shows the conditions for each of the device’s operating modes.

**Table 5-4. Power Mode Selections**

| Mode of Operation | XXCSR BDM | SPMSC1 PMC | | SPMSC2 PMC | | CPU & Peripheral CLKs | Affects on Sub-System | |
|---|---|---|---|---|---|---|---|---|
| | ENBDM [1] | LVDE | LVDSE | LPR | PPDC | | BDM Clock | Voltage Regulator |
| Run mode | 0 | x | x | 0 | x | on. | off | on |
| | | 1 | 1 | 1 | | | | |
| | 1 | x | x | x | | | on | |
| LPRun mode | 0 | 0 | x | 1 | 0 | 32MHz osc clock divided by 64 | off | standby |
| | | 1 | 0 | | | | | |
| Wait mode - (Assumes WAIT instruction executed.) | 0 | x | x | 0 | x | CPU clock is off; peripheral clocks on. | off | on |
| | | 1 | 1 | 1 | | | | |
| | 1 | x | x | x | | | on | |
| LPWait mode - (Assumes WAIT instruction executed.) | 0 | 0 | x | 1 | 0 | CPU clock is off; peripheral clocks at low speed (250 kHz). | off | standby |
| | | 1 | 0 | | | | | |
| Stop3 - (Assumes STOPE bit is set and STOP instruction executed.) | 0 | 0 | x | x | 0 | CPU and peripheral clocks are off. 32 MHz Osc and 32 kHz Osc optionally on | off | standby |
| | 0 | 1 | 0 | x | 0 | | off | |
| | 0 | 1 | 1 | x | x | | off | on - stop currents will be increased |
| | 1 | x | x | x | x | | on | |

[1] Design Interlock: If STOP was issued from LPRun, then Stop3 will be entered

**Table 5-5. MCU Allowable Power Mode Transitions for MC13234/MC13237**

| Mode | Digital Regulator State |
|---|---|
| Run | Full on |
| Wait | Full on |
| LPRun | Standby |
| LPWait | Standby |
| Stop3 | Standby |

Figure 5-4 shows mode state transitions allowed between the legal states.

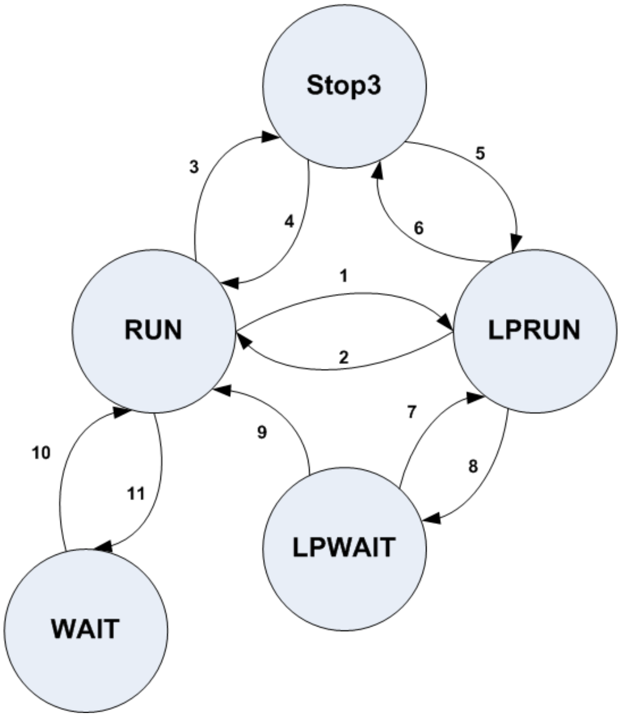


Figure 5-4. Stop Mode State Diagram

Table 5-6 defines triggers for the various state transitions shown in Figure 5-4.

Table 5-6. Triggers for Transitions.

| Transition # | From | To | Trigger |
|---|---|---|---|
| 1 | Run | LPRun | Configure settings shown in Table 5-3, **switch LPR=1 last** |
| 2 | LPRun | Run | Clear LPR |
| | | | **Interrupt** when LPWUI=1 |
| 3 | Run | Stop3 | **STOP instruction** and STOPE=1 |
| 4 | Stop3 | Run | **Interrupt** IF LPR = 0<br>or LPR=1 AND LPWUI=1<br>or reset |
| 5 | Stop3 | LPRun | **Interrupt** when LPWUI=0 |
| 6 | LPRun | Stop3 | **STOP instruction** and STOPE=1 |
| 7 | LPWait | LPRun | **Interrupt** when LPWUI=0 |
| 8 | LPRun | LPWait | **WAIT instruction** |
| 9 | LPWait | Run | **Interrupt** when LPWUI=1 |
| 10 | Wait | Run | **Interrupt** or reset |
| 11 | Run | Wait | **WAIT instruction** |

### 5.7.8 On-Chip Peripheral Modules in Stop and Low Power Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. See Section 5.7.6, “Stop3 Mode”, for specific information on system behavior in Stop3 mode.

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 5-7. Stop and Low Power Mode Behavior**

| Peripheral | Mode | | |
|---|---|---|---|
| | Stop3 | LPWait | LPRun |
| 802.15.4 Transceiver (transceiver) | Off | Optionally On | Optionally On |
| 802.15.4 Transceiver Timers | Optionally On[1] | Optionally On | Optionally On |
| Voltage Regulators Analog | Off | Off | On during PHY active |
| CPU | Standby | Standby | On |
| RAM | Standby | Standby | On |
| Flash | Standby | Standby | On |
| Port I/O Registers | Standby | Standby | On |
| Port I/O Pins | Peripheral Control | Peripheral Control | On |
| BDM | Optionally On | Off[2] | Off[2] |
| COP | Off | Optionally On | Optionally On |
| AES | Off | Optionally On | Optionally On |
| IIC | Standby | Optionally On | Optionally On |
| KBIx | Optionally On | Optionally On | Optionally On |
| LVD/LVW | Optionally On | Optionally On | Optionally On |
| RTC | Optionally On | Optionally On | Optionally On |
| IRQ | Optionally On | Optionally On | Optionally On |
| SCI | Standby | Optionally On | Optionally On |
| SPI | Standby | Optionally On | Optionally On |
| TPMx | Standby | Optionally On | Optionally On |
| CMT | Standby | Optionally On | Optionally On |
| Voltage Regulator Digital | Partial Power down | On | On |
| XOSC_32M | Optionally On | On | On |
| XOSC_32K | Optionally On | Optionally On | Optionally On |
| 1KHz_RC | Optionally On | Optionally On | Optionally On |

[1] Requires the digital regulator to be in high accuracy mode during Stop3.

[2] If ENBDM is set when entering LPRun or LPWait, the MCU will actually stay in run mode or enter wait mode, respectively.

## 5.8 System Control Register Definitions

The registers that manage system control are detailed in the following sections.

### 5.8.1 Interrupt Pin Request Status and Control Register (IRQSC)

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

IRQSC 0x0076

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

▨ = Unimplemented or Reserved

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

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

| Field | Description |
|---|---|
| 6<br>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.<br>0 IRQ pull device enabled if IRQPE = 1.<br>1 IRQ pull device disabled if IRQPE = 1. |
| 5<br>IRQEDG | **Interrupt Request (IRQ) Edge Select** — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When IRQEDG = 1 and the internal pull device is enabled, the pullup device is configured as an optional pulldown device.<br>0 IRQ is falling edge or falling edge/low-level sensitive.<br>1 IRQ is rising edge or rising edge/high-level sensitive. |
| 4<br>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.<br>0 IRQ pin function is disabled pin PTA3 enable.<br>1 IRQ pin function is enabled. |
| 3<br>IRQF | **IRQ Flag** — This read-only status bit indicates when an interrupt request event has occurred.<br>0 No IRQ request.<br>1 IRQ event detected. |
| 2<br>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<br>IRQIE | **IRQ Interrupt Enable** — This read/write control bit determines whether IRQ events generate an interrupt request.<br>0 Interrupt request when IRQF set is disabled (use polling).<br>1 Interrupt requested whenever IRQF = 1. |
| 0<br>IRQMOD | **IRQ Detection Mode** — This read/write control bit selects either edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels that are detected as interrupt request events. See Section 5.6.2.2, "Edge and Level Sensitivity" for more details.<br>0 IRQ event on falling edges or rising edges only.<br>1 IRQ event on falling edges and low levels or on rising edges and high levels. |

## 5.8.2 System Reset Status Register (SRS)

All bits in the SRS register are read only. The SRS register is updated whenever a reset occurs and indicates the cause of the last system reset. Every active reset source has its corresponding bit set. Every inactive reset source has its corresponding bit cleared.

Writes to this address affect the COP function.

| | SRS | | | | 0x1800 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | POR | PIN | COP | ILOP | ILAD[1] | 0 | LVD | 0 |
| W | Writing any value to SRS register clears the COP watchdog timer | | | | | | | |
| POR Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| LVD Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Other Resets: | 0 | Note[2] | Note[2] | Note[2] | 0 | 0 | 0 | 0 |

**Figure 5-6. System Reset Status Register (SRS)**

[1] Illegal address detection is dependent on memory map usage.

[2] Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set: bits corresponding to sources that are not active at the time of reset entry will not be set

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

| Field | Description |
|---|---|
| 7<br>POR | **Power-On Reset** — The POR bit indicates a power-on reset has been detected.<br>0 Power-on reset not detected.<br>1 Power-on reset detected |
| 6<br>PIN | **External PIN Reset** — The PIN bit indicates an external reset has been detected.<br>0 External reset not detected.<br>1 External reset detected. |
| 5<br>COP | **Computer Operating Properly Reset**— The COP bit indicates a COP reset has been detected.<br>0 COP reset not detected.<br>1 COP reset detected. |
| 4<br>ILOP | **Illegal Opcode Reset** — The ILOP bit indicates an illegal opcode reset has been detected.<br>0 Illegal opcode reset not detected.<br>1 Illegal opcode reset detected |
| 4<br>ILAD | **Illegal Address Reset** — The ILAD bit indicates an illegal address reset has been detected.<br>0 Illegal address reset not detected.<br>1 Illegal address reset detected. |
| 1<br>LVD | **Low Voltage Reset** — The LVD bit indicates a low voltage reset or a POR reset has been detected.<br>0 Low voltage reset not detected.<br>1 Low voltage reset detected. |

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

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

| SBDFR | | | | | 0x1801 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | BDFR[1] |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

[1] BDFR is writable only through serial background debug commands, not from user programs.

**Figure 5-7. System Background Debug Force Reset Register (SBDFR)**

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

| Field | Description |
|---|---|
| 0<br>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 a user program.To enter user mode, PTD7/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTD7/BKGD/MS must be low immediately after issuing WRITE_BYTE command. |

**NOTE**

Forcing a system reset using the BDFR results in a reload of the SRS register. Resetting by this method is directly controlled by the user and does not result in the set of any bits of the SRS register unless they become pending on the same cycle.

### 5.8.4 System Options Register 1 (SOPT1)

All bits of the SOPT1 register are write-once after any system reset.

| SOPT1 | | | | | 0x1802 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | COPE | COPT | STOPE | ADCHT | 0 | 0 | BKGDPE | 0 |
| W | | | | | ▨ | ▨ | | ▨ |
| POR/LVD Resets: | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| All other Resets: | 1 | 1 | 0 | 0 | 0 | 0 | 1 | u[1] |

▨ = Unimplemented or Reserved

**Figure 5-8. System Options Register 1 (SOPT1)**

[1] u = unaffected

**Table 5-11. SOPT1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COPE | **COP Enable** — The COPE bit enables the COP.<br>0 COP disabled.<br>1 COP enabled. |
| 6<br>COPT | **COP Time-out** — The COPT bit selects the COP time-out period. See 5.5.2, "Computer Operating Properly (COP) Timer" for additional details of operation.<br>0 COP short time-out period selected.<br>1 COP long time-out period selected. |
| 5<br>STOPE | **STOP Enable** — The STOPE bit enables the STOP instruction. If the STOP instruction is disabled and a STOP instruction is executed, then an illegal opcode reset occurs. See Section 5.7.6, "Stop3 Mode" for details on reset handling.<br>0 STOP instruction disabled<br>1 STOP instruction enabled |
| 4<br>ADCHT | **ADC Hardware Trigger**<br>0 ADC hardware trigger source is RTC<br>1 ADC hardware trigger source is PTC[6] |
| 1<br>BKGDPE | **BKGD Pin Enable** — When BKGD/MS is shared with general-purpose I/O through chip-level hookup, The BKGDPE bit enables the BKGD/MS pin to function as BKGD. When the BKGDPE bit is clear, the pin functions as general-purpose I/O, which must be implemented as output-only.<br>0 BKGD pin disabled.<br>1 BKGD pin enabled. |

## 5.8.5 System Options Register 2 (SOPT2)

The COPCLKS bit in the SIMOPT2 register is a write-once field after any system reset.

**SOPT2** **0x1803**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | COPCLKS[1] | 0 | 0 | ADCCG | 0 | 0 | PID | |
| W | | ▨ | ▨ | | ▨ | ▨ | ▨ | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | PID[2] | |

▨ = Unimplemented or Reserved

**Figure 5-9. SIM Options Register 2 (SIMOPT2)**

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

[2] See Table 5-12 for reset values.

Table 5-12. SOPT2 Register Field Descriptions

| Field | Description |
|---|---|
| 7<br>COPCLKS | **COP Clock Source Select** — The COPCLKS bit selects the COP clock source.{See 5.5.2, “Computer Operating Properly (COP) Timer” for additional details of operation.<br>0 COP logic runs from independent on-chip 1kHz clock source.<br>1 COP logic runs from system bus clock. |
| 4<br>ADCCG | **ADC Clock Gating** — The ADCCG bit must be set for the ADC to be enabled. This is only applicable to MC13237.<br>0 ADC not enabled<br>1 ADC enabled |
| 1:0<br>PID | **Product ID** — The read-only field PID identifies which part you are using.<br>00 MC13234<br>11 MC13237 |

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

These high page read-only registers are included so host development systems can identify the MC13234/MC13237 version. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in the MCU.

| SDIDH | | | | | 0x1806 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | ▨ | ▨ | ▨ | ▨ | ID11 | ID10 | ID9 | ID8 |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset: | x | x | x | x | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

Figure 5-10. System Device Identification Register — High (SDIDH)

Table 5-13. SDIDH Register Field Descriptions

| Field | Description |
|---|---|
| 3:0<br>ID[11:8] | **Part Identification Number** — MC13234/MC13237 identification number. See ID bits in Table 5-15. |

| SDIDL | | | | | 0x1807 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset: | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

▨ = Unimplemented or Reserved

Figure 5-11. System Device Identification Register — Low (SDIDL)

Table 5-14. SDIDL Register Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>ID[7:0] | **Part Identification Number** — MC13234/MC13237 identification number. See ID bits in Table 5-15. |

Table 5-15. Device ID Lookup

| Device | SIDIDH | SIDIDL |
|---|---|---|
| MC13234 | 0x20 | 0x44 |
| MC13237 | 0x30 | 0x44 |

## 5.8.7 System Power Management Status & Control 1 (SPMSC1)

SPMSC1 contains the status and control bits associated with the PMC Voltage Regulator (VREG) and Low Voltage Detect (LVD) functions.

SPMSC1 0x1808

| | 7 | 6 | 5 | 4 | 3 | 2 | 1[1] | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVDF | 0 | LVDIE | LVDRE[2] | LVDSE | LVDE[2] | ▨ | ▨ |
| W | ▨ | LVDACK | | | | | | |
| RESET: | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

▨ = Unimplemented or Reserved U = Unaffected by MCU Reset

[1] Bit 1 is a reserved bit that must always be written to 0.

[2] Write once only after any system reset

Figure 5-12. System Power Management Status & Control 1 Register (SPMSC1)

Table 5-16. SPMSC1 Register Field Descriptions

| Field | Description |
|---|---|
| LVDF | Low-Voltage Detect Flag. This bit indicates the Low-Voltage Detect status if LVDE is set.<br>1 = Low voltage is being or has been detected.<br>0 = Low voltage has not been detected. |
| LVDACK | Low-Voltage Detect Interrupt Acknowledge.<br>Writing a logic 1 to LVDACK clears the LVD interrupt request and clears LVDF to a logic 0 if low voltage is not currently being detected. |
| LVDIE | Low-Voltage Detect Interrupt Enable. This bit enables hardware interrupt requests for LVDF.<br>1 = LVD interrupt enabled when LVDF=1.<br>0 = LVD interrupt disabled (use polling). |

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

| Field | Description |
|---|---|
| LVDRE | Low-Voltage Detect Reset Enable. This write-once bit enables LVDF events to generate a hardware reset. LVD reset has priority over LVD interrupt, if both are enabled. In all test modes, the LVDRE bit value is ignored and functions as if the LVDRE bit value is equal to 0.<br>1 = Force an MCU reset when LVDF=1.<br>0 = LVDF does not generate hardware resets. |
| LVDSE | Low-Voltage Detect Stop Enable. Provided LVDE=1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode.<br>1 = Low-voltage detect enabled during stop mode.<br>0 = Low-voltage detect disabled during stop mode. |
| LVDE | Low-Voltage Detect Enable Bit. This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register.<br>1 = Low-Voltage Detect logic is enabled.<br>0 = Low-Voltage Detect logic is disabled. |

## 5.8.8 System Power Management Status & Control 2 (SPMSC2)

SPMSC2 contains status and control bits associated with the PMC power down modes.

**SPCSC2** **0x1809**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LPR | LPRS | LPWUI | 0 | ▨ | 0 | PPDE[1] | PPDC |
| W | | ▨ | | ▨ | | ▨ | | |
| RESET: | 0 | 0 | 0 | 0 | U | 0 | 1 | 0 |
| POR and LVD: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

▨ = Unimplemented or Reserved    U = Unaffected by MCU Reset

[1] Write once only after any system reset.

**Figure 5-13. System Power Management Status & Control 2 Register (SPMSC2)**

**Table 5-17. SPMSC2 Register Field Descriptions**

| Field | Description |
|---|---|
| LPR | Low Power Regulator Control. This bit controls entry into the low power run and wait modes in which the voltage regulator is put into loose regulation. This bit cannot be set if PPDC is set. If PPDC and LPR are set in a single write instruction, then only PPDC will be set.<br>1 = Low power run and wait modes are enabled.<br>0 = Low power run and wait modes are disabled. |
| LPRS | Low Power Regulator Status. This read-only status bit indicates that the voltage regulator has entered into loose regulation for the low power run or wait mode. Low power run mode cannot be entered if ENBDM is asserted or if ICS_SAFE_FOR_LPRUN_MODE is clear.<br>1 = The voltage regulator is currently in loose regulation.<br>0 = The voltage regulator is not currently in loose regulation. |

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

| Field | Description |
|---|---|
| LPWUI | Low Power Wake Up on Interrupt. This bit controls whether or not the voltage regulator exits loose regulation when any active MCU interrupt occurs.<br>1 = The voltage regulator will exit loose regulation on an interrupt and clear LPR.<br>0 = The voltage regulator will remain in loose regulation on an interrupt. |
| PPDE | Partial Power Down Enable. This write-once bit enables the partial power down feature.<br>0 = Stop3 mode enabled |
| PPDC | Partial Power Down Control. This bit controls which power down mode is selected. This bit cannot be set if LPR is set. If LPR and PPDC are set in a single write instruction, then only PPDC will be set. PPDE must be set to set PPDC.<br>0 = Stop3 low power mode enabled. |

## 5.8.9 System Power Management Status & Control 3 (SPMSC3)

SPMSC3 contains status and control bits associated with the PMC Low Voltage Warning (LVW) and with selecting the low voltage trip point levels .

| | SPCSC3 | | | | 0x180B | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | LVWF | 0 | LVDV | LVWV | LVWIE | 0 | 0 | 0 |
| W | ▨ | LVWACK | | | | ▨ | ▨ | ▨ |
| RESET: | 0[1] | 0 | U | U | 0 | 0 | 0 | 0 |
| POR: | 0[1] | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LVD: | 0[1] | 0 | U | U | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved U = Unaffected by MCU Reset

[1] LVWF will be set not just in the case when Vsupply transitions below the trip point but also after reset and Vsupply is already below $V_{LVW}$. So, LVWF value for RESET and POR will be logic 1 if Vsupply is already below $V_{LVW}$.

**Figure 5-14. System Power Management Status & Control 3 Register (SPMSC3)**

**Table 5-18. SPMSC3 Register Field Descriptions**

| Field | Description |
|---|---|
| LVWF | Low-Voltage Warning Flag. This bit indicates the Low Voltage Warning status if LVDE is set.<br>1 = Low voltage warning is present or was present.<br>0 = Low voltage warning not present. |
| LVWACK | Low-Voltage Warning Acknowledge. Writing a logic 1 to LVWACK clears LVWF to a logic 0 if a low voltage warning is not currently present. |
| LVDV | Low-Voltage Detect Voltage Select. This bit selects the low voltage detect trip point voltage (VLVD). See Table 5-19 for definition of these voltages.<br>1 = High trip point selected (VLVD = VLVDH).<br>0 = Low trip point selected (VLVD = VLVDL). |

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

| Field | Description |
|---|---|
| LVWV | Low-Voltage Warning Voltage Select. This bit selects the low voltage warning trip point voltage (VLVW). See Table 5-19 for definition of these voltages.<br>1 = High trip point selected (VLVW = VLVWH).<br>0 = Low trip point selected (VLVW = VLVWL). |
| LVWIE | Low-Voltage Warning Interrupt Enable. This bit enables hardware interrupt requests for LVWF.<br>1 = LVW interrupt enabled when LVWF=1.<br>0 = LVW interrupt disabled (use polling). |

Table 5-19. LVD and LVW Trip Point Table[1]

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| High LVD trip point (VSUPPLY falling, LVDV=1) | VLVDH | 2.23 | 2.26 | 2.29 | V |
| High LVD trip point (VSUPPLY rising, LVDV=1) | VLVDH | 2.23 | 2.26 | 2.29 | V |
| Low LVD trip point (VSUPPLY falling, LVDV=0) | VLVDL | 1.67 | 1.7 | 1.73 | V |
| Low LVD trip point (VSUPPLY rising, LVDV=0) | VLVDL | 1.74 | 1.77 | 1.8 | V |
| High LVW trip point (VSUPPLY falling, LVWV=1) | VLVWH | 2.29 | 2.32 | 2.35 | V |
| High LVW trip point (VSUPPLY rising, LVWV=1) | VLVWH | 2.33 | 2.36 | 2.39 | V |
| Low LVW trip point (VSUPPLY falling, LVWV=0) | VLVWL | 1.78 | 1.81 | 1.84 | V |
| Low LVW trip point (VSUPPLY rising, LVWV=0) | VLVWL | 1.81 | 1.84 | 1.81 | V |

[1] Not measured on test; checked periodically.

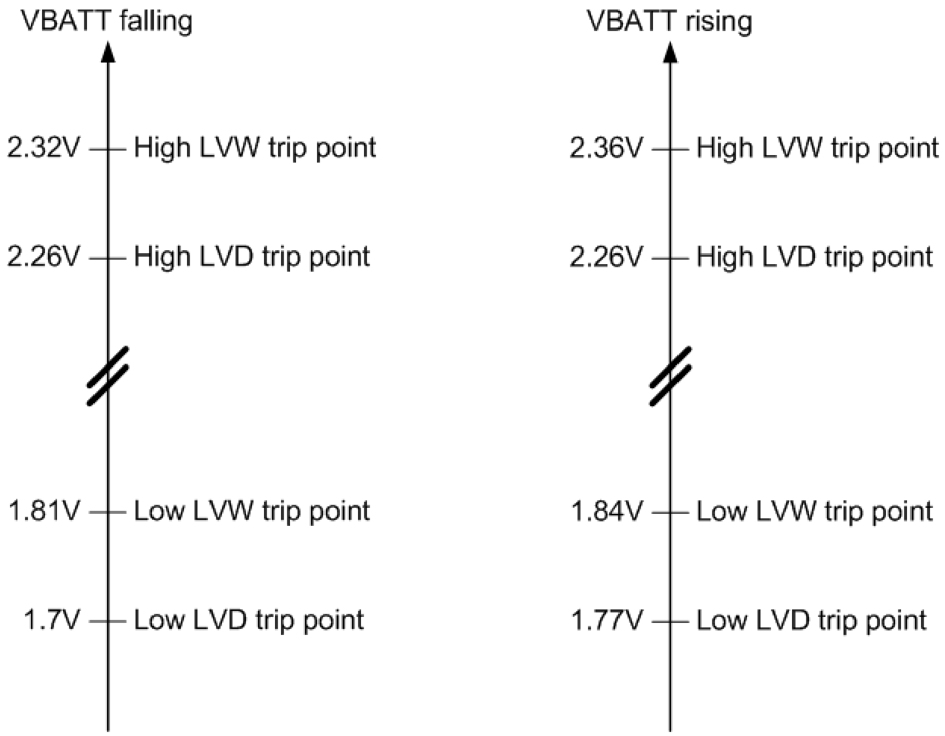


**Figure 5-15. LVD and LVW Illustration**

## 5.8.10 System Oscillator Management and Control Register 1 (SOMC1)

This register contains control bits for the 32 kHz (XTAL0) and 32 MHz (XTAL1) oscillators.

| SOMC1 | | | | | 0x180C | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | RDIV | | | XTAL0EN | XTAL0STEN[1] | XTAL1STEN[1] | Reserved | XTAL0_TST_OUT |
| PoR Exit: | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Other Reset: | u | u | u | u | u | u | 0 | 0 |

▨ = Unimplemented or Reserved

[1] These bits are write-once till next reset

**Figure 5-16. System Oscillator Management and Control Register 1 (SOMC1)**

Table 5-20. SOMC1 Register Field Descriptions

| Field | Description |
|---|---|
| [7:5]<br>RDIV[2:0] | **Bus Frequency Divider** — This controls the CPU clock rate. It sets the ratio to divide down the 32 MHz reference clock. See Table 5-21. |
| 4<br>XTAL0EN | **Crystal 0 Oscillator (32.768 kHz) Enable**<br>0 Oscillator disabled.<br>1 Oscillator enabled. |
| 3<br>XTAL0STEN | **Crystal 0 Oscillator (32.768 kHz) Stop3 Mode Enable**<br>0 Oscillator disabled in Stop3 mode.<br>1 Oscillator enabled in Stop3 mode. |
| 2<br>XTAL1STEN | **Crystal 1 Oscillator (32MHz) Stop3 Mode Enable**<br>0 Oscillator disabled in Stop3 mode.<br>1 Oscillator enabled in Stop3 mode. |
| 1<br>Reserved | **Reserved** |
| 0<br>XTAL0_TST_OUT | **Crystal 0 Test Out (32 kHz) Enable**<br>0 Oscillator output not available at pin.<br>1 Oscillator output available at GPIO pin. |

Table 5-21. CPU Clock vs. RDIV[2:0] Field

| RDIV[2:0] | Divide Ratio | CPU Clock |
|---|---|---|
| 000 | 1 (Default) | 32 MHz |
| 001 | 2 | 16 MHz |
| 010 | 4 | 8 MHz |
| 011 | 8 | 4 MHz |
| 100 | 16 | 2 MHz |
| 101 | 32 | 1 MHz |
| 110 | 64 | 500 kHz |
| 111 | 64 | 500 kHz |

## 5.8.11 System Oscillator Management and Control Register 2 (SOMC2)

This register controls the trim capacitance value for the 32 MHz reference oscillator. The primary capacitive loading for the crystal oscillator is discrete external capacitors and the on board capacitance can be used to trim the crystal load and thus center the oscillator frequency. There are two capacitor arrays (course and fine tune) that are controlled by the XTAL1_TRIM field. See Section 3.5.3, “32 MHz Crystal Trimming”.

| SOMC2 | | | | | 0x180D | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | XTAL1_TRIM | | | | | | | |
| W | | | | | | | | |
| PoR Exit: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Other Reset: | u | u | u | u | u | u | u | u |

▨ = Unimplemented or Reserved

**Figure 5-17. System Oscillator Management and Control Register 2 (SOMC2)**

This register controls the trim value for the 32 MHz osc.

| Field | Description |
|---|---|
| [7:0]<br>XTAL_TRIM[7:0] | **Crystal Oscillator 1 Capacitor Trim**: Warps the crystal frequency:<br>• XTAL_TRIM[7:4] is the coarse tune field<br>• XTAL_TRIM[3:0] is the fine tune field |

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

This register contains control bits to enable or disable the bus clock to the TPMx, CMT, IIC, and SCI modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents.

| SCGC1 | | | | | 0x180E | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 1 | TPM4 | TPM3 | TPM2 | TPM1 | CMT | IIC | SCI |
| W | ▨ | | | | | | | |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

**Table 5-22. SCGC1 Register Field Descriptions**

| Field | Description |
|---|---|
| 6<br>TPM4 | **TPM4 Clock Gate Control** — This bit controls the clock gate to the TPM4 module.<br>0 Bus clock to the TPM4 module is disabled.<br>1 Bus clock to the TPM4 module is enabled. |
| 5<br>TPM3 | **TPM3 Clock Gate Control** — This bit controls the clock gate to the TPM3 module.<br>0 Bus clock to the TPM3 module is disabled.<br>1 Bus clock to the TPM3 module is enabled. |
| 4<br>TPM2 | **TPM2 Clock Gate Control** — This bit controls the clock gate to the TPM2 module.<br>0 Bus clock to the TPM2 module is disabled.<br>1 Bus clock to the TPM2 module is enabled. |

Table 5-22. SCGC1 Register Field Descriptions (continued)

| Field | Description |
|---|---|
| 3<br>TPM1 | **TPM1 Clock Gate Control** — This bit controls the clock gate to the TPM1 module.<br>0 Bus clock to the TPM1 module is disabled.<br>1 Bus clock to the TPM1 module is enabled. |
| 2<br>CMT | **CMT Clock Gate Control** — This bit controls the clock gate to the CMT module.<br>0 Bus clock to the CMT module is disabled.<br>1 Bus clock to the CMT module is enabled. |
| 1<br>IIC | **IIC Clock Gate Control** — This bit controls the clock gate to the IIC module.<br>0 Bus clock to the IIC module is disabled.<br>1 Bus clock to the IIC module is enabled. |
| 0<br>SCI | **SCI Clock Gate Control** — This bit controls the clock gate to the SCI module.<br>0 Bus clock to the SCI module is disabled.<br>1 Bus clock to the SCI module is enabled. |

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

This high page register contains control bits to enable or disable the bus clock to the Debug, Flash, IRQ, KBIx, RTC and SPI modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents.

SCGC2 0x180F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ▨ | DBG | FLS | IRQ | KBI1 | KBI2 | RTC | SPI |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

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

Table 5-23. SCGC2 Register Field Descriptions

| Field | Description |
|---|---|
| 6<br>DBG | **Debug Clock Gate Control** — This bit controls the bus clock gate to the Debug module.<br>0 Bus clock to the Debug module is disabled.<br>1 Bus clock to the Debug module is enabled. |
| 5<br>FLS | **Flash Clock Gate Control** — This bit controls the bus clock gate to the Flash Interface (FTSR) module.<br>0 Bus clock to the Flash interface module is disabled.<br>1 Bus clock to the Flash interface module is enabled. |
| 4<br>IRQ | **IRQ Clock Gate Control** — This bit controls the bus clock gate to the IRQ module.<br>0 Bus clock to the IRQ module is disabled.<br>1 Bus clock to the IRQ module is enabled. |

Table 5-23. SCGC2 Register Field Descriptions (continued)

| Field | Description |
|---|---|
| 3<br>KBI1 | **KBI1 Clock Gate Control** — This bit controls the bus clock gate to the KBI1 module.<br>0 Bus clock to the KBI1 module is disabled.<br>1 Bus clock to the KBI1 module is enabled. |
| 2<br>KBI2 | **KBI2 Clock Gate Control** — This bit controls the bus clock gate to the KBI2 module. Available only on the MC13234.<br>0 Bus clock to the KBI2 module is disabled.<br>1 Bus clock to the KBI2 module is enabled. |
| 1<br>RTC | **RTC Clock Gate Control** — This bit controls the clock gate to the RTC module. Only the bus clock is gated, the ICSERCLK and LPOCLK are still available to the RTC.<br>0 Bus clock to the RTC module is disabled.<br>1 Bus clock to the RTC module is enabled. |
| 0<br>SPI | **SPI Clock Gate Control** — This bit controls the bus clock gate to the SPI module.<br>0 Bus clock to the SPI module is disabled.<br>1 Bus clock to the SPI module is enabled. |

# Chapter 6
# IEEE 802.15.4 Transceiver (including 802.15.4 Module)

## 6.1 Introduction

The MC13234/MC13237 provides a complete on board system solution for the IEEE 802.15.4 Standard from the RF Physical Layer requirements through a complete MAC Layer implemented via dedicated hardware as well as flash-based code. Key characteristics of the MC13234/MC13237 802.15.4 implementation include:

- 2.4 GHz ISM Band operation
- Over-the-air data rate of 250 kbps
- 16 Standard independent channels
- Programmable transmitter output power
- Uses carrier sense multiple access with collision avoidance (CSMA-CA) protocol
- Provides Energy Detection (ED), Clear Channel Assessment (CCA), and Link Quality Indication (LQI) through MAC services
- Dedicated hardware to off load 802.15.4 services from CPU
  - PHY Layer sequence Manager
  - Direct memory access (DMA) hardware to move data between transceiver and RAM
- Hardware acceleration for AES security services of combined counter with CBC-MAC (cipher block chaining message authentication code)
- Special partial power down “listen” receive mode (PPD_RX) for lower current or to reduce receiver sensitivity.

The MC13234/MC13237 transceiver implementation consists of the 2.4 GHz radio, modem functions, sequence manager, and control registers. These elements combined with the flash-resident MAC services running on the CPU provide the complete 802.15.4 MAC service. In addition, a dedicated timer block is available to supplement the MCU TPM timer resources.

This chapter first provides an overview and general description for the radio and the modem. Following the overview, the sequence manager, timer resources, and control functions are described. The descriptions of control interface registers are then provided.

**NOTE**

> Although this chapter is provided for use of the transceiver and knowledge of the IEEE 802.15.4 Standard services, the user is directed to Freescale’s stacks and codebases for complete MAC services.

## 6.2 Features

Features of the MC13234/MC13237 transceiver and IEEE 802.15.4 services implementation include:

- Receive sensitivity of -94 dBm (typical) at 1% PER, 20-byte packet, much better than the IEEE 802.15.4 Standard of -85 dBm
- Programmable output power with 0 dBm nominal output power, programmable from –30 dBm to +2 dBm typical
- Small RF footprint
  - Integrated transmit/receive switch
  - Differential input/output port used with balun
  - Low external component count
- DMA interface to move packet data between transceiver and RAM
- Separate hardware AES-128 Security module; see Chapter 7, “Advanced Security Module (ASM)”
- 16-Bit random number generator
- 802.15.4 Auto-sequence support
  - Automated sequence management requires no software intervention
  - Standard packet sequences (transmission, reception, CCA services)
  - Software or timer-initiated sequences
  - Supports both 2003 and 2006 versions of IEEE 802.15.4 Standard
  - Supports slotted and unslotted modes
  - Supports beacon-enabled and non-beacon-enabled networks
- 802.15.4 Receiver frame filtering.
- Four additional 24-bit transceiver timer comparators available for MAC operations

## 6.3 Transceiver Overview

The MC13234/MC13237 transceiver is capable of complete standalone operation. After it has been programmed through the transceiver control registers, the sequence manager has the highest level control to the radio and modem, and the DMA passes packet data to/from the RAM during a transmit or receive operation. The CPU is released to do other parallel tasks, and this also eliminates any timing dependency required to move data between the transceiver and memory. Also, the DMA works on a cycle-steal basis from the CPU bus, and data are transferred on a byte-by-byte, as-required basis.

### 6.3.1 Transceiver Block Diagram

Figure 6-1 shows a simplified block diagram of the MC13234/MC13237 IEEE 802.15.4 transceiver. As the diagram shows, the transceiver is loosely divided between the analog (radio) and digital functions.

- The radio contains all the analog circuitry used to transmit and receive an RF signal.

- On transmission, the radio is provided the encoded serial data stream from header and symbol generator blocks (transmit modem function), uses it to phase-shift modulate the carrier signal from the synthesizer, and then drives the programmable power amplifier (PA)
- On reception, the radio transfers the RF signal through the transmit/receive switch (T/R SW), amplifies it through the low noise amplifier (LNA), down converts and produces “I” and “Q” channels through a 2-stage mixer chain, and then samples the resulting I and Q signals through high-speed analog-to-digital (ADC) converters.

- The main VCO and synthesizer block provides the proper RF LO1 frequencies required by the radio to carry the modulated transmit signal while supporting the receiver's front end and signal down conversion path.
- The “802.15.4 Module” includes all the control/status logic, modem logic, and Event Timer – Within the digital modem logic, the transmit and receive paths are separate and are controlled by the control register logic block. Also, with the sequence manager and DMA functions to supervise the actual radio operation and packet transfers, the transceiver is capable of standalone operation allowing the CPU to run independent application code.

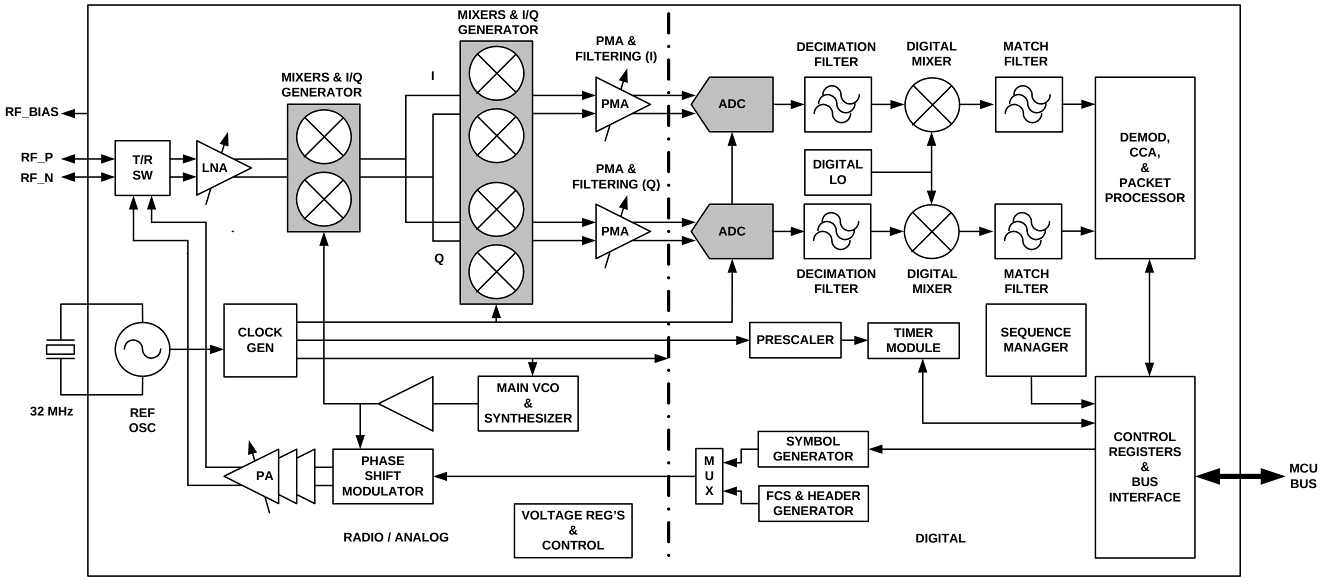


**Figure 6-1. MC13234/MC13237 IEEE 802.15.4 Transceiver Block Diagram**

## 6.3.2 Direct Memory Access (DMA) Controller

A simple cycle-stealing DMA controller moves packet data between the transceiver and memory during a transmit or receive operation. This off-loads the data transfer task from the CPU and improves performance. The DMA operates as follows:

- During a receive operation –
  - The DMA receive address pointer is loaded from the RXD_ADR_PNTR[[23:0] field, see Section 6.13.2, “Receive Data Pointer Registers (RXD_ADR_PNTR0 and RXD_ADR_PNTR1)”.

- As the data are received, they are directly placed into RAM buffer starting at the pointer address. The received FCS data is included in the data transferred to the RAM buffer. The receiver will transfer the data as one byte (octet) at a time (every 32 uS) into RAM. When a new byte is received the destination address will increment accordingly.
  - The first element of the received packet to be placed in the buffer will be the Frame Length field.
  - The last element will the Frame Check Sequence.
  - The length field defines the number of bytes in the MPDU. Per definition the MPDU does not include the length field count itself but includes the Frame Check Sequence (FCS). Thus the user must allocate a buffer 1 byte larger than the expected MPDU length.
- As the receive buffer is loaded, the byte count is shown in the Receiver Byte Count Register, see Section 6.13.10, "Receiver Byte Count (RX_BYTE_COUNT)". The receive byte counter can be used to generate an interrupt request as the frame is received. When the byte count matches a preset level or "watermark", see Section 6.13.9, "Receiver Byte Count Watermark Threshold (RX_WTR_MARK)", a status is set and can be enabled to generate the IRQ.

- During transmit operations -
  - The transmit packet must have been pre-assembled and available in the memory buffer.
  - The data is automatically retrieved from the memory buffer starting from the address specified by the TXD_ADR_ PNTR[[23:0] field, see Section 6.13.1, "Transmit Data Pointer Registers (TXD_ADR_PNTR0 and TXD_ADR_PNTR1)".
  - The first byte retrieved is the FLI (see Section 6.4.1, "PHY Protocol Data Unit (PPDU)") and the DMA controller uses this value to determine how many bytes to fetch for the complete packet
  - The memory buffer does not include the two FCS (see Section 6.4.1, "PHY Protocol Data Unit (PPDU)") bytes, as these are calculated by the hardware generator and appended to the payload data sourced from memory.

## 6.4 IEEE 802.15.4 Standard PHY Packet and Frame Structures

A basic knowledge of the PHY packet and frame structure is important to best understand basic operation of the transceiver and sequence manager.

The MC13234/MC13237 transceiver supports the IEEE 802.15.4 2450 MHz Physical Layer (PHY) specification. This standard uses serial 250 kbps offset quadrature phase-shift keying (O-QPSK) data in 5.0 MHz channels and full spread-spectrum encode and decode. It operates on one of 16 selectable channels in the 2.4 GHz ISM band. The data are transmitted least significant symbol first within a byte (octet), and least significant byte first within a data field.

**NOTE**

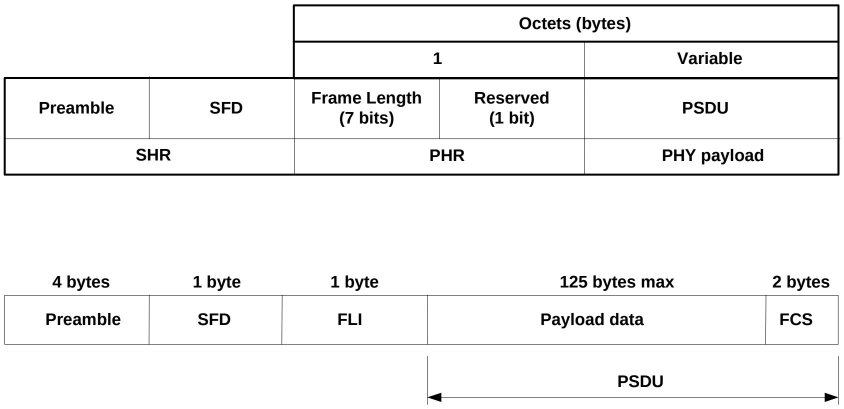
The following descriptions only provide an overview of the IEEE 802.15.4 PHY protocol data unit and frame structures. For more information see the IEEE 802.15.4 Standard, "*Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Std 802.15.4™-2006*

## 6.4.1 PHY Protocol Data Unit (PPDU)

The basic unit of the PHY transmission is the PHY protocol data unit (PPDU), see Figure 6-2. Each PPDU consists of three basic components:

- A synchronization header (SHR), which allows a receiving device to synchronize and lock onto the bit stream. The SHR is composed of -
  - — Preamble – is a minimum of 4 bytes in length and allows the receiver to synchronize to the incoming packet
  - — Start Frame Delimiter (SFD) – is a 1-byte field with a unique bit pattern that indicates start of packet data
- A PHY header (PHR), which contains frame length information; it is a 1-byte field composed of -
  - — 7-bit Frame Length Field (FLI) – specifies total number of following bytes of the payload data (127 bytes maximum)
  - — 1-bit Reserved Field
- PHY Payload – is a variable length field which carries the MAC sublayer frame. Designated as the PHY Service Data Unit (PSDU) -
  - — Maximum length is127 bytes
  - — Last two bytes are always Frame Check Sequence (FCS) data
  - — Total length (FLI) is actual payload data plus FCS data

Figure 6-2 shows two views of the packet / PPDU structure. The first view reflects the PPDU field definitions, and the second view is a simplified view emphasizing field byte lengths.



| | | Octets (bytes) | | |
|---|---|---|---|---|
| | | 1 | | Variable |
| Preamble | SFD | Frame Length<br>(7 bits) | Reserved<br>(1 bit) | PSDU |
| SHR | | PHR | | PHY payload |

| 4 bytes | 1 byte | 1 byte | 125 bytes max | 2 bytes |
|---|---|---|---|---|
| Preamble | SFD | FLI | Payload data | FCS |

Figure 6-2. IEEE 802.15.4 PPDU / Basic Packet Structure

## 6.4.2 MAC Frame Structures

Understanding the MAC frame structures is useful to understanding operation of the Sequence Manager.

In the MAC frame structure, The MAC Protocol Data Unit (MPDU) which originates from the MAC sublayer and makes up the PSDU of the PPDU is the MAC sublayer frame. Maximum length is 127 bytes. The MPDU can consist of three fields -

- MAC header (MHR) – is always present and can contain
  - MAC Frame Control field – 2 bytes in length and contains information defining the frame type, addressing fields, and other control flags. Required field.
  - Sequence Number field – 1 byte in length and specifies the sequence identifier for the frame (either a beacon sequence number (BSN) or a data sequence number (DSN). Optional field as required by frame type.
  - Addressing Fields – Destination PAN Identifier (0/2 bytes), Destination Address (0/2/8 bytes), Source Address (0/2/8). Use as required by frame type.
  - Auxiliary Security Header – (0/5/6/10/14 bytes). Optional field as required by security use.
- Mac Payload – Variable length. Specific information to the frame. Optional field as required by frame type.
- MAC footer (MFR) – Frame check sequence (FCS) field (2 bytes). Required field.

Table 6-1 shows the general MAC frame format.

**Table 6-1. General MAC Frame Format**

| Octets:2 | 1 | 0/2 | 0/2 | 0/2/8 | 0/2 | 0/5/6/10/14 | Variable | 2 |
|---|---|---|---|---|---|---|---|---|
| Frame Control | Sequence Number | Destination PAN Identifier | Destination Address | Source PAN Identifier | Source Address | Auxiliary Security Header | Frame Payload | FCS |
| | | Addressing Fields | | | | | | |
| MHR | | | | | | | MAC Payload | MFR |

The IEEE 802.15.4 Standard defines four frame structures as described in Table 6-2.

**NOTE**

Table 6-2 shows only the PSDU field contents. The complete PPDU or PHY packet also contains the Preamble, SFD, and FLI fields.

Table 6-2. MAC Frame Format

| Field | Description |
|---|---|
| Beacon Frame | The Beacon Frame is used by a coordinator to transmit beacons. A coordinator can transmit network beacons in a beacon-enabled PAN. The frame contains:<br>• The MHR – contains the MAC Frame Control field, beacon sequence number (BSN), addressing fields, and optionally the auxiliary security header.<br>• The MAC payload – contains the superframe specification, GTS fields, pending address fields, and beacon payload.<br>• The MFR – contains a 16-bit frame check sequence (FCS). |
| Data Frame | The Data Frame is used for all transfers of data and originates from the upper layers. The data payload is passed to the MAC sublayer and becomes the MPDU. The frame contains:<br>• The MHR – contains the MAC Frame Control field, data sequence number (DSN), addressing fields, and optionally the auxiliary security header.<br>• The MAC payload – upper layer data payload<br>• The MFR – contains a 16-bit frame check sequence (FCS). |
| Acknowledgment Frame | The Acknowledgment Frame originates within the MAC sublayer and is used for confirming successful frame reception. Having no MAC payload, the frame contains:<br>• The MHR – contains the MAC Frame Control field and data sequence number (DSN)<br>• The MFR – contains a 16-bit frame check sequence (FCS). |
| MAC Command Frame | The MAC Command Frame originates within the MAC sublayer and is used for handling all MAC peer entity control transfers. The frame contains:<br>• The MHR – contains the MAC Frame Control field, data sequence number (DSN), addressing fields, and optionally the auxiliary security header.<br>• The MAC payload – MAC command payload.<br>• The MFR – contains a 16-bit frame check sequence (FCS). |

## 6.5 Basic PHY Operations

Using a simplistic approach, all 802.15.4 PHY transactions are built around three basic 802.15.4 PHY operations:

- Clear Channel Assessment (CCA) / Energy Detect (ED) – testing for energy on the selected channel:
  - CCA cycle – Looking for an idle channel condition as part of the collision avoidance protocol, before executing a transmission.
    - For a CCA Mode 1 operation – the channel energy is compared to a preset threshold
    - For a CCA Mode 2 operation – if a 802.15.4 modulated signal is found, the channel is considered busy
  - Observing individual channel energy (ED) as part of a channel scan for the MAC services
  - Measuring energy of a received packet to report LQI
- Packet transmission (TX) – transmitting or sending a PHY protocol data unit using one of the frame structures previously defined
- Packet reception (RX) – receiving a PHY protocol data unit.

## 6.5.1 Clear Channel Assessment (CCA) Operation (including ED and Link Quality Indication)

A CCA operation is a broad designation for any form of measuring channel energy. The measurement is done using the receiver and uses different algorithms or modes.

### 6.5.1.1 CCA Modes, Energy Detect, and Link Quality Detect Algorithms

The Clear Channel Assessment (CCA) mode and energy detect algorithms described here are consistent with the IEEE 802.15.4 Standard definitions.

#### 6.5.1.1.1 CCA Mode 1 Algorithm

The CCA Mode 1 algorithm measures an average energy in any received baseband signal (on the programmed channel). For a Mode 1 operation, the receiver moves from an IDLE condition to a RECEIVE condition (designated as a 144 µs RX Warm-up) and then does an RSSI measurement for 128 µs (as defined by the 802.15.4 standard and equals 8 symbol times). Figure 6-3 shows the operation timing.

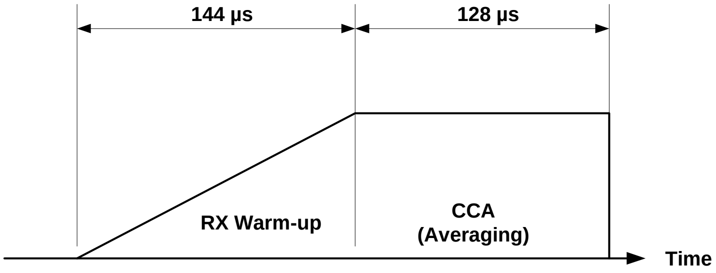


**Figure 6-3. CCA Sequence Timing**

For CCA Mode 1, the RSSI measurement is done by calculating a 128µs (8 symbol) average of:

$$\sqrt{I^2 + Q^2}$$

CCA Mode 1 is used to report an “energy above threshold” status for a CCA cycle. The threshold level is provided by the CCA_THRESHOLD Register, and if the detected energy is above the threshold the CCA status is set to “Busy”

#### 6.5.1.1.2 CCA Mode 2 Algorithm

The CCA Mode 2 algorithm looks for the presence of an <u>802.15.4 compliant received signal</u> (on the programmed channel) and signal strength is unimportant. For a Mode 2 operation, the receiver first executes a 144 µs RX Warm-up, similar to Mode 1. However, Mode 2 differs in that the preamble detect function is activated for the 128 µs, 8 symbol time period, and looks for correlation of an 802.15.4 signal.

CCA Mode 2 can be used to report a “carrier sense” status for a CCA cycle. The CCA status will be set to “Busy” is a carrier is detected.

#### 6.5.1.1.3 Energy Detect (ED) Algorithm

The energy detect algorithm also measures an average energy in any received baseband signal (on the programmed channel). The difference from the Mode 1 algorithm is that a 64 µs AGC settling time is inserted between the RX Warm-up cycle and the 128 µs signal strength measurement. Figure 6-4 shows the operation timing.

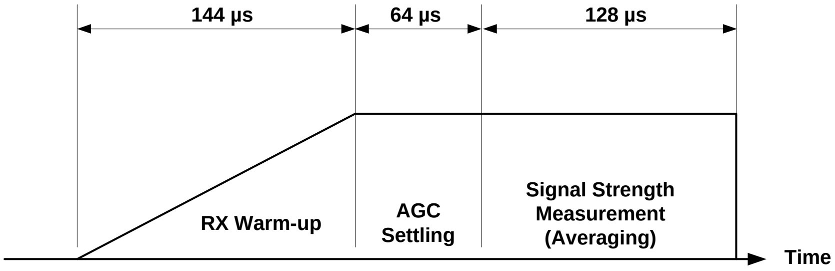


Figure 6-4. ED Sequence Timing

ED mode is typically used to report a status for channel scan.

#### 6.5.1.1.4 Link Quality Indication (LQI) Algorithm

LQI is a measurement of average energy of the received baseband signal during a packet reception. The energy measurement is similar to the ED measurement except that the sample time is delayed until the packet preamble detect. Figure 6-5 shows the operation timing.

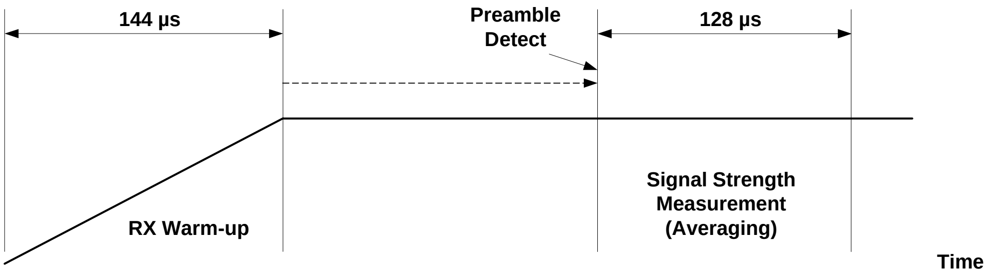


Figure 6-5. LQI Measurement Timing

The LQI value is reported with the receive packet.

### 6.5.1.2 CCA-Associated Control Register Fields

The execution of CCA/energy measurement functions is controlled by the Sequence Manager, and their use and status related to a given sequence is associated with the following registers and register fields:

- CNTRL1 Register – see Section 6.13.3, “Transceiver Control Register 1 (CNTRL1)”
  - — CCABFRTX Bit : this bit enables/disables a CCA operation before a transmit in a Transmit or a Transmit/Receive sequence.
- CNTRL2 Register – see Section 6.13.4, “Transceiver Control Register 2 (CNTRL2)”
  - — CCATYPE[1:0] : this 2-bit field selects the CCA algorithm used during a sequence (if enabled)
- CNTRL3 Register – see Section 6.13.5, “Transceiver Control Register 3 (CNTRL3)”

  - — CCAMSK : this enables/disables an interrupt request due to a “CCAIRQ” status
- STATUS1 Register – see Section 6.13.11, “Status Register 1 (STATUS1)”
  - — CCA : this bit reports the result of a CCA/ED operation
- STATUS2 Register – see Section 6.13.12, “Status Register 2 (STATUS2)”
  - — CCAIRQ : this bit reports completion of an CCA/ED operation
- CCAFNL Register – see Section 6.13.3, “Transceiver Control Register 1 (CNTRL1)”
  - — CCAFNL[7:0] : this 7-bit field is the reported value of the energy measurement from a CCA, ED, or LQI operation
- CCA_THRESHOLD Register – see Section 6.13.30, “CCA Energy Threshold (CCA_THRESHOLD)”
  - — CCA_THRESHOLD[7:0] : this 7-bit field is the energy threshold value used during a CCA cycle to compare for a clear channel
- CCA_OFFSET_CMP Register – see Section 6.13.31, “CCA Power Compensation (CCA_OFFSET_CMP)”
  - — CCA_OFFSET_CMP[7:0] : this 7-bit field is an offset correction value that is added to trim the measured energy level

### 6.5.1.3 Reported Energy Values

The average value of the signal power is calculated and stored in field CCA_FINAL[7:0] for a CCA, ED, or LQI indication. The reported 8-bit binary value is an indication of the signal strength in dBm. However, to determine the decimal equivalent of the value stored in CCA_FINAL[7:0], one must convert the hex value to its decimal value, divide by two, and change the sign; where this calculated value is equivalent to the received signal strength in dBm:

Signal strength in dBm = – (dec (CCA_FINAL[7:0] / 2))

The value stored in CCA_FINAL[7:0] is also affected by an offset stored in field CCA_OFFSET_CMP[7:0]. The default value of CCA_OFFSET_CMP[7:0] = 0x8D and is added to the measured value during the CCA/ED/LQI function to give the stored value in CCA_FINAL[7:0].

The compensation number added to the internal measured value is CCA_OFFSET_CMP[7:0] / 2 and the resulting addition is shown in Figure 6-6.

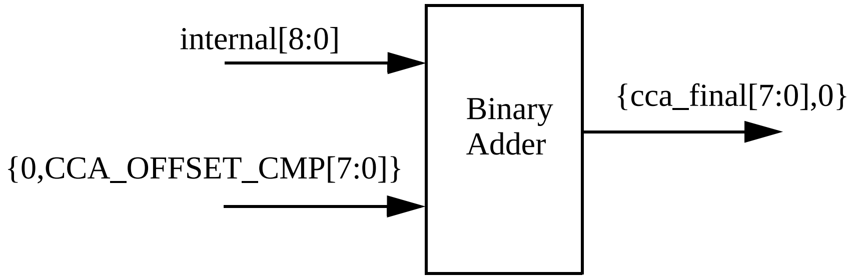


**Figure 6-6. Compensation Factor Added to Internal CCA Value to Produce Corrected Final Value**

The resulting formula for dBm means that the CCA_OFFSET_CMP[7:0] value must increment or decrement by 4 to cause a 1 dBm change, which equates to a 1/4 dBm resolution for the CCA_OFFSET_CMP correction.

The CCA_OFFSET_CMP[7:0] default value is 0x8D and the value must increment to set a lower final CCA value. As an example, if one were putting in a -30 dBm signal and the CCA_FINAL[7:0] was reporting -26 dBm, then the correction value in power_comp[7:0] should be increased by (4 * delta) or (4 * 4) in this case. The new compensated value of power_comp[7:0] = 0x8D + 0x10 = 0x9D.

**NOTE**

As previously stated, the default value for power_comp[7:0] = 0x8D. For the MC13234/MC13237 device, an offset of 3.5 dBm is needed to center the reported CCA/LQI value over temperature. As a result, it is suggested that a value of 0x9B be programmed into CCA_OFFSET_CMP[7:0] for best accuracy.

Because the AGC is set to a fixed gain during the CCA procedure, input signals above -65 dBm will not be reflected correctly due to saturation. This is not a problem because the purpose of this measurement is to detect a signal above a threshold that is not to exceed -75 dBm. See the 802.15.4 Standard for details.

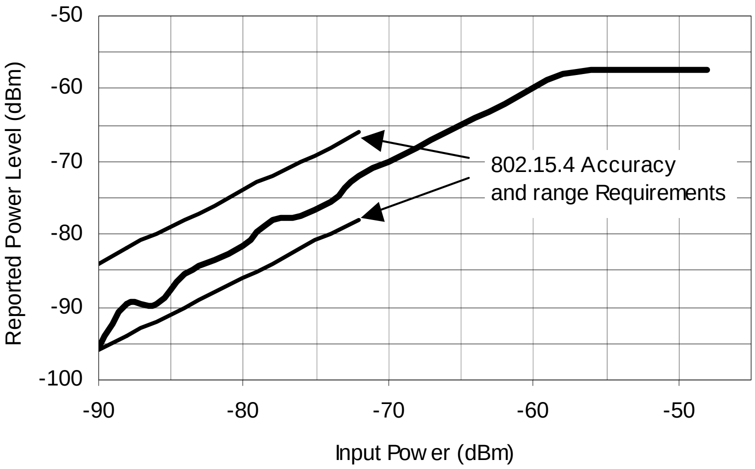


**Figure 6-7. CCA Reported Power Level vs. Input Power**

### 6.5.1.3.1 CCA Result

For a CCA operation, the value contained in CCA_FINAL[7:0] is compared to the preset threshold of field CCA_THRESHOLD[7:0]. If CCA_FINAL[7:0] is equal to or less than the threshold, then status CCA is set to 1, indicating a busy channel. If CCA_FINAL[7:0] is greater than the threshold, CCA remains at 0. After the CCA operation is complete, CCAIRQ is asserted.

The value of CCA_THRESHOLD[7:0] is calculated:

Threshold value = hex (| (Threshold Power in dBm) * 2 |)

A suggested threshold is -82 dBm or 0xA4 = CCA_THRESHOLD[7:0].

### 6.5.1.3.2 Energy Detect Function

With the energy detect algorithm, the same energy results are reported, but the operation is done without the threshold comparison.

**NOTE**

For the following graph, the required 802.15.4 Standard accuracy and range limits are shown. A 3.5 dBm offset has been programmed into the ED/LQI reporting level to center the level over temperature in the graph.

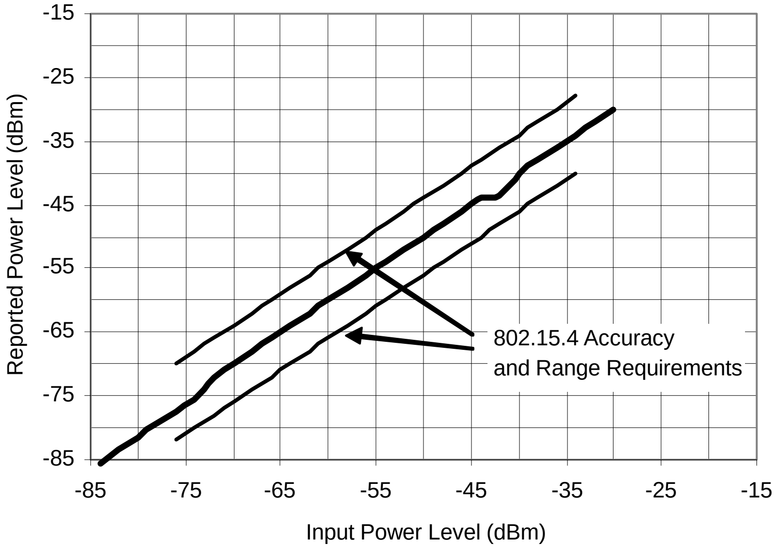


**Figure 6-8. ED and LQI Reported Power vs. Input Power**

Status bit CCA is unaffected by energy detect. After the energy detect operation is complete, CCAIRQ is asserted.

### 6.5.1.3.3 Link Quality Indication

Link Quality Indication is a measure of the signal quality during an actual receive operation. Its value is also stored in field CCA_FINAL[7:0].

**NOTE**

In Figure 6-8, the required 802.15.4 Standard accuracy and range limits are shown. A 3.5 dBm offset has been programmed into the LQI reporting level to center the level over temperature. Typical values of LQI returned from an RX operation are from about -95 dBm to about -18 dBm giving a CCA_FINAL[7:0] range of decimal values 190 (0xBE) to 36 (0x24). When used in an 802.15.4 application, the LQI must be scaled and limited to have a range of decimal 0 to 255. Values of LQI above approximately -30 to -25 dBm can be non-linear and should not be used as an indication of absolute received power.

## 6.5.2 Packet Transmission (TX)

A packet transmission sends a PHY packet as shown in Figure 6-2. The packet consists of (in order) the preamble, SFD, FLI, payload data, and last, the FCS. In the transceiver, the transmit operation is automated through use of the DMA and the Sequence Manager:

- The DMA pointer must be preloaded
- The data buffer must be preloaded in RAM (includes the FLI and payload data, not including FCS)
- The Sequence Manager options/controls must be initiated.
- The transmit may or may not be preceded by a CCA operation
- Transmit results as the execution of a Sequence T or Sequence T/R operation

For either transmit sequence, a very simple overview consists of (for more detail, see Section 6.9.5, “Basic Transmit Sequence – Sequence T (Transmit)”):

1. Transmitter warm-up
2. Preamble transmitted (4 octets from hardware)
3. SFD transmitted (octet from hardware)
4. FrameLength octet transmitted -fetched from from RAM
5. Payload data transmitted – fetched from RAM on byte-by-byte basis as determined by FLI
6. FCS transmitted (2 octets generated in hardware)
7. Transmitter warm-down

## 6.5.3 Packet Reception (RX)

Packet reception is more complex and includes hardware filtering for accepting/rejecting packets.

### 6.5.3.1 Basic RX Operation

Packet reception is enabled either as a simple receive sequence or as part of a longer sequence.

- The receive DMA pointer must be preloaded
- The data buffer area in RAM should be reserved.
- The filtering options must be preset
- The Sequence Manager options/controls must be initiated.

For a simple receive sequence, an overview for a successful receive consists of :

1. Receiver warm-up
2. Receiver enters receive mode and waits for signal
3. Preamble detected
4. SFD detected
5. FrameLength octet received – saved to check byte count as well as saved to RAM
6. Payload data received and transferred to RAM on byte-by-byte basis as determined by FLI (includes received CRC bytes)
7. Frame filtering checked as received and CRC checked

8. Frame ready for examination.

#### 6.5.3.2 Receive Operating Modes

The MC13234/MC13237 is unique in that it supports two operational receive modes:

- Standard receive (RX) mode – the default receive mode sets the receiver to maximum sensitivity and operates in a “normal” mode. The receiver is fully on and stays active until a frame is received or a time-out occurs (if enabled).
- Partial Power Down receive (PPD_RX) – this mode can be enabled as an option.
  - Whenever a receive cycle is initiated, the receiver is not fully on to save current until energy of a preset level is detected.
  - The receiver turns on fully upon being triggered by energy at the preset level, and then receives the expected frame. The full-on state is the same as the standard receive state.

Use of the PPD_RX mode can provide two distinct advantages:

1. Reduced “listen” mode current – the receive current is significantly reduced while waiting for a frame. If a node is a coordinator, router, or gateway and it spends a significant percentage of its RF-active time waiting for incoming frames from clients or other devices, the net power savings can be significant. There is a cost of reduced sensitivity.
2. Reduced sensitivity as a desired effect – the PPD_RX mode can provide different levels of reduced sensitivity. If a node operates in a densely populated area, it may be desirable to de-sensitize the receiver such that the device does not respond to incoming frames with an energy level below the desired threshold. This could be useful for security, net efficiency, reduced noise triggering, as well as other purposes.

**NOTE**

Freescale Codebases provide function calls for enabling, disabling and configuring the PPD_RX mode.

#### 6.5.3.3 Receive Packet Filtering and Responses

The MC13234/MC13237 transceiver hardware has the ability to accept/reject a received frame and enable “auto” sequence features based on the frame attributes.

As briefly described in Section 6.4.2, “MAC Frame Structures”, each data packet contains a MAC header that can contain a MAC Frame Control Field (2 bytes), Sequence Number Field (1 byte), Address Fields (variable length), and an auxiliary security header (variable). Table 6-1 shows this structure. To further describe the filtering features, the Frame Control Field (FCF) format is shown in Table 6-3. The transceiver filtering and response mechanisms are gated by these FCF sub-fields and the Addressing Fields.

Table 6-3. Frame Control Field Format

| Bits:<br>15-14 | 13-12 | 11-10 | 9-7 | 6 | 5 | 4 | 3 | 2-0 |
|---|---|---|---|---|---|---|---|---|
| Source Addressing Mode | Frame Version | Destination Addressing Mode | Reserved | PAN ID Compression | Ack Request | Frame Pending | Security Enabled | Frame Type |

When receiving a packet, the following capabilities are possible:

1. Promiscuous Mode – this mode will receive ALL frames with no filtering except for Frame Length checking and optional CRC checking.
   - Mode is enabled by PROMISCUOUS bit, CNTRL2 Register – see Section 6.13.4, "Transceiver Control Register 2 (CNTRL2)"
   - CRC is disabled by CRC_MSK bit, CNTRL3 Register – see Section 6.13.5, "Transceiver Control Register 3 (CNTRL3)"
2. Hardware generated Ack response – for the Receive and Transmit/Receive sequences, this feature will follow a received frame that passes required filtering and that requests an Ack (FCF "Ack Request" sub-field = 1) with a transmitted Ack frame. The Ack frame uses the Sequence Number from the received frame. Promiscuous mode disables this feature.
3. Programmable frame filtering – based on FCF sub-fields, Sequence Number field, and Addressing Fields.
   - Pan Coordinator enable – the PANCORDNTR bit of CNTRL2 enables the device as Pan Coordinator. This enables reception of MAC command or data frames with no destination address
   - Frame Type FCF sub-field – as defined in Table 6-4, the frame type is used to help determine the filter characteristics. See paragraphs below.
   - RX_FRAME_FILTER Register control bits (see Section 6.13.28, "Receive Frame Filter (RX_FRAME_FILTER)"- as summarized in Table 6-5 set the accept/reject criteria, See paragraphs below.
4. IEEE 802.15.4 Standard Version control

Table 6-4. IEEE 802.15.4 Frame Type Subfield

| Frame Type [2:0] | Frame Type Designation |
|---|---|
| 000 | Beacon |
| 001 | Data |
| 010 | Acknowledgement |
| 011 | MAC command |
| 100-111 | Reserved |

Frame filtering uses the 3-bit Frame Type subfield (Table 6-4) value within the receive frame 2-byte FCF to determine rejection of the incoming receive frame. Control bits in the RX_FRAME_FILTER Register set the accept/reject options as shown in Table 6-5. The frame type is further qualified by the address field(s).

Table 6-5. Frame Filter Control Bits

| Control Bit | Frame Type (FT[2:0]) | Filter Option Description |
|---|---|---|
| NS_FT | Not Specified (100-111) | When set to 1, non-specified frames are accepted |
| CMD_FT | Command (011) | When set to 1, Command frames are accepted |
| ACK_FT | Ack (010) | When set to 1, Ack frames are accepted |
| DATA_FT | Data (001) | When set to 1, Data frames are accepted |
| BEACON_FT | Beacon (000) | When set to 1, Beacon frames are accepted |

The receive operation will complete only if the following conditions are met:

1. Destination address:
    a) If PAN ID is included other than the broadcast, it must match the MACPANID[15:0] field (see Section 6.13.19, “MAC Pan ID Registers (MACPANID0 and MACPANID1)”) else the PAN ID must be the broadcast value (0xFFFF)
    b) If a short destination address is included in the frame, it must match the MACSHORTADDRESS[15:0] field (see Section 6.13.18, “MAC Short Address Registers (MACSHORTADDRS0 and MACSHORTADDRS1)”) else the broadcast address (0xFFFF)
    c) If a long destination address is include in the frame, it must match the MACSLONGADDRESS[63:0] field (see Section 6.13.26, “MAC Long Address Registers (MACLONGADDRS0 – MACLONGADDRS7)”)
    d) If only source addressing fields are included in a Data or MAC Command frame, the frame shall be accepted only if the device is enabled as a PAN coordinator (PANCOORD = 1) and the source PAN identifier matches MACPANID[15:0].
2. Illegal Frame types are rejected, unless the NS_FT bit is set.
3. Beacon frames are accepted under the following conditions
    a) Beacon frames enabled; the BEACON_FT bit is set.
    b) The source PAN ID matches MACPANID[15:0] or it equals 0xFFFF
    c) Length >= 9
    d) The Destination Addressing Mode sub-field is 0 (no destination address)
    e) The Source Addressing Mode sub-field is 2 or 3 (i.e. a source address is included)
4. Data frames are accepted under the following conditions:
    a) Data frames are enabled; the DATA_FT bit is set.
    b) Length >= 9
    c) If the device is a PAN coordinator (PANCOORD bit) and the source PAN identifier matches MACPANID[15:0]. This means that if the destination address is not present, the frame will be accepted only if the PANCOORD = 1 and the PANID in the source address field is equal to the

receiving device MACPANID[15:0]. If the device is NOT a PAN coordinator, the Destination Address Mode must be 2 or 3, and destination address-matching rules apply.

5. Acknowledgement Frames are accepted under the following conditions:
   a) Acknowledge Frames are enabled; the ACK_FT bit is set.
   b) Length == 5
   c) Sequence Number has to match the previously sent TX frame from this device.
6. Command Frames are accepted under the following conditions:
   a) Command Frames are enabled; the CMD_FT bit is set.
   b) Length >= 9
   c) Source or Destination Address are included, else it must be a coordinator (PANCOORD = 1) in which case the PANID must equal MACPANID[15:0]. If the device is NOT a PAN coordinator, the Destination Addressing Mode must be 2 or 3, and destination address-matching rules apply.

If a frame is rejected, the transceiver will stay enabled as a receiver, and start searching for a new frame after the rejected frame has been received.

## 6.6 VCO Frequency Synthesizer (Channel Frequency)

The VCO frequency synthesizer uses a Fractional-N (Frac-N) PLL divider chain. The MC13237 operates in the 2.4 GHz ISM band, covering 16 channels and uses 5 MHz of spacing between each channel. The synthesizer VCO output frequency is referenced as LO1 and must be programmed to the channel frequency. The Frac-N has fractional and integer components that must be programmed properly to perform a transceiver operation on a particular channel.

- The integer setting is LO1_INT[7:0], Register LO1_INT
- The fractional setting is LO1_FRAC[15:0] Registers LO1_FRAC0 – LO1_FRAC1

Table 6-6 lists LO1_INT and LO1_FRAC settings for the sixteen standard 802.15.4 Channels.

**Table 6-6. 802.15.4 Channel LO1 Settings**

| 802.15.4 Channel Number | Frequency (MHz) | $([N+1].frac)_{dec}$ | Integer Setting LO1_INT[7:0] (Dec / Hex) | Fractional Setting LO1_FRAC[15:0] (Dec / Hex) |
|---|---|---|---|---|
| 11 | 2405 | 75.15625 | 74 / 0x4A (default) | 10241 / 0x2801 (default) |
| 12 | 2410 | 75.31250 | 74 / 0x4A | 20481 / 0x5001 |
| 13 | 2415 | 75.46875 | 74 / 0x4A | 30721 / 0x7801 |
| 14 | 2420 | 75.62500 | 74 / 0x4A | 40961 / 0xA001 |
| 15 | 2425 | 75.78125 | 74 / 0x4A | 51201 / 0xC801 |
| 16 | 2430 | 75.93750 | 74 / 0x4A | 61441 / 0xF001 |
| 17 | 2435 | 76.09375 | 75 / 0x4B | 06145 / 0x1801 |
| 18 | 2440 | 76.25000 | 75 / 0x4B | 16385 / 0x4001 |
| 19 | 2445 | 76.40625 | 75 / 0x4B | 26625 / 0x6801 |

Table 6-6. 802.15.4 Channel LO1 Settings

| 802.15.4 Channel Number | Frequency (MHz) | ([N+1].frac)$_{dec}$ | Integer Setting LO1_INT[7:0] (Dec / Hex) | Fractional Setting LO1_FRAC[15:0] (Dec / Hex) |
|---|---|---|---|---|
| 20 | 2450 | 76.56250 | 75 / 0x4B | 36865 / 0x9001 |
| 21 | 2455 | 76.71875 | 75 / 0x4B | 47105 / 0xB801 |
| 22 | 2460 | 76.87500 | 75 / 0x4B | 57345 / 0xE001 |
| 23 | 2465 | 77.03125 | 76 / 0x4C | 02049 / 0x0801 |
| 24 | 2470 | 77.18750 | 76 / 0x4C | 12289 / 0x3001 |
| 25 | 2475 | 77.34375 | 76 / 0x4C | 22529 / 0x5801 |
| 26 | 2480 | 77.50000 | 76 / 0x4C | 32769 / 0x8001 |

## 6.7 Event Timer Block

The MC13237 transceiver contains an Event Timer block that provides programmable timer delays and is also used in conjunction with the sequence manager. A simplified block diagram is shown in Figure 6-9.

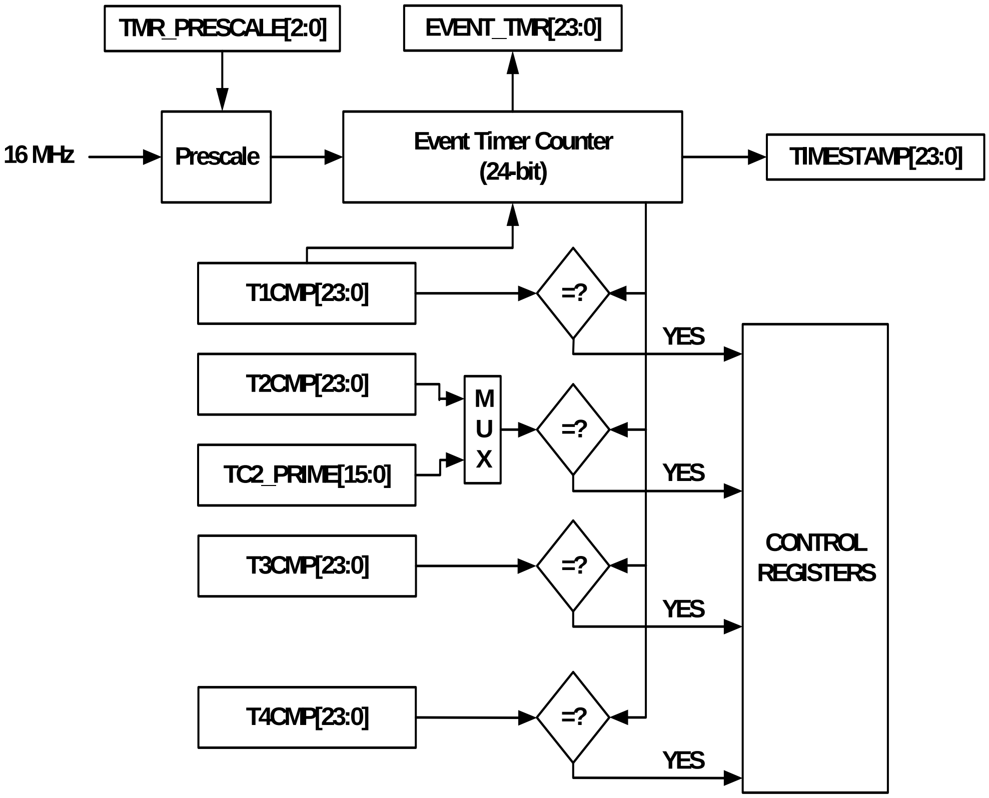


Figure 6-9. Event Timer Block Diagram

The Event Timer consists of a prescaler and a 24-bit counter which increment whenever the reference clock is operating. Interrupts to the MCU may be generated when the “current time” of the counter

(EVENT_TMR[23:0]) matches several timer comparator registers. The current time (counter value) is accessible at any time and can also be over-programmed.

The Event Timer provides the following functions:

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

### 6.7.1 Event Timer Time Base

The Event Timer’s base clock is derived from a programmable prescaler which is clocked at 16 MHz derived from the 32 MHz reference source. The prescaler provides different counter input frequencies from 500 kHz down to 15.625 kHz. The prescaler and the Event Timer only increment when the reference oscillator is active. The field TMR_PRESCALE[2:0] in the MACFRAMEVER Register 9 (Section 6.13.29, “MAC Frame Version Compatibility and Timer Prescale Register (FMR_REV_TMR)”) establishes the base clock frequency as shown in Table 6-7.

**Table 6-7. Event Timer Prescaler Settings**

| MACFRAMEVER[7:5], TMR_PRESCALE[2:0] | Event Timer Base Clock | Maximum Event Timer Duration |
|---|---|---|
| 000 | reserved | NA |
| 001 | reserved | NA |
| 010 | 500 kHz | 33.554 seconds |
| 011 (default) | 250 kHz | 67.109 seconds |
| 100 | 125 kHz | 134.218 seconds |
| 101 | 62.5 kHz | 268.436 seconds |
| 110 | 31.25 kHz | 536.871 seconds |
| 111 | 15.625 kHz | 1073.742 seconds |

The 24-bit counter automatically rolls over upon reaching its maximum value, and the corresponding maximum possible Event Timer durations are also provided in Table 6-7.

### 6.7.2 Setting Current Time

“Current Time” is defined as the value of the Event Timer internal counter. The current time is programmable, and is set to zero by a reset condition. Current time advances from zero at the selected prescale clock rate and the counter rolls over to zero after reaching its maximum value.

To set current time, the T1CMP[23:0] Registers are used (see Section 6.13.20, “Event Timer Comparator 1 Registers (T1CMP0, T1CMP1, and T1CMP2)”. Programming “current time” is accomplished by:

1. Programming the desired 24-bit value value into the T1CMP[23:0] field

2. Writing a one to the TMRLOAD bit of the CNTRL1 Register (see Section 6.13.3, “Transceiver Control Register 1 (CNTRL1)”. The TMRLOAD bit is self-clearing and always reads as zero.

### 6.7.3 Reading Current Time

The current value of the Event Timer can be read from the Event Timer [23:0] Registers (see Section 6.13.14, “Event Timer Registers (EVENT_TMR0, EVENT_TMR1, and EVENT_TMR2)”. Reading the EVENT_TMR2 Register (least significant byte) latches the “current time” in the 24-bit field. This register should be read first, followed by reads of EVENT_TMR1 Register (next most significant), and EVENT_TMR0 Register (most significant), in that order. The read of EVENT_TMR0 releases the latched value.

### 6.7.4 Receive Packet Timestamp

The MC13234/MC13237 latches a “Timestamp” of current time whenever a packet is received and the value is stored in the TIMESTAMP [23:0] Registers (see Section 6.13.15, “Time Stamp Registers (TIMESTAMP0, TIMESTAMP1, and TIMESTAMP2)”. This timestamp value is latched at the beginning of a receive packet after detect of the SFD. The timestamp remains latched until another packet is received, at which point value is updated and re-latched.

### 6.7.5 Event Timer Comparators

The MC13237 incorporates four 24-bit programmable fields that compare to the Event Timer’s “current time”. The intent of these compares is to enable the host to schedule events relative to the “current time”. When a match between the “current time” and any one of the four timer compare values occurs (when enabled), a corresponding status flag is set. An interrupt request can be generated from the valid status.

Table 6-8 lists the applicable registers and control bits for the comparators. A typical procedure for programming a compare operation includes:

1. Disable comparator (TMRxCMP_EN = 0) if required – this is the default condition.
2. Read current time from Registers EVENT_TIMER0 – EVENT_TIMER2.
3. Add desired offset to current time to get “compare time”- the offset value will be determined by the prescale rate and the desired time delay.
4. Load TxCMP0 – TxCMP2 registers with “compare time”
5. Enable interrupt request if required (TMRxIRQ = 0) – disabled is the default condition
6. Enable comparator and interrupt; set TMRxCMP_EN = 1.
7. Count reaches TxCMP value – status becomes valid and interrupt request asserted.
8. Comparator should be disabled as part of interrupt service routine.

**Table 6-8. Event Timer Comparator Registers**

| Comp | Field | Registers | Enable Bit (CNTRL4 Reg) | Status Bit | Mask Bit (CNTRL4 Reg) | Comments |
|---|---|---|---|---|---|---|
| 1 | T1CMP[23:0] | T1CMP0<br>T1CMP2 | TMR1CMP_EN | TMR1IRQ | TMR1MSK | Use to load new counter value. UseTMRLOAD bit |
| 2 | T2CMP[23:0] | T2CMP0<br>T2CMP2 | TMR2CMP_EN | TMR2IRQ | TMR2MSK | Basic 24-bit mode |
|  | TC2_PRIME[15:0] | TC2_PRIME0<br>TC2_PRIME1 | TMR2CMP_EN | TMR2IRQ | TMR2MSK | Optional 16-bit mode. Use TC2'EN |
| 3 | T3CMP[23:0] | T3CMP0<br>T3CMP2 | TMR3CMP_EN | TMR3IRQ | TMR3MSK | Optional sequence abort function. Use TC3TMOUT bit |
| 4 | T4CMP[23:0] | T4CMP0<br>T4CMP2 | TMR4CMP_EN | TMR4IRQ | TMR4MSK |  |

There are additional special features for the comparators:

1. Timer Comparator 1 (T1CMP[23:0]) is also used to set the current time counter – the comparator function should be disabled. After T1CMP[23:0] is written for the desired current time, writing a one to the TMRLOAD bit of the CNTRL1 Register loads the counter with the new time.
2. Timer Comparator 2 / TC2_PRIME[15:0] Registers – the two TC2_PRIME registers hold a 16-bit compare value used by Event Timer Comparator 2 for a special sequences. The TC2'EN control bit in register CNTRL2 enables or disables the 16-bit versus the 24-bit mode for Comparator 2; this bit must be a "1" to enable TC2_PRIME.
3. Timer Comparator 3 – T3CMP can be used as a "bracketing" timer, i.e., to abort certain sequences after a software-determined time-out period. See Section 6.7.5.1, "Timer-Triggered Transceiver Sequence Events".

### 6.7.5.1 Timer-Triggered Transceiver Sequence Events

The event timer comparators T2CMP and T3CMP can optionally be used with the Sequence Manager:

1. TC2CMP can be used to initiate a sequence at a precise time – This operation is controlled by the TMRTRIGEN control bit, see Section 6.9, "Supported Sequences".
2. T3CMP can be used to abort transceiver sequences – This time-out mechanism is enabled by setting the TC3TMOUT control bit in CNTRL2 register to 1. If TC3TMOUT = 0, then the abort mechanism is disabled, and timer TC3 functions merely as a software timer. The T3CMP hardware-abort functions are summarized in the Table 6-9.

**Table 6-9. T3CMP Comparator Sequence Abort Function**

| SEQUENCE | TC3TMOUT=0 | TC3TMOUT=1 |
|---|---|---|
| I | S/W timer | S/W timer |
| R | S/W timer | Abort sequence |
| T | S/W timer | S/W timer |
| C | S/W timer | S/W timer |

Table 6-9. T3CMP Comparator Sequence Abort Function

| SEQUENCE | TC3TMOUT=0 | TC3TMOUT=1 |
|---|---|---|
| TR | S/W timer | Abort sequence, (during receive operation only) |
| CCCA | S/W timer | Abort sequence |

## 6.8 Sequence Manager

The Sequence Manager is a hardware state machine which controls the transceiver timing and management for all transmit, receive, and CCA operations which minimizes the application memory footprint, off loads the CPU, and allows operations not otherwise possible.

Sequences can be initiated by software directly, or can be initiated via a hardware timer. The sequence parameters are programmed prior to initiating the sequences and include:

- Whether CCA is required ahead of a transmit sequence
- Whether an automatic Ack frame should be transmitted after a receive sequence
- Slotted vs. unslotted mode
- Type of CCA (e.g., Energy Detect, CCA Mode 1, or CCA Mode 2)
- Whether the device is a PAN Coordinator or not

**NOTE**

The sequence manager supports some of the more complex features of the IEEE 802.15.4 Standard including beacon-enabled communication and guaranteed time slots. It is important that the user be familiar with these concepts:

- See Appendix A, “IEEE 802.15.4 PHY Messaging Overview” for a brief overview of beacon-enabled communication.
- For more in-depth information , reference the IEEE 802.15.4 Standard, *“Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”, IEEE Std 802.15.4™-2006*

### 6.8.1 Features

The sequence manager provides the following features:

- Supports complex sequences
- Unburdens software by automating many transceiver operations
- New data-polling hardware enables critical-timing sequences
- Includes new continuous CCA for low-latency-transmit applications
- Includes capability to conditionally perform CCA prior to every transmit operation
- Includes capability to conditionally transmit an Ack frame after every receive operation
- Supports software- or timer-initiated sequences
- Fully supports 2003 and 2006 versions of the 802.15.4 standard

- Supports slotted and unslotted modes
- Supports beacon-enabled and non-beacon-enabled networks
- Supports orderly transceiver shutdown on PLL unlock event
- Supports timer-based sequence abort

### 6.8.2 Functional Overview

The Sequence Manager controls and automates all transceiver sequences.

- Software selects the desired sequence, sets the general parameters, and optionally configures a hardware timer to trigger the sequence at a precise time in the future.
- Subsequently, the MCU can enter a low-power state, or tend to other activities, while the sequence manager state machine conducts the programmed transceiver operations.
- When the sequence completes and a status is set, an interrupt request can alert the MCU. At this point the MCU can check the completion status of the sequence, fetch received data from RAM (if any), and then prepare the next sequence.

Sequences can be simple transceiver operations. For example, transmit one frame or perform a CCA operation on a channel. Alternatively, the sequence manager supports complex sequences, which use simple operations as building blocks, strung together in a back-to-back fashion, with precise timing intervals between operations. For example, a “TR” sequence calls for a transmit frame followed by a receive frame. For complex sequences, interrupts can optionally be generated at the completion of each stage of the sequence, providing greater visibility for the MCU into the time line of the sequence. Software can opt to abort a sequence at any time; when this happens, the sequence manager will return to its IDLE state in an orderly fashion.

The sequence manager is responsible for the implementation of the Wireless Physical Layer (PHY) specifications described in the IEEE 802.15.4 Standard. The timing of the basic operations is hard wired and fixed. The higher-level timing parameters, such as the interval between operations in a complex sequence, are largely programmable, except for those defined as constants in the IEEE 802.15.4 standard.

**NOTE**

> Implementation of the full IEEE 802.15.4 Standard MAC sublayer services requires software applications code that interfaces to and builds upon the embedded hardware services. Freescale provides the complete MAC solution as part of its BeeKit codebases. As a result, the descriptions for the control and use of the sequence manager are presented primarily for understanding as opposed to an in-depth applications guide. The user is directed to use the supplied codebase(s).

### 6.8.3 Control Model Summary

The control and status fields associated with the sequence manager are shown in Table 6-10.

**Table 6-10. Sequence Control Model**

| Register Field | Register Name | Description |
|---|---|---|
| **Control Fields** | | |
| XCVSEQ[2:0] | CNTRL1 [2:0] | Sequence select field — writing a command to this field will start execution of the sequence unless a timer-triggered mode is enabled. See Table 6-11 for coding and supported sequences |
| TMRTRIGEN | CNTRL1 [7] | Timer 2 Trigger Enable — this bit enables the timer-triggered mode. When enabled, execution of the sequence is delayed until a Timer Comparator 2 match is valid to initial the sequence |
| CCABFRTX | CNTRL1 [5] | CCA Before TX — this bit enables a CCA cycle prior to a transmit. Applies only to T and T/R sequences |
| SLOTTED | CNTRL1 [6] | Slotted Mode — this bit enables slotted mode for beacon-enabled networks. Applies only to T, T/R, and R sequences. |
| RXACKRQD | CNTRL1 [4] | Receive Acknowledgement Required — used only with the T/R sequence, this bit enables the option where an Ack frame is expected to be received immediately following a transmission. |
| AUTOACK | CNTRL1 [3] | Auto Acknowledge Enable — used only with the R and T/R sequences, this bit enables the option where the sequence manager will send (transmit) a hardware generated Ack frame following a received frame that requires an immediate Ack response |
| TC3TMOUT | CNTRL2 [6] | Timer Comparator 3 Time-out — this bit enables using T3CMP as a time-out for a receive cycle. Applies only to R and T/R sequences |
| CCATYPE [1:0] | CNTRL2 [4:3] | CCA Type Select — this field selects the algorithm used during a CCA cycle. Used with T, T/R, C and CCCA sequences |
| PANCORDNTR | CNTRL2 [5] | Pan Coordinator Enable — this bit enables the device as a Pan Coordinator and impacts receive filtering during an R or T/R sequence |
| PROMISCUOUS | CNTRL2 [1] | Promiscuous Mode Enable — this bit disables all receive filtering except length and CRC checking during a R or T/R sequence |
| TC2’EN | CNTRL2 [0] | TC2’ Enable — this bit selects 24-bit (T2CMP) versus 16-bit (TC2’) compare when Timer 2 is used to initiate start of a sequence |
| CRC_MSK | CNTRL3 [7] | CRC Redundancy Check — this bit can disable the CRC on a received frame |
| **Status Fields** | | |
| CRCVALID | STATUS1 [7] | Code Redundancy Check Valid — this flag indicates the compare result between the FCS field, in the just received frame, and the internally calculated CRC value |
| CCA | STATUS1 [6] | CCA Valid — this flag reports the result of a CCA operation |
| SRCADDR | STATUS1 [5] | Source Address Indication — this flag reports a valid incoming data request frame with matching source address and valid source address check sum |
| PI | STATUS1 [4] | Poll Indication — this flag reports a valid incoming data request frame |
| TMR3IRQ[1] | STATUS1 [2] | Timer Comparator 3 Flag — indicates a valid T3CMP compare. Can be used with using T3CMP for read sequence time-out |
| TMR2IRQ[1] | STATUS1 [1] | Timer Comparator 2 Flag — indicates a valid T2CMP compare. Can be used with using T2CMP for timer-triggered sequence execution |

Table 6-10. Sequence Control Model

| Register Field | Register Name | Description |
|---|---|---|
| RX_FRM_PEND | STATUS2 [7] | Status of the frame pending bit of the frame control field for the just received packet |
| LO1_UNLOCK_IRQ | STATUS2 [6] | Local Oscillator Unlock — this flag A indicates an unlock condition in the RF PLL |
| FILTERFAIL_IRQ | STATUS2 [5] | Receiver Packet Filter Fail — this flag indicates that the just received packet has been rejected due to filtered elements within the packet |
| CCAIRQ | STATUS2 [3] | Clear Channel Assessment — this flag indicates completion of a CCA operation. |
| RXIRQ | STATUS2 [2] | Receive — this flag indicates the completion of a receive operation |
| TXIRQ | STATUS2 [1] | Transmit — this flag indicates the completion of a transmit operation |
| SEQIRQ | STATUS2 [0] | Sequence — this flag indicates the completion of a sequence operation |
| CCAFNL[7:0] | CCAFNL [7:0] | Clear Channel Assessment Average — final result of a CCA operation or LQI from a receive operation. In a T/R sequence, the final value is the LQI, not CCA. |

[1] TMR1IRQ and TMR4IRQ also are present but do not interact with the Sequence Manager

## 6.9 Supported Sequences

The following sections describe the control and execution of supported sequences in more detail.

The supported sequences are shown in Table 6-11.

Table 6-11. Transceiver Sequences

| XCVSEQ | Sequence | Description |
|---|---|---|
| 0 | I | Idle |
| 1 | R | Receive Sequence – conditionally followed by a TxAck |
| 2 | T | Transmit Sequence |
| 3 | C | Standalone CCA |
| 4 | TR | Transmit /Receive Sequence – transmit unconditionally followed by either an R or RxAck |
| 5 | CCCA | Continuous CCA – sequence completes when channel is idle |
| 6 | Reserved | |
| 7 | Reserved | |

**NOTE**

In the sequence descriptions, status bits are set as appropriate to the completion of a condition. Although not specifically stated, an interrupt request can be initiated with setting of a given status, the the IRQ is enabled.

### 6.9.1 Initiating a Sequence

A sequence can be initiated directly (non-timer mode) or using a timer-triggered mode:

- For non-timer mode (TMRTRIGEN bit must be 0) – All sequences can be initiated immediately by writing the desired sequence command to XCVSEQ (CNTRL1) during IDLE state to activate each new sequence. Hardware detects the write to this register during IDLE, and initiates the sequence. The XCVSEQ bits of CNTRL1 always read back what was previously written, they do not necessarily reflect the state of the Sequence Manager. Because hardware detects the write to the XCVSEQ register during IDLE, the XCVSEQ field must be set to IDLE after the sequence completes (SEQIRQ interrupt), before initiating a new sequence.
- For timer-trigger mode, the initiation of a sequence at a precise time in the future can be programmed via Timer Comparator 2 (TMRTRIGEN bit must be 1) -
  - The desired start time is determined by the Timer Comparator 2 and enabled by setting the TMRTRIGEN bit.
  - Either the 24-bit T2CMP register or the 16-bit TC2PRIME register can be used for the compare value (this is determined by the TC2'EN bit of the CNTRL2 register, see Section 6.13.4, "Transceiver Control Register 2 (CNTRL2)"). The desired start time is written to the selected register.
  - The desired sequence command is written to the XCVSEQ field and the sequence will commence when the timer compare is valid.
  - After the sequence completes (SEQIRQ interrupt), the XCVSEQ field must be set to IDLE (0x0), before initiating a new sequence and the Timer Comparator 2 register must be re-written to arm a new sequence. To be specific, at least the LS byte of either T2CMP or TC2PRIME must be written to re-arm the sequence manager for a new sequence. The order of writing the command to CNTRL1 or the timer register is not important, as long as a timer-match does not occur prior to the completion of this programming.

## 6.9.2 Aborting a Sequence

Several means can abort a sequence.

### 6.9.2.1 Writing an IDLE Command to XCVSEQ

While a sequence is ongoing, software can abort the sequence at any time by writing IDLE (0x0) to XCVSEQ, and the sequence manager state machine will respond with an orderly return to the SEQ_IDLE state, and a SEQIRQ status will be generated.

While a sequence is underway, software should not attempt to change XCVSEQ (other than aborting the sequence as previously described); the sequence manager will ignore all mid-sequence attempts to change XCVSEQ.

**NOTE**

The sequence manager control logic is designed such that the command field XCVSEQ must always be written to IDLE between sequences. This is true for both successful completion of a sequence and an aborted sequence. Upon end of the sequence, the XCVSEQ must be written to IDLE before writing the next desired sequence.

#### 6.9.2.2 Timer Comparator 3 Time-out

Timer Comparator 3 can be used as a “bracketing” timer, i.e., to abort certain sequences after a software-determined time-out period. This time-out mechanism is enabled by setting the TC3TMOUT control bit in CNTRL2 Register to 1, and the comparator must be enabled by the TMR3CMP_EN control bit in CNTRL4 Register.

#### 6.9.2.3 RF PLL Unlock Condition (LO1 Unlock)

All sequences can be aborted by an RF PLL unlock detection; referred as LO1 unlock. After the operation warm-up time, an unlock detection aborts the sequence at that point and an unlock status is set.

### 6.9.3 Sequence I (IDLE)

When the IDLE command is written to XCVSEQ, the sequence manager goes to its IDLE state. If not already in idle, the sequence manager executes an orderly return to IDLE state. A SEQIRQ status is then issued.

- Writing IDLE value to XCVSEQ is the proper way to abort a sequence.
- A write of IDLE to XCVSEQ should always be issued between two consecutive sequences including the back-to-back execution of the same sequence.

### 6.9.4 Basic Receive Sequence – Sequence R (Receive)

Sequence R is the basic receive sequence and can be used to receive all types of 802.15.4 PHY- and MAC-compliant frames, including reserved frame types. Using Sequence R, all receive frames which pass frame-filtering rules and CRC check, are transferred to RAM.

#### 6.9.4.1 Sequence R Basic Execution

The basic execution of a successful Sequence R is as follows:

1. Optional timer initiated -
   — MCU writes T2CMP compare value to the desired initiation time
   — Sets TMRTRIGEN
2. MCU initializes RXD_ADR_PNTR to desired start of RX buffer
3. MCU writes XCVSEQ = R
4. Optional timer initiated – Wait for T2CMP match (if TMRTRIGEN=1)
5. SEQ_MGR executes RxWarmup and enters receive mode.
6. Preamble Detected
7. SFD Detected
8. DMA RX Address Pointer initialized to RXD_ADR_PNTR
9. FrameLength (FLI) octet received and transferred to RAM
10. *N* additional octets received (*N* = FrameLength) and transferred to RAM.

**NOTE**

If FrameLength > MAXFRAMELENGTH (127), then N = MAXFRAMELENGTH.

11. If frame-filtering rules and CRC check pass:
    - RXIRQ status issued
    - SEQ_MGR executes RxWarmdown
    - If received Frame Control Field indicates AckRequest = 1
        - SEQ_MGR executes TxWarmup
        - an Ack frame is transmitted using the received Sequence Number
        - SEQ_MGR executes TxWarmdown
12. If frame-filtering rules or CRC check fail:
13. SEQIRQ status issued

**NOTE**

Reception of Acknowledge frames is not recommended using Sequence R. This is because reception of an Acknowledge frame, usually follows a transmitted frame, with a designated Sequence Number, so that the received Acknowledge frame can be verified for the matching Sequence Number. In a standalone Sequence R, there is no Sequence Number to verify against, so any Acknowledge frame is merely transferred (DMAed) to RAM. The appropriate way to receive Acknowledge frames is with Sequence TR with the RXACKRQD bit asserted.

#### 6.9.4.2 Sequence R T3CMP Time-out

After the Sequence R receiver warm-up, timer T3CMP can be enabled as a “bracketing” timer. If enabled and a T3CMP timer match occurs (no incoming frame is detected), the sequence manager will warm down the receiver and return to SEQ_IDLE state.

#### 6.9.4.3 Sequence R Exceptions

Programmed options and incoming RX frame parameters affect Sequence R execution. Programmed options that impact Sequence R include:

1. AUTOACK – when enabled, Sequence R transmits a hardware generated TX Ack frame after the received frame if all necessary conditions are met.
2. PROMISCUOUS – when enabled, all filtering is disabled except for frame length check, and CRC check is conditional on the CRCMSK. This mode also disables TxACK if enabled.

Frame filtering is programmable and the RX frame will normally be accepted or rejected on the basis of the filter options. The above options provide exceptions and additions to the normal flow as shown in Table 6-12.

Table 6-12. Sequence R Execution versus Programmed Options

| Option Select | | Description |
|---|---|---|
| PROMIS-CUOUS | AUTOACK | |
| Enabled | Don't Care | Promiscuous mode enabled —<br>• Disables all filtering except frame length check, and CRC check is conditional<br>• AUTOACK option is disabled – ro received frame will be acknowledged |
| Disabled | Disabled | Standard reception; no TxACK— when a frame is received, it is validated against all enabled filtering, and no TxAck will be returned. |
| Disabled | Enabled | Standard reception; TxACK enabled —<br>• When a frame is received, it is validated against all enabled filtering<br>• Hardware-generated TxAck is returned if required (uses Rx frame Sequence Number). See Table 6-13.<br>- AckRequest bit in Rx frame FrameControlField = 1 (ack requested)<br>- No CCA cycle is executed before the TxAck<br>- SLOTTED not enabled – RX-to-TX turnaround time is 192 µs max<br>- SLOTTED enabled; TxAck meets required symbol timing |

When a TxAck frame is transmitted, its Frame Control Field is generated in hardware as defined in Table 6-13. The Sequence Number for the Ack packet, will be the Sequence Number obtained from the previously received frame.

Table 6-13. Frame Control Field for Hardware-generated Auto-TxAck Frame

| FCF Field | Value |
|---|---|
| FrameType | Acknowledge |
| SecurityEnabled | 0 |
| FramePending | • If SRCADDR_EN=1, FramePending gets the value of FramePendingFlag from Source Address Matching hardware<br>• If SRCADDR_EN=0, FramePending gets the value of ACK_FRM_PND. |
| AckRequest | 0 |
| PanIDCompression | 0 |
| Reserved (bits 7-9) | 000 |
| DstAddrMode | 0 |
| FrameVersion | copy FrameVersion from the previously-received frame |
| SrcAddrMode | 0 |

### 6.9.5 Basic Transmit Sequence – Sequence T (Transmit)

Sequence T is the basic, standalone transmit sequence and can be used to transmit all types of 802.15.4 PHY- and MAC-compliant frames. However, it is not recommended for transmitting an Ack frame nor for transmitting frames that expect a return Ack, see NOTE.

**NOTE**

> Sequence T is not recommended for transmission of Acknowledge frames. This is because transmission of an Ack frame usually follows a received frame with a designated Sequence Number. The recommended method is use of Sequence R with the AUTOACK bit asserted where the sequence manager automates the transmission of an Ack frame generated by hardware following a received frame. However, the sequence manager hardware does not prohibit generation of Ack frames using Sequence T.
>
> Sequence T is not recommended for transmitting frames that expect a return Acknowledge frame. Use a T/R sequence instead.

Sequence T allows for the insertion of 1 or 2 CCA (clear channel assessment) measurements prior to transmission, to ensure that the selected channel is idle. The CCA-before-TX feature is governed by two register bits, CCABFRTX and SLOTTED. All CCA measurements must indicate channel-idle, in order for the sequence manager to initiate the transmission; if the channel is determined by CCA to be busy, the sequence manager will terminate the sequence without a transmission. The number of CCA measurements attempted by the sequence manager, and the timing of the measurements, is shown in Table 6-14.

**Table 6-14. CCA Cycle Control and Timing**

| CCABFRTX | SLOTTED | Number of CCA measurements | Timing of Initiation of CCA measurement #1 | Timing of Initiation of CCA measurement #2 | Timing of First Bit of Transmitted Frame (assumes channel idle) |
|---|---|---|---|---|---|
| 0 | X | 0 | | | Immediately follows TxWarmup |
| 1 | 0 | 1 | Immediately follows RxWarmup | | Immediately follows RxPartialWarmdown and TxPartialWarmup |
| 1 | 1 | 2 | Immediately follows RxWarmup | 320us after Initiation of CCA measurement #1 | 320us after Initiation of CCA measurement #2 |

Only CCA Type 1 or CCA Type 2 should be used with Sequence T (and Sequence TR). Software must configure the CCATYPE bits prior to initiating the auto sequence.

### 6.9.5.1 Sequence T Basic Execution

The basic execution of a successful Sequence T is as follows:

1. Optional timer initiated -
   — MCU writes T2CMP compare value to the desired initiation time
   — Sets TMRTRIGEN
2. Optional use of CCA cycle(s):
   — MCU sets CCABFRTX if one or more CCA’s are required prior to transmit
   — MCU sets SLOTTED if in slotted mode (2 CCA’s will be attempted)
3. MCU initializes TXD_ADR_PNTR to start of stored TX buffer
4. MCU writes XCVSEQ = T
5. Optional timer initiated – Wait for T2CMP match (if TMRTRIGEN=1)

6. If CCABFRTX=1:
   - — SEQ_MGR executes RxWarmup
   - — SEQ_MGR initiates CCA measurement (takes 128us)
   - — if CCA indicates channel busy: sequence terminates and CCAIRQ & SEQIRQ status issued
   - — otherwise, if CCA indicates channel idle:

   a) if SLOTTED=0, SEQ_MGR issues a CCAIRQ status and executes a RxPartialWarmdown

   b) if SLOTTED=1, SEQ_MGR initiates a 2nd CCA, 320us after initiating 1st CCA.
      - – if SLOTTED=1, and channel busy, sequence terminates, CCAIRQ status & SEQIRQ status issued
      - – if SLOTTED=1, and channel idle, SEQ_MGR issues a CCAIRQ status and executes a RxPartialWarmdown
7. If SLOTTED=1, SEQ_MGR waits until 320us elapses after 2nd CCA initiation
8. SEQ_MGR executes TxPartialWarmup.
9. DMA TX Address Pointer initialized to TXD_ADR_PNTR
10. Preamble transmitted
11. SFD transmitted
12. FrameLength octet is fetched from first byte of TX buffer from RAM and transmitted
13. CRC engine is reset
14. *N* additional octets are fetched from RAM and transmitted (*N* = FrameLength – 2)
15. Each octet that is transmitted, is shifted through the CRC engine
16. After *N*th octet transmitted, FCS is available from CRC engine.
17. FCS lower octet transmitted
18. FCS upper octet transmitted
19. SEQ_MGR issues TXIRQ status
20. SEQ_MGR executes TxWarmdown
21. SEQ_MGR issues SEQIRQ status

**NOTE**

Timer T3CMP is not used with Sequence T.

### 6.9.5.2 Sequence T Exceptions

The programmed CCA and slotted options affect Sequence T execution.

- The use of one or two CCA cycles is determined by the options
- The timing of the CCA cycles is impacted by the slotted option
- The slotted option is used in beacon-enabled networks
- The CCAIRQ status is always issued at completion of the CCA cycle(s)
- A “busy” result for a CCA cycle terminates the sequence; any retries are controlled via software.

### 6.9.6 Basic Sequence C (Standalone CCA)

Sequence C is the basic standalone CCA sequence where the sequence manager executes a Clear Channel Assessment (CCA) cycle(s).

- Any of the CCA algorithms can be selected
- A CCA status is not reported for an ED CCA cycle, it has no meaning as an ED cycle is meant as a channel scan function
- This sequence does not support slotted CCA timing for beacon-enabled mode

The result of the CCA measurement is reported back to software in the CCA bit. The CCA bit indicates either a busy channel (1), or an  idle channel (0).

#### 6.9.6.1 Sequence C Basic Execution

The basic execution of a successful Sequence C is as follows:

1. Optional timer initiated -
   - MCU writes T2CMP compare value to the desired initiation time
   - Sets TMRTRIGEN
2. MCU initializes CCATYPE
3. MCU writes XCVSEQ = C
4. Optional timer initiated – Wait for T2CMP match (if TMRTRIGEN=1)
5. SEQ_MGR executes RxWarmup
6. SEQ_MGR initiates CCA (using selected CCATYPE) by signalling the receiver
7. SEQ_MGR waits for CCA to complete
8. CCA bit is set to the CCA status (1 = Busy, 0 = Idle) except for CCATYPE = ED
9. CCAIRQ status issued
10. SEQ_MGR executes RxWarmdown
11. SEQIRQ status issued

**NOTE**

- Timer T3CMP is not used with Sequence C.
- The CCA cycle timing is consistent with Figure 6-3 and Figure 6-4.

### 6.9.7 Sequence T/R (Transmit/Receive)

Sequence TR is a very useful combination Transmit/Receive sequence. The sequence is executed as a concatenation of a transmit operation followed by a receive operation, with a minimum TX-to-RX turnaround time in between.

#### 6.9.7.1 Sequence T/R Execution

There are 2 permutations of Sequence TR, depending on the RXACKRQD bit. For both permutations, the Sequence T which constitutes the first half of a Sequence TR, is identical to the standalone Sequence T

(XCVSEQ=2). This means that the transmit operation can be slotted or unslotted, and can be preceded by 1 or 2 CCA measurements, depending on the state of the CCABFRTX and SLOTTED bits.

The permutations are differences in the R sequence as determined by RXACKRQD control bit:

- RXACKRQD = 0 – then Sequence TR is executed as a Sequence T followed by a Sequence R, with a minimum TX-to-RX turnaround time in between. The Sequence R is identical to the standalone Sequence R (XCVSEQ = 1). This means that the receive operation can be followed by an automatic, hardware-generated transmit Acknowledge frame, if all the necessary conditions are met.
- RXACKRQD = 1 – then Sequence TR is executed as a Sequence T followed by a Receive-Acknowledge-Only frame. This type of receive operation is special, and not the same as a standalone Sequence R. The Ack-only receive operation filters all incoming frames, looking only for an Acknowledge frame whose Sequence Number matches the Sequence Number which was transmitted in the Sequence T portion of the sequence. All non-matching frames are discarded, and after each non-matching frame, the sequence manager will return the receiver to preamble-detect mode. This receive operation will continue until the matching Ack frame is received. Because this is a Receive-Acknowledge-Only operation, and not a Sequence R, no receive octets are transferred to RAM. The sequence ends with a RXIRQ interrupt, which indicates that the matching Ack frame was successfully received.

If the transmit operation of a Sequence TR is requesting an Acknowledge frame of the receiving end device (Frame Control Field: AckRequest=1), then Sequence TR with RXACKRQD=1 is the appropriate procedure. Using Sequence TR with *RXACKREQ = 0* is *not* recommended for this scenario, because there is no Sequence Number matching.

Conversely, if the transmit operation of a Sequence TR is *not* requesting an Acknowledge frame of the receiving end device (Frame Control Field: AckRequest = 0), then Sequence TR with RXACKRQD = 1 should *never* be used, because a receive acknowledge frame will not be forthcoming. Instead, use Sequence TR with RXACKRQD = 0, or simply Sequence T, if no follow-up receive frame is expected.

#### 6.9.7.2 Sequence T/R T3CMP Time-out

Similar to the Sequence R usage, after the Sequence T/R receiver warm-up, timer T3CMP can be enabled as a “bracketing” timer. If enabled and a T3CMP timer match occurs (no incoming frame is detected), the sequence manager will warm down the receiver and return to SEQ_IDLE state.

#### 6.9.7.3 Sequence T/R Exceptions

See Section 6.9.4.3, “Sequence R Exceptions” and Section 6.9.5.2, “Sequence T Exceptions” for the appropriate execution for the transmit or receive exceptions.

### 6.9.8 Sequence CCCA (Continuous CCA)

Sequence CCCA is the Continuous CCA sequence. This sequence is designed to accommodate situations where channel availability may be infrequent, or low-duty-cycle, and a device needs to be able to transmit at the earliest available opportunity. During Sequence CCCA, the sequence manager repeats CCA measurements continuously until a channel-idle condition is found.

#### 6.9.8.1 Sequence CCCA Execution

For the CCAA operation:

- Only CCA Mode 1 and Mode 2 may be used.
- There is a 54us delay between CCA iterations. The actual interval between consecutive CCA measurements, in the CCCA sequence, is exactly 188us.
- After each CCA iteration, the CCA bit is set to the result of the measurement. As long as the CCA bit is high (busy) at the end of the iteration, the sequence manager will initiate a new measurement.
- Unlike Sequence C (standalone CCA), there is no CCAIRQ interrupt generated at the end of each CCA measurement, until the channel-idle condition is detected, at which point CCAIRQ is issued.
- The SLOTTED bit has no effect on this sequence.
- A Sequence CCCA is aborted by the Sequence Manager if a TMR3 match occurs, and TC3TMOUT=1.

The basic execution of a Sequence CCCA is as follows:

1. Optional timer initiated -
   - MCU writes T2CMP compare value to the desired initiation time
   - Sets TMRTRIGEN
2. MCU initializes CCATYPE to desired algorithm
3. MCU writes XCVSEQ = CCCA
4. Optional timer initiated – Wait for T2CMP match (if TMRTRIGEN=1)
5. SEQ_MGR executes RxWarmup and enters receive mode.
6. SEQ_MGR initiates CCA by signalling the receiver
7. SEQ_MGR waits for CCA to complete
8. CCA bit is set to the CCA status (1 = Busy 0 = Idle)
9. If CCA=1, initiate a new CCA measurement immediately and repeat the previous 3 steps
10. If CCA=0, CCAIRQ status is issued
11. SEQ_MGR executes RxWarmdown
12. SEQIRQ status issued

#### 6.9.8.2 Sequence CCCA T3CMP Time-out

Timer T3CMP can be enabled as a “bracketing” timer. If enabled and a T3CMP timer match occurs (no incoming frame is detected), the sequence manager will warm down the receiver and return to SEQ_IDLE state.

#### 6.9.8.3 Sequence CCCA Exceptions

Because of the repeated CCA operations:

- The CCA bit (STATUS1 register) and the CCAFNL register will change dynamically during the CCA measurement, and are only considered valid (and sampled by the Sequence Manager), at the end of the CCA measurement; therefore the user should not continuously poll CCA and CCAFNL and expect continuously-valid results, because they are not latched.

- The user can continuously monitor the Sequence CCCA status by reading the CCAIRQ status

### 6.9.9 Effect of Timer TC3 on Sequence Manager

Timer TC3 can be used as a “bracketing” timer, to abort certain sequences after a software-determined time-out period. This time-out mechanism is enabled by setting the TC3TMOUT bit to 1. If TC3TMOUT=0, then the abort mechanism is disabled, and timer TC3 functions merely as a software timer.

Timer TC3’s hardware-abort functions are summarized in the following table.

**Table 6-15. TC3 Hardware Abort Functions**

| SEQUENCE | TC3TMOUT=0 | TC3TMOUT=1 |
|---|---|---|
| I | S/W timer | S/W timer |
| R | S/W timer | Abort sequence |
| T | S/W timer | S/W timer |
| C | S/W timer | S/W timer |
| TR | S/W timer | Abort sequence, (during receive operation only) |
| CCCA | S/W timer | Abort sequence |

## 6.10 Transmitter Power Amplifier (PA) Control

The on board differential power amplifier (PA) can be programmed to set output power, see Section 6.13.25, “Power Amplifier (PA) Power Control Register (PA_PWR_CNTL)”. The PA outputs are enabled via hardware when a transmit operation occurs.

## 6.11 Hardware Pseudo-Random Number Generator

To assist in generating seeds/keys for the AES security, the MC13234/MC13237 provides hardware generation of a pseudo-random 16-bit number. The accumulator of the FracN counter used in the LO1 synthesizer is read to provide this number. Access to the FracN hardware uses an indirect addressing technique that is normally reserved for chip test using the indirect access registers described in Section 6.13.33, “Transceiver Indirect Access Registers (INDEX and DATA)”

**NOTE**

To generate the pseudo-random 16-bit number, a software function activates the synthesizer for a random time from between 1 – 10 ms and then read accesses the FracN accumulator. Freescale provides this function call as part of its various software stacks. The user is directed to use this function.

## 6.12 Transceiver Interrupt Requests (IRQs)

The transceiver components can be programmed to generate an interrupt request from a number sources and status flags. Table 6-16 lists source status bits, mask bits, and interrupt vector numbers for transceiver IRQs.

Table 6-16. Transceiver Interrupt Request Sources

| Item | Status Flag | Mask Bit | Source Description | Interrupt Clear Mechanism | Interrupt Vector No. |
|---|---|---|---|---|---|
| **General IRQ Disable** | | | | | |
| 1 | | TRCV_MSK | General control that allows status flags from the modem (transmitter/ receiver) to generate an interrupt request. This bit disables all IRQs | | |
| **RX DMA** | | | | | |
| 2 | RXWTRMRKIRQ | RX_WMRK_MSK | Received DMA byte count matches “watermark” level | Write “1” to status flag | 7 |
| **Event Timer Block** | | | | | |
| 3 | TMR1IRQ | TMR1MSK | Timer Comparator 1 value matched event timer counter | Write “1” to status flag | 6 |
| | TMR2IRQ | TMR2MSK | Timer Comparator 2 value matched event timer counter | Write “1” to status flag. This flag is shared between T2CMP and TC2’ | |
| | TMR3IRQ | TMR3MSK | Timer Comparator 3 value matched event timer counter | Write “1” to status flag | |
| | TMR4IRQ | TMR4MSK | Timer Comparator 4 value matched event timer counter | Write “1” to status flag | |
| **Transmit Function** | | | | | |
| 4 | LO1_LOCK_IRQ | LO1_UNLOCK_MSK | Local Oscillator 1 (RF PLL) has experienced an unlock condition | Write “1” to status flag | 5 |
| | TXIRQ | TXMSK | Flag indicates completion of a transmit operation | Write “1” to status flag | |
| | **SEQIRQ** | SEQMSK | Flag indicates completion of the programmed sequence | Write “1” to status flag | |

Table 6-16. Transceiver Interrupt Request Sources (continued)

| Item | Status Flag | Mask Bit | Source Description | Interrupt Clear Mechanism | Interrupt Vector No. |
|---|---|---|---|---|---|
| **Receive / CCA Function** | | | | | |
| 5 | LO1_LOCK_IRQ | LO1_UNLOCK_MSK | Local Oscillator 1 (RF PLL) has experienced an unlock condition | Write 1 to status flag | 4 |
| | RXIRQ | RXMSK | Flag indicates completion of a receive operation | Write 1 to status flag | |
| | FILTERFAIL_IRQ | FILTERFAIL_MSK | Received packet has been rejected due to enabled filtering elements | Write 1 to status flag | |
| | CCAIRQ | CCAMSK | Flag indicates completion of a CCA operation | Write 1 to status flag | |
| | SEQIRQ | SEQMSK | Flag indicates completion of the programmed sequence | Write 1 to status flag | |

**NOTE**

The LO1 unlock and SEQIRQ status vector to a different Interrupt Vector Number based on the enabled sequence:

- For a T or T/R sequence — these use vector No. 5
- For an R, C, or CCCA sequence — these use vector No. 4

## 6.13 Register Definitions

The following sections describe the transceiver control and status registers. The transceiver registers are organized in two blocks;

- Direct Page registers – more commonly accessed registers, always directly accessible
- High-Page registers – less commonly accessed registers, located at Page 0x1800

Table 6-17 lists both Direct and High Page register addresses. The register descriptions are organized by address.

Table 6-17. Transceiver Register Addresses

| Direct Page Address | Register Name | High Page Address | Register Name |
|---|---|---|---|
| 0x0040 | TXD_ADR_PNTR0 | 0x1840 | T1CMP0 |
| 0x0041 | TXD_ADR_PNTR1 | 0x1841 | T1CMP1 |
| 0x0042 | RXD_ADR_PNTR0 | 0x1842 | T1CMP2 |
| 0x0043 | RXD_ADR_PNTR1 | 0x1843 | T2CMP0 |
| 0x0044 | CNTRL1 | 0x1844 | T2CMP1 |

**Table 6-17. Transceiver Register Addresses (continued)**

| Direct Page Address | Register Name | High Page Address | Register Name |
|---|---|---|---|
| 0x0045 | CNTRL2 | 0x1845 | T2CMP2 |
| 0x0046 | CNTRL3 | 0x1846 | T4CMP0 |
| 0x0047 | CNTRL4 | 0x1847 | T4CMP1 |
| 0x0048 | SRC_CNTRL | 0x1848 | T4CMP2 |
| 0x0049 | SRC_ADDRS_SUM_DATA0 | 0x1849 | LO1_FRAC0 |
| 0x004A | SRC_ADDRS_SUM_DATA1 | 0x184A | LO1_FRAC1 |
| 0x004B | RX_WTR_MARK | 0x184B | LO1_INT |
| 0x004C | RX_BYTE_COUNT | 0x184C | PA PWR CNTL |
| 0x004D | STATUS1 | 0x184D | MACLONGADDRS0 |
| 0x004E | STATUS2 | 0x184E | MACLONGADDRS1 |
| 0x004F | Reserved | 0x184F | MACLONGADDRS2 |
| 0x0050 | CCAFNL | 0x1850 | MACLONGADDRS3 |
| 0x0051 | EVENT_TMR0 | 0x1851 | MACLONGADDRS4 |
| 0x0052 | EVENT_TMR1 | 0x1852 | MACLONGADDRS5 |
| 0x0053 | EVENT_TMR2 | 0x1853 | MACLONGADDRS6 |
| 0x0054 | TIMESTAMP0 | 0x1854 | MACLONGADDRS7 |
| 0x0055 | TIMESTAMP1 | 0x1855 | MAXFRAMELENGTH |
| 0x0056 | TIMESTAMP3 | 0x1856 | RX FRAME FILTER |
| 0x0057 | T3CMP0 | 0x1857 | MAC FRAME VER FMR REV&TMR |
| 0x0058 | T3CMP1 | 0x1858 | CCA_THRESHOLD |
| 0x0059 | T3CMP2 | 0x1859 | CCA_OFFSET_CMP |
| 0x005A | TC2 PRIME0 | 0x185A | FSM |
| 0x005B | TC2 PRIME1 | 0x185B | INDIRECT ACCESS INDEX |
| 0x005C | MACSHORTADDRS0 | 0x185C | INDIRECT ACCESS DATA |
| 0x005D | MACSHORTADDRS1 | 0x185D | Reserved |
| 0x005E | MACPANID0 | 0x185E | Reserved |
| 0x005F | MACPANID1 | 0x185F | Reserved |

### 6.13.1 Transmit Data Pointer Registers (TXD_ADR_PNTR0 and TXD_ADR_PNTR1)

The Transmit Data Pointer [12:0] Register holds the base address of the RAM buffer from which data is to be supplied for packet transmit. The 13-bit address pointer is composed of two registers:

- TXD_ADR_PNTR0 – which contains the most significant 5 bits (TXD_ADR_PNTR[12:8])
- TXD_ADR_PNTR1 – which contains the least significant 8bits (TXD_ADR_PNTR[7:0])

The DMA uses this base address to start fetching payload data for a packet transmission.

| | TXD_ADR_PNTR0 | | | | 0x0040 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | — | — | — | TXD_ADR_PNTR[12:8] | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | TXD_ADR_PNTR1 | | | | 0x0041 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | TXD_ADR_PNTR[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-10. Transmit Data Pointer Registers**

### 6.13.2 Receive Data Pointer Registers (RXD_ADR_PNTR0 and RXD_ADR_PNTR1)

The Receive Data Pointer [12:0] Register holds the base address of the RAM buffer to which data is to be sent for a packet reception. The 13-bit address pointer is composed of two registers:

- RXD_ADR_PNTR0 – which contains the most significant 5 bits (RXD_ADR_PNTR[12:8])
- RXD_ADR_PNTR1 – which contains the least significant 8bits (RXD_ADR_PNTR[7:0])

The DMA uses this base address to start writing payload data from a packet reception.

**RXD_ADR_PNTR0** **0x0042**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | — | — | — | RXD_ADR_PNTR[12:8] | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RXD_ADR_PNTR1** **0x0043**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RXD_ADR_PNTR[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-11. Receive Data Pointer Registers**

### 6.13.3 Transceiver Control Register 1 (CNTRL1)

CNTRL1 is one of several registers that control basic operation of the transceiver. Included in this register are fields for:

- Direct versus timer-initiated operation
- Slotted versus normal timing operation
- CCA operation before transmit
- Auto ACK during a T/R sequence
- Transmit auto ACK
- Transceiver operation – sets sequence manager operation mode

| | CNTRL1 | | | | 0x0044 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | TMRTRIGEN | SLOTTED | CCABFRTX | RXACKRQD | AUTOACK | XCVSEQ | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-12. Transceiver Control Register 1**

**Table 6-18. CNTRL1 Field Descriptions**

| Field | Description |
|---|---|
| TMRTRIGEN | **Timer Trigger Enable** — enables timer-triggered sequences using Timer Comparator 2<br>0 Selected transceiver operation initiated manually upon write of XCVSEQ command<br>1 Selected transceiver operation initiated via Event Timer Comparator 2 or TC2_PRIME (see TC2'EN) |
| SLOTTED | **Slotted Mode** — enables slotted mode for beacon-enabled networks. Applies only to Sequences T, TR, and R, ignored during all other sequences. Used in concert with CCABFRTX to determine how many CCA cycles are required prior to a transmit operation. Also used during R sequence to determine whether the ensuing transmit acknowledge frame (if any) needs to be synchronized to a back off slot boundary.<br>0 Normal operation<br>1 Slotted mode – specifies to the TX, CCA and AutoAck mechanism that an auto back off needs to be computed and applied to their timing.<br>• For AutoAck operations, this computed back off is only applied to the TxAck transmissions when both the SLOTTED and AUTOACK bits are enable (set to '1').<br>• In the case of a CCA sequence a SLOTTED bit setting of '1' will cause the CCA to be executed two times, with the second CCA occurring 20 symbols after the beginning of the first one. |

Table 6-18. CNTRL1 Field Descriptions

| Field | Description |
|---|---|
| CCABFRTX | **Clear Channel Assessment Before Transmit** — this bit controls the introduction of a CCA operation prior to the start of transmit operation. Applies only to Sequences T and TR, ignored during all other sequences.<br>This bit only affects sequences in which the transmit operation is the first element of the sequence. For instance a Transmit only sequence will begin with a CCA followed by TX when this bit is enabled.<br>The CCA operation will start the sequence in accordance to the result by the CCA operation. If the CCA flag indicates a clear channel (set to 0), the sequence continues to the transmit operation; otherwise it is terminated.<br>0 Disabled – no CCA executed, transmit operation begins immediately.<br>1 Enabled – at least one CCA measurement is required prior to the transmit operation (see also SLOTTED) |
| RXACKRQD | **RX Acknowledge Required** — Applies only to Sequence TR, ignored during all other sequences.<br>0 Disabled – an ordinary receive frame (not an Ack frame) follows the transmit frame; thus a simple Receive operation is executed in the TR sequence.<br>1 Enabled – a receive Ack frame follows the transmit frame. |
| AUTOACK | **Auto Acknowledge Enable** — applies only to Receive (R) sequence or T/R Sequence (ignored otherwise).<br>0 Disabled – sequence manager will not follow a receive frame with a Tx Ack frame under any conditions; the sequence will terminate after the receive frame<br>1 Enabled – sequence manager will follow a receive frame with an automatic hardware-generated TX Ack frame, assuming other necessary conditions are met. |
| [2:0]<br>XCVSEQ | **Transceiver Sequence** — Writing a command selects one of six possible transceiver operations.<br>• Writing a new command while in the IDLE state initiates a new sequence.<br>• The sequence start can be optionally delayed through a timer-triggered mode (see TMRTRIGEN)<br>• Writing an IDLE command aborts any ongoing sequence.<br>• Writing of an IDLE command must always be performed after a sequence is complete, before a new sequence is programmed or the new command will be ignored.<br>• Any write to XCVSEQ other than XCVSEQ = IDLE during an ongoing sequence, shall be ignored.<br>See Table 6-19 for the available operations. |

Table 6-19. Transceiver Field XCVSEQ[2:0] Operations

| XCVSEQ[2:0] | Sequence Name | Description |
|---|---|---|
| 000 | IDLE (I) | Idle State (default) – should be set after completed sequence. Setting IDLE during a sequence aborts the sequence. |
| 001 | Receive (R) | Receive Sequence – may be conditionally followed by a TxAck |
| 010 | Transmit (T) | Transmit Sequence – a proceeding CCA can be optionally executed |
| 011 | CCA (C) | Clear Channel Assessment – standalone sequence |
| 100 | TR | Transmit / Receive Sequence – transmit sequence (conditional CCA) followed by a frame receive or if the RXACKRQRD bit is set, an RxAck will also be performed. |
| 101 | CCCA (C) | Continuos CCA – CCA operations are executed back-to-back in a continuos stream until an idle channel status is found or the sequence is aborted by an IDLE command. |
| 110 | Reserved | Not used |
| 111 | Reserved | Not Used |

### 6.13.4 Transceiver Control Register 2 (CNTRL2)

CNTRL2 is one of several registers that control basic operation of the transceiver. Included in this register are fields for:

- Master mask bit for enabling transceiver interrupt requests
- Timer Comparator 3 mode control
- PAN Coordinator enable
- CCA type selection
- Loading Event Timer counter (current time)
- Receiver promiscuous mode enable
- Event Timer TC2 Comparator mode control

CNTRL2 0x0045

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TRCV_MSK | TC3TMOUT | PANCORDNTR | CCATYPE | | 0 | PROMISCUOUS | TC2’EN |
| W | | | | | | TMRLOAD | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-13. Transceiver Control Register 2**

**Table 6-20. CNTRL2 Field Descriptions**

| Field | Description |
|---|---|
| TRCV_MSK | **Modem Interrupt Mask** — this bit is the general interrupt request mask for the transceiver. IRQs must be enabled via this bit for the other masks to have effect<br>0 Interrupt requests enabled<br>1 Interrupt requests disabled |
| TC3TMOUT | **Timer Comparator 3 Time-out** — this bit selects use of the Timer Comparator 3 when it is enabled<br>0 Event Timer Comparator 3 used as standard Event Timer.<br>1 Event Timer Comparator 3 is dedicated to RX time-out. |
| PAN-CORDNTR | **Pan Coordinator Enable** — this bit enables the node as a PAN coordinator<br>0 Device is not a Pan Coordinator<br>1 Device is a Pan Coordinator, packet filtering can accept MAC command or data frames with no destination address. |
| [1:0]<br>CCATYPE | **CCA Type** — this field selects one of three possible types of CCA algorithms. See Table 6-21 for details. |
| TMRLOAD | **Timer Load** — A low to high transition of this bit causes the value of field 'T1CMP[23:0]' to be loaded into the Event Timer Counter. This is a self-clearing bit, always reads zero. |

Table 6-20. CNTRL2 Field Descriptions (continued)

| Field | Description |
|---|---|
| PROMIS-CUOUS | **Promiscuous Mode Enable** — enable a “promiscuous” receiver mode; also prevents auto-TxAck on received packets<br>0 Normal operation<br>1 All filtering disabled except for length checking. CRC checking determined by CRCMSK bit; prevents auto-TxAck |
| TC2’EN | **TC2 Prime Enable** — This bit is used to select whether the TC2' or the TC2CMP comparator starts the sequence.<br>0 Use TC2 (24-bit compare)<br>1 Use TC2' (16-bit compare) |

Table 6-21. CCATYPE Select Options

| CCATYPE[1:0] | Type | Description |
|---|---|---|
| 00 | ED – energy detect | Use only with Sequence C as part of channel scan |
| 01 | CCA 802.15.4 Mode 1 | Measures any energy on the channel. Use with Sequences T, T/R, C, and CCA |
| 10 | CCA 802.15.4 Mode 2 | Measures only 802.15.4 modulated energy on the channel. Use with Sequences T, T/R, C, and CCA |
| 11 | Reserved | Not used |

## 6.13.5 Transceiver Control Register 3 (CNTRL3)

CNTRL3 is one of several registers that control basic operation of the transceiver. Included in this register are fields for:

- CRC disable for reception
- Masks for enabling interrupt requests
  - LO1 unlock
  - RX packet filtering
  - RX DMA “water mark”
  - CCA complete
  - RX complete
  - TX complete
  - Sequence complete

| CNTRL3 | | | | | 0x0046 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | CRC_MSK | LO1_UNLOCK_MSK | FILTERFAIL_MSK | RX_WMRK_MSK | CCAMSK | RXMSK | TXMSK | SEQMSK |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-14. Transceiver Control Register 3**

**Table 6-22. CNTRL3 Field Descriptions**

| Field | Description |
|---|---|
| CRC_MSK | **Cyclic Redundancy Check**<br>0 CRC result ignore by sequence manager.<br>1 Normal operation |
| LO1_UNLOCK_MSK | **Local Oscillator 1 Unlock Interrupt Mask Enable**<br>0 allows lo1_lock_irq to generate an interrupt pin.<br>1 lo1_lock_irq status bit is set, but an interrupt is not generated. The operation is aborted. |
| FLITERFAIL_MSK | **FLITERFAIL Interrupt Mask Enable**<br>0 Allows for RX Packet filtering to set the FILTERFAIL_IRQ flag<br>1 FILTERFAIL_IRQ not generated but filtering will take place. If filtering is not wanted enable Promiscuous mode. |
| RX_WMRK_MSK | **Receiver Water Mark Interrupt Mask Enable**<br>0 Allows for Received Count Water Mark interrupt to occur.<br>1 Received Count Water Mark interrupts inhibited |
| CCAMSK | **Clear Assessment Interrupt Mask**<br>0 Allows generation of CCA Interrupts and setting of CCAIRQ flag after CCA operation is completed.<br>1 Interrupt is masked, CCA is completed and CCA status is set accordingly after completion, but the interrupt is not generated. |
| RXMSK | **Receiver Interrupt Mask**<br>0 Allows generation of Receiver Interrupts and setting of RXIRQ flag after RX operation is completed.<br>1 Receiver Interrupt Mask, disables interrupt generation from the receiver operation |
| TXMSK | **Transmitter Interrupt Mask**<br>0 Allows generation of Transmitter Interrupts and setting of TXIRQ flag after TX operation is completed.<br>1 Transmitter Interrupt Mask, disables interrupt generation from the transmit operation |
| SEQMSK | **Sequence Interrupt Mask**<br>0 Allows generation of Receiver Interrupts and setting of SEQIRQ flag after Sequence is completed.<br>1 Sequence Interrupt Mask, disables interrupt generation from the sequence completion. |

## 6.13.6 Transceiver Control Register 4 (CNTRL4)

CNTRL4 is one of several registers that control basic operation of the transceiver. Included in this register are fields for:

- Mask for enabling Event Timer Comparators 1-4
- Enable bits for Event Timer Comparators 1-4

| | CONTROL4 | | | | 0x0047 | | | |
|---|---|---|---|---|---|---|---|---|
| 0x0047 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | TMR4MSK | TMR3MSK | TMR2MSK | TMR1MSK | TMR4CMP_EN | TMR3CMP_EN | TMR2CMP_EN | TMR1CMP_EN |
| Reset | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-15. Transceiver Control Register 4**

**Table 6-23. CNTRL4 Field Descriptions**

| Field | Description |
|---|---|
| TMR4MSK | **Timer Comparator 4 Interrupt Mask**<br>0 allows interrupt when comparator matches event timer count.<br>1 Interrupt generation is disable, but an TMR4IRQ flag it set. |
| TMR3MSK | **Timer Comparator 3 Interrupt Mask**<br>0 allows interrupt when comparator matches event timer count.<br>1 Interrupt generation is disable, but an TMR3IRQ flag it set. |
| TMR2MSK | **Timer Comparator 2 Interrupt Mask**<br>0 allows interrupt when comparator matches event timer count.<br>1 Interrupt generation is disable, but an TMR2IRQ flag it set. |
| TMR1MSK | **Timer Comparator 1Interrupt Mask**<br>0 allows interrupt when comparator matches event timer count.<br>1 Interrupt generation is disable, but an TMR1IRQ flag it set. |
| TMR4CMP_EN | **Timer Comparator 4 Enable**<br>0 Interrupt is disabled and TMR4IRQ status bit flag is cleared.<br>1 Standard operation |
| TMR3CMP_EN | **Timer Comparator 3 Enable**<br>0 Interrupt is disabled and TMR3IRQ status bit flag is cleared.<br>1 Standard operation |
| TMR2CMP_EN | **Timer Comparator 2 Enable**<br>0 Interrupt is disabled and TMR2IRQ status bit flag is cleared.<br>1 Standard operation |
| TMR1CMP_EN | **Timer Comparator 1 Enable**<br>0 Interrupt is disabled and TMR1IRQ status bit flag is cleared.<br>1 Standard operation |

### 6.13.7 Transceiver Source Address Control Register (SRC_CNTRL)

The SRC_CNTRL register enables and controls the the Source Address Matching function in the sequence manager.

| SRC_CNTRL | | | | | 0x0048 | | | |
|---|---|---|---|---|---|---|---|---|
| 0x0048 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | INDEX | | | | ACK_FRM_<br>PND | SRCADDR_<br>EN | INDEX_EN | INDEX_DISA<br>BLE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-16. Transceiver Source Address Control Register**

**Table 6-24. SRC_CNTRL Field Descriptions**

| Field | Description |
|---|---|
| INDEX | **Index —** The index field selects one of 12 table entries to be updated. The match state machine will return the index of the element whose value matches the incoming frame. If there was no match, the Index will be b1111 (0xF). |
| ACK_FRM_PND | **Frame Pending Acknowledge —** If SRCADDR_EN enable field is a 0, then ACK_FRM_PND gets copied into the TXACK frame. |
| SRCADDR_EN | **Source Address Matching Enable —** This bit enables the Source Address Matching function. If this bit is equal to zero, the pend bin in the TxAck frame will be taken from ACK_FRM_PND.<br>Note: AUTOACK enable bit must be set to send any auto-TxAck frame |
| INDEX_EN | **Index Enable —** Setting this bit to a '1' enables the selected index.<br>Write only (read operations return '0') |
| INDEX_DISABLE | **Index Disable—** Setting this bit to a '1' the disables the selected index.<br>Write only (read operations return '0') |

## 6.13.8 Transceiver Source Address Sum Data Registers (SRC_ADDRS_SUM_DATA0 and SRC_ADDRS_SUM_DATA1)

The Source Address Sum Data [15:0] Register holds the 16-bit sum data compare value as calculated from the enabled IEEE 802.15.4 address mode:

- SRC_ADDRS_SUM_DATA0 – the most significant 8 bits (SRC_ADDRS_SUM_DATA[15:8])
- SRC_ADDRS_SUM_DATA1 – the least significant 8 bits (SRC_ADDRS_SUM_DATA[7:0])

| | SRC_ADDRS_SUM_DATA0 | | | | 0x0049 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | SRC_ADDRS_SUM_DATA[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | SRC_ADDRS_SUM_DATA1 | | | | 0x004A | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | SRC_ADDRS_SUM_DATA [7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-17. Transceiver Source Address Sum Data Registers**

**Table 6-25. SRC_ADDRS_SUM_DATA Field Description**

| Field | Description |
|---|---|
| SRC_<br>ADDRS_<br>SUM_DATA | **Source Address Sum Data—** The Source Address Sum register is composed of 2 8-bit registers. These registers are loaded by the MCU software with the sum of address elements. The software must compute this 16-bit data field in accordance with the following:<br><br>Short Address:<br>SourcePanID[15:0] (if intra-pan substitutes Destination Pan ID)<br>+ SourceAddr[15:0]<br>Calculated Value to be placed in Table and Compare Against<br><br>Long Address sum data is calculated:<br>SrcPanID[15:0]<br>SourceAddr[15: 0]<br>SourceAddr[31:16]<br>SourceAddr[47:32]<br>+ SourceAddr[63:48]<br>Calculated Value to be place in Table and Compare Against |

## 6.13.9 Receiver Byte Count Watermark Threshold (RX_WTR_MARK)

The RX_WTR_MARK register holds an 8-bit field that sets the level of received packet bytes needed to enable a receive interrupt request.

- Maximum usable count is 125 bytes.
- A count of 126 or larger (default 0xFF) will never generate an interrupt request

| | RX_WTR_MARK | | | | 0x004B | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | RX_WTR_MARK[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-18. Receiver Byte Count Watermark Threshold**

**Table 6-26. RX_WTR_MARK Field Description**

| Field | Description |
|---|---|
| [7:0]<br>RX_WTR_MARK | **Receiver Byte Count Interrupt Watermark**— Receive byte count (octets) needed to trigger a RX interrupt. The count is started at the beginning after the SFD field. A setting of 0 generates an interrupt at end of the SFD field. |

## 6.13.10 Receiver Byte Count (RX_BYTE_COUNT)

The read-only RX_BYTE_COUNT register reports the present number of received bytes in a packet.

| | RX_BYTE_COUNT | | | | 0x004C | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | RX_BYTE_COUNT[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-19. Receiver Byte Count**

**Table 6-27. RX_BYTE_COUNT Field Description**

| Field | Description |
|---|---|
| [7:0]<br>RX_BYTE_COUNT | **Receiver Byte Count** — Received byte count (octets) after the SFD field.<br>Read only. |

## 6.13.11 Status Register 1 (STATUS1)

STATUS1 is one of two status registers for the transceiver. Included in this register are fields for:

- CRC valid status
- CCA result
- Received packet Source Address test result
- Poll Indication status
- Event timer Comparator 1-4 compare status

| | STATUS1 | | | | 0x004D | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | CRCVALID | CCA | SRCADDR | PI | TMR4IRQ | TMR3IRQ | TMR2IRQ | TMR1IRQ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-20. Status Register 1**

**Table 6-28. STATUS1 Field Descriptions**

| Field | Description |
|---|---|
| CRCVALID | **Code Redundancy Check Valid** — This flag indicates the compare result between the FCS field, in the just received frame, and the internally calculated CRC value. This flag is clear at receiver warm up.<br>0 RX FCS <> CRC (incorrect)<br>1 RX FCS = CRC (correct) |
| CCA | **Clear Channel Assessment Result** — This flag works in conjunction with the CCA type selected<br><br>CCA Type 0:<br>Not used.<br><br>CCA Type 1:<br>This flag is set as the result of comparison between the Energy measured (ED) and the CCA threshold register (CCA_THRESHOLD).<br>If ED <= CCA_THRESHOLD, channel busy detected, CCA = 1<br>If ED > CCA_THRESHOLD, channel idle detected, CCA = 0<br><br>CCA Type 2:<br>This flag is set as the result of comparison between number of correlation peaks (NUM_CORR_PEAKS) that exceeded the threshold value and the CCA Mode 2 threshold register (CCA2_MIN_NUM_CORR_TH)<br>If NUM_CORR_PEAKS ? CCA2_MIN_NUM_CORR_TH, channel busy detected, CCA = 1<br>If NUM_CORR_PEAKS < CCA2_MIN_NUM_CORR_TH, channel idle detected, CCA = 0<br><br>CCA Type 3:<br>Reserved. |

Table 6-28. STATUS1 Field Descriptions (continued)

| Field | Description |
|---|---|
| SRCADDR | **Source Address Checksum —** Part of the source address feature. Indicates that:<br>1 – The incoming packet was a data request;<br>2 – Source address matching is enabled, SRCADDR_EN = 1<br>3 – Source address check sum computed by packet processor matched at least one entry in the source address table.<br>This is a read only bit. |
| PI | **Poll Indication** —<br>1 the received packet was a data request, regardless of whether a source address table match occurred, or whether source address matching is enabled (See SRCADDR_EN bit)<br>0 the received packet was not a data request. |
| TMR4IRQ | **Timer Comparator 4 Interrupt**<br>1 Indicates comparator value matched event timer counter. This is write 1 to clear bit. |
| TMR3IRQ | **Timer Comparator 3 Interrupt**<br>1 Indicates comparator value matched event timer counter. This is write 1 to clear bit. |
| TMR2IRQ | **Timer Comparator 2 Interrupt**<br>1 Indicates comparator value matched event timer counter. This is write 1 to clear bit.<br>This flag is shared between TC2’ and TC2. |
| TMR1IRQ | **Timer Comparator 1 Interrupt**<br>1 Indicates comparator value matched event timer counter. This is write 1 to clear bit. |

### 6.13.12 Status Register 2 (STATUS2)

STATUS2 is the second of two status registers for the transceiver. Included in this register are fields for:

- Frame pending status
- LO1 unlock status
- Receiver Packet Filter Fail status
- RX watermark status
- CCA completion status
- RX completion status
- TX completion status
- Sequence completion status

| | STATUS2 | | | | 0x004E | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W2C | RX_FRM_PEND | LO1_UNLOCK_IRQ | FILTERFAIL_IRQ | RXWTRMRKIRQ | CCAIRQ | RXIRQ | TXIRQ | SEQIRQ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-21. Status Register 2**

**Table 6-29. STATUS2 Field Descriptions**

| Field | Description |
|---|---|
| RX_FRM_PEND | **Receiver Frame Pending —** Status of the frame pending bit of the frame control field for the just received packet |
| LO1_UNLOCK_IRQ | **Local Oscillator Unlock** — A 1 indicates an unlock condition in the PLL.This is write a 1 to clear bit |
| FILTERFAIL_IRQ | **Receiver Packet Filter Fail** — A 1 indicates that the just received packet has been rejected due to elements within the packet. This is write a 1 to clear bit. |
| RXWTRMRKIRQ | **Receiver Byte Count Water Mark** — A 1 indicates that the number of bytes specified in the RX_WTR_MARK register has been reached.This is write a 1 to clear bit. |
| CCAIRQ | Clear Channel Assessment flag. A 1 indicates completion of CCA operation. This is write a 1 to clear bit. |
| RXIRQ | Receiver flag. A '1' indicates the completion of a received operation. This is write a 1 to clear bit. |
| TXIRQ | Transmitter flag. A '1' indicates the completion of a transmit operation. This is write a 1 to clear bit. |
| SEQIRQ | Sequence flag. A '1' indicates the completion of a sequence operation. This is write a 1 to clear bit. |

### 6.13.13 Clear Channel Assessment Final Value (CCAFNL)

The read-only CCAFNL register reports either of two values:

- The final CCA value as averaged over the selected CCA type-appropriate sample time (after a CCA cycle)
- The LQI value for a received frame.

**CCAFNL** **0x0050**

|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CCAFNL[7:0] |  |  |  |  |  |  |  |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-22. Clear Channel Assessment Final Value**

**Table 6-30. CCAFNL Field Description**

| Field | Description |
|---|---|
| [7:0]<br>CCAFNL | **Clear Channel Assessment Average** — Average result of the selected CCA algorithm selected through the CCATYPE field.<br>00 CCA calculated over 4 symbols – 4*16uS = 64uS<br>01 CCA calculated over 8 symbols – 8*16uS = 128uS<br>10 CCA calculated over 8 symbols – 8*16uS = 128uS<br>11 Reserved<br>At the end of any received packet, this field contains the Link Quality Indicator (LQI) value; which is the measurement of the received energy that occurs during the received frame, immediately the preamble, and over 4 symbol period (64uS) |

### 6.13.14 Event Timer Registers (EVENT_TMR0, EVENT_TMR1, and EVENT_TMR2)

The read-only Event_Timer [23:0] Registers holds the current value of the event timer counter. The 24-bit counter value is composed of three registers:

- EVENT_TMR0 – the most significant 8 bits (EVENT_TMR[23:16])
- EVENT_TMR1 – the next most significant 8 bits (EVENT_TMR[15:8])
- EVENT_TMR2 – the least significant 8 bits (EVENT_TMR[7:0])

**EVENT_TMR0** **0x0051**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EVENT_TMR[23:16] | | | | | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENT_TMR1** **0x0052**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EVENT_TMR[15:8] | | | | | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EVENT_TMR2** **0x0053**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | EVENT_TMR[7:0] | | | | | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-23. Event Timer Registers**

**Table 6-31. EVENT_TMR Field Description**

| Field | Description |
|---|---|
| [23:0]<br>EVENT_TMR | **Event Timer —** Holds the current value of the event timer counter.<br>• The hardware latches on a read of least significant byte<br>• The latched value is released on the read of the most significant byte |

### 6.13.15 Time Stamp Registers (TIMESTAMP0, TIMESTAMP1, and TIMESTAMP2)

The read-only Timestamp [23:0] Registers hold the latched event timer counter value (current time) corresponding to the beginning of the last received RX packet. The value is latched at the SFD detect. The 24-bit timestamp value is composed of three registers:

- TIMESTAMP0 — the most significant 8 bits (TIMESTAMP[23:16])
- TIMESTAMP1 — the next most significant 8 bits (TIMESTAMP[15:8])
- TIMESTAMP2 — the least significant 8 bits (TIMESTAMP[7:0])

**TIMESTAMP0** **0x0054**

| 0x0054 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP[23:16] | | | | | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TIMESTAMP1** **0x0055**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP[15:8] | | | | | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TIMESTAMP2** **0x0056**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP[7:0] | | | | | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-24. Time Stamp Registers**

**Table 6-32. TIMESTAMP Field Description**

| Field | Description |
|---|---|
| [23:0]<br>TIMESTAMP | **Time Stamp** — Holds the latched value of the Event Timer current time corresponding to the beginning of the just received RX packet, at SFD detect. Software should read only at idle state of sequence manager. |

### 6.13.16 Event Timer Comparator 3 Registers (T3CMP0, T3CMP1, and T3CMP2)

The T3CMP[23:0] Registers hold the compare value used by Event Timer Comparator 3. The 24-bit compare value is composed of three registers:

- T3CMP0 – the most significant 8 bits (T3CMP[23:16])
- T3CMP1 – the next most significant 8 bits (T3CMP[15:8])
- T3CMP2 – the least significant 8 bits (T3CMP[7:0])

| T3CMP0 | | | | | 0x0057 | | | |
|---|---|---|---|---|---|---|---|---|
| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R<br>W | T3CMP[23:16] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| T3CMP1 | | | | | 0x0058 | | | |
|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| R<br>W | T3CMP[15:8] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| T3CMP2 | | | | | 0x0059 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | T3CMP[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-25. Event Timer Comparator 3 Registers**

**Table 6-33. T3CMP Field Description**

| Field | Description |
|---|---|
| [23:0]<br>T3CMP | **Event Timer Comparator 3** — Event Timer 3 absolute time compare value. In addition to standard use as an event timer, TC3 can also be used as a time-out timer for the receiver. This is controlled through bit TC3TMOUT in Control Reg 2. Software must clear the TMR3CMP_EN bit prior to updating this register. |

### 6.13.17 Time Comparator 2 Prime Registers (TC2_PRIME0 and TC2_PRIME1)

The TC2_PRIME[15:0] Registers hold the 16-bit compare value used by Event Timer Comparator 2 for a special sequence. The 16-bit compare value is composed of two registers:

- TC2_PRIME0 – the most significant 8 bits (TC2_PRIME[15:8])
- TC2_PRIME1 – the least significant 8 bits (TC2_PRIME[7:0])

**TC2_PRIME0** **0x005A**

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TC2_PRIME[15:8] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**TC2_PRIME1** **0x005B**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TC2_PRIME[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-26. Time Comparator 2 Prime Registers**

**Table 6-34. TC2_PRIME Field Description**

| Field | Description |
|---|---|
| [15:0]<br>TC2_PRIME | **Event Timer Comparator 2 Prime** — Event Timer 2 absolute time compare value (lower 16 bits only). Matches the lower 16 bits of the Event Timer counter against the programmed value.<br>• Works in conjunction with TMRTRIGEN to initiate a particular transceiver sequence (XCVSEQ) at the matched point.<br>• Software must clear the TMR2CMP_EN bit prior to updating this register. |

### 6.13.18 MAC Short Address Registers (MACSHORTADDRS0 and MACSHORTADDRS1)

The MACSHORTADDRS[15:0] Registers hold the 16-bit MAC Short Address value used by the device when Short Address Mode is used. The 16-bit address value is composed of two registers:

- MACSHORTADDRS0 — the most significant 8 bits (MACSHORTADDRS[15:8])
- MACSHORTADDRS1 — the least significant 8 bits (MACSHORTADDRS[7:0])

**MACSHORTADDRS0** **0x005C**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MACSHORTADDRS[15:8] | | | | | | | |
| W | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**MACSHORTADDRS1** **0x005D**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | MACSHORTADDRS[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-27. MAC Short Address Registers**

**Table 6-35. MACSHORTADDRS Field Description**

| Field | Description |
|---|---|
| [15:0]<br>MACSHORTADDRS | **Mac Short Address** — Value used as device MAC Short Address when that mode is enabled |

### 6.13.19 MAC Pan ID Registers (MACPANID0 and MACPANID1)

The MACPANID[15:0] Registers hold the 16-bit MAC PAN ID value used by the device. The 16-bit PAN ID value is composed of two registers:

- MACPANID0 – the most significant 8 bits (MACPANID[15:8])
- MACPANID1 – the least significant 8 bits (MACPANID[7:0])

| | MACPANIDS0 | | | | 0x005E | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | MACPANID[15:8] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | MACPANID1 | | | | 0x005F | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | MACPANID[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-28. MAC Pan ID Registers**

**Table 6-36. MACPANID Field Description**

| Field | Description |
|---|---|
| [15:0]<br>MACPANID | **Mac Short Address** — Value used as device PAN ID |

## 6.13.20 Event Timer Comparator 1 Registers (T1CMP0, T1CMP1, and T1CMP2)

The T1CMP[23:0] Registers hold the compare value used by Event Timer Comparator 1. The 24-bit compare value is composed of three registers:

- T1CMP0 – the most significant 8 bits (T1CMP[23:16])
- T1CMP1 – the next most significant 8 bits (T1CMP[15:8])
- T1CMP2 – the least significant 8 bits (T1CMP[7:0])

**T1CMP0** **0x1840**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | T1CMP[23:16] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**T1CMP1** **0x1841**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | T1CMP[15:8] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**T1CMP2** **0x1842**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | T1CMP[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

░ = Unimplemented or Reserved

**Figure 6-29. Event Timer Comparator 1 Registers**

**Table 6-37. T1CMP Field Description**

| Field | Description |
|---|---|
| [23:0]<br>T1CMP | **Event Timer Comparator 1** — Event Timer 1 absolute time compare value. Software must clear the TMR1CMP_EN bit prior to updating this register. |

### 6.13.21 Event Timer Comparator 2 Registers (T2CMP0, T2CMP1, and T2CMP2)

The T2CMP[23:0] Registers hold the compare value used by Event Timer Comparator 2. The 24-bit compare value is composed of three registers:

- T2CMP0 – the most significant 8 bits (T2CMP[23:16])
- T2CMP1 – the next most significant 8 bits (T2CMP[15:8])
- T2CMP2 – the least significant 8 bits (T2CMP[7:0])

| | T2CMP0 | | | | 0x1843 | | | |
|---|---|---|---|---|---|---|---|---|
| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R<br>W | T2CMP[23:16] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | T2CMP1 | | | | 0x1844 | | | |
|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| R<br>W | T2CMP[15:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | T2CMP2 | | | | 0x1845 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | T2CMP[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-30. Event Timer Comparator 2 Registers**

**Table 6-38. T2CMP Field Description**

| Field | Description |
|---|---|
| [23:0]<br>T2CMP | **Event Timer Comparator 2** — Event Timer 2 absolute time compare value. Software must clear the TMR2CMP_EN bit prior to updating this register. |

## 6.13.22 Event Timer Comparator 4 Registers (T4CMP0, T4CMP1, and T4CMP2)

The T4CMP[23:0] Registers hold the compare value used by Event Timer Comparator 4. The 24-bit compare value is composed of three registers:

- T4CMP0 – the most significant 8 bits (T4CMP[23:16])
- T4CMP1 – the next most significant 8 bits (T4CMP[15:8])
- T4CMP2 – the least significant 8 bits (T4CMP[7:0])

**T4CMP0** **0x1846**

| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | T4CMP[23:16] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**T4CMP1** **0x1847**

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | T4CMP[15:8] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**T4CMP2** **0x1848**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | T4CMP[7:0] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-31. Event Timer Comparator 4 Registers**

**Table 6-39. T4CMP Field Description**

| Field | Description |
|---|---|
| [23:0]<br>T4CMP | **Event Timer Comparator 4** — Event Timer 4 absolute time compare value. Software must clear the TMR4CMP_EN bit prior to updating this register. |

### 6.13.23 LO1 FracN Numerator Value Registers (LO1_FRAC0 and LO1_FRAC1)

The LO1_FRAC[15:0] Registers hold the 16-bit LO1 FracN Numerator value used by the VCO synthesizer. The LO1_FRAC value is composed of two registers:

- LO1_FRAC0 – the most significant 8 bits (LO1_FRAC[15:8])
- LO1_FRAC1 – the least significant 8 bits (LO1_FRAC[7:0])

To calculate the FracN Numerator, see Section 6.6, “VCO Frequency Synthesizer (Channel Frequency)”.

| | **LO1_FRAC0** | | | | **0x1849** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | LO1_FRAC[15:8] | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

| | **LO1_FRAC1** | | | | **0x184A** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | LO1_FRAC[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-32. LO1 FracN Numerator Value Registers**

**Table 6-40. LO1_FRAC Field Description**

| Field | Description |
|---|---|
| LO1_FRAC | **LO1 FracN Divide Value** — used by the VCO synthesizer. LO1 FREQUENCY = 32e6 * (L01_INT + 1 + (LO1_FRAC / 65536)). Set frequency on channel |

## 6.13.24 LO1 Integer Register (LO1_INT)

The LO1_INT[7:0] Register hold the 8-bit LO1 Integer divide value used by the VCO synthesizer.

To calculate the LO1 Integer, see Section 6.6, “VCO Frequency Synthesizer (Channel Frequency)”.

| LO1_INT | | | | | 0x184B | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | LO1_INT | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-33. LO1 Integer Register**

**Table 6-41. LO1_INT Field Description**

| Field | Description |
|---|---|
| [7:0]<br>LO1IDIV | **LO1 Integer Divide Value** — used by the VCO synthesizer. LO1 FREQUENCY = 32e6 * (LO1_INT + 1 + (LO1_FRAC / 65536)). Set frequency on channel |

## 6.13.25 Power Amplifier (PA) Power Control Register (PA_PWR_CNTL)

The PA_PWR_CNTL Register is one of four registers that sets the RF PA power out during a transmission. The additional three registers are accessed through the Transceiver Indirect Access Registers (see Section 6.13.33, “Transceiver Indirect Access Registers (INDEX and DATA)”). The indirect access addresses are 0x410, 0x411, and 0x4020).

### NOTE

Use Freescale provided software utilities for setting TX RF output power. The utilities provide 15 power steps where the typical output power is shown in Table 6-42.

**Table 6-42.**

| Power Step (Dec) | Typical Output Power (dBm) |
|---|---|
| 0 | -30 |
| 1 | -29 |
| 2 | -25 |
| 3 | -23 |
| 4 | -21 |
| 5 | -19 |
| 6 | -16 |

Table 6-42.

| Power Step (Dec) | Typical Output Power (dBm) |
|---|---|
| 7 | -14 |
| 8 | -13 |
| 9 | -9 |
| 10 | -7 |
| 11 | -4 |
| 12 | -2 |
| 13 | 0 |
| 14 | +1 |

| PA_PWR_CNTL | | | | | 0x184C | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | Reserved | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-34. Power Amplifier (PA) Power Control Register**

**Table 6-43. PA_PWR_CNTL Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>Reserved | Reserved — use default value |

## 6.13.26 MAC Long Address Registers (MACLONGADDRS0 – MACLONGADDRS7)

The MAC Long Address Registers supply the 64-bit IEEE 802.15.4 Long Device Address MACLONGADDRS [63:0] :

- Register MACLONGADDRS0 – field MACLONGADDRS [63:56] most significant byte
- Register MACLONGADDRS1 – field MACLONGADDRS [55:48]
- Register MACLONGADDRS2 – field MACLONGADDRS [47:40]
- Register MACLONGADDRS3 – field MACLONGADDRS [39:32]
- Register MACLONGADDRS4 – field MACLONGADDRS [31:24]
- Register MACLONGADDRS5 – field MACLONGADDRS [23:16]

- Register MACLONGADDRS6 – field MACLONGADDRS [15:8]
- Register MACLONGADDRS7 – field MACLONGADDRS [7:0] least significant byte

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x184D | MACLONGADDRS[63:56] | | | | | | | |
| 0x184E | MACLONGADDRS[55:48] | | | | | | | |
| 0x184F | MACLONGADDRS[47:40] | | | | | | | |
| 0x1850 | MACLONGADDRS[39:32] | | | | | | | |
| 0x1851 | MACLONGADDRS[31:24] | | | | | | | |
| 0x1852 | MACLONGADDRS[23:16] | | | | | | | |
| 0x1853 | MACSHORTADDRS[15:8] | | | | | | | |
| 0x1854 | MACLONGADDRS[7:0] | | | | | | | |
| R/W | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-35. MAC Long Address Registers**

**Table 6-44. MACLONGADDRS Field Description**

| Field | Description |
|---|---|
| MACLONG-<br>ADDRS<br>[63:0] | **Mac Long Address** — Device 64-bit IEEE 802.15.4 Long Address. |

## 6.13.27 Max Frame Length Register (MAXFRAMELENGTH)

The MAXFRAMELENGTH Register provides the programmed maximum byte count.

| | MAXFRAMELENGTH | | | | 0x1855 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | MAXFRAMELENGTH[7:0] | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 6-36. Max Frame Length Register**

**Table 6-45. MAXFRAMELENGTH Field Description**

| Field | Description |
|---|---|
| maxFrameLength<br>[7:0] | **Maximum Frame Length Accepted** — Used by frame filtering to limit maximum byte count of frames to be received. |

### 6.13.28 Receive Frame Filter (RX_FRAME_FILTER)

Within the receive frame 2-byte Frame Control field, there is a 3-bit Frame Type subfield. Frame filtering allows rejection of the incoming receive frame based on the value of this subfield. Frame type as defined by the Frame Type subfield are listed in

**Table 6-46. IEEE 802.15.4 Frame Type Subfield**

| Frame Type [2:0] | Frame Type Designation |
|---|---|
| 000 | Beacon |
| 001 | Data |
| 010 | Acknowledgement |
| 011 | MAC command |
| 100–111 | Reserved |

Control bits in the RX_FRAME_FILTER Register set the accept/reject option

The receive will only complete if the following conditions are met:

1. Destination address:
   a) If PAN ID is included other than the broadcast, it must match the macPANID else the PAN ID must be the broadcast value (0xFFFF)
   b) If a short destination address is include in the frame, it must match the macShortAddress field else the broadcast address (0xFFFF)
   c) If a long destination address is include in the frame, it must match the aExtendedAddress.field
   d) If only source addressing fields are included in a Data or MAC command frame, the frame shall be accepted only if the device is a PAN coordinator (PANCOORD = 1) and the source PAN identifier matches macPANId.
2. Illegal Frame types are rejected, unless the NS_FT field is set.
3. Beacon frames are accepted under the following conditions
   a) Beacon frames enable
   b) The source PAN ID matches macPANID or it equals 0xFFFF
   c) Length >= 9
   d) The destination address mode is 0 (no destination address)
   e) The source address mode is 2 or 3 (i.e. a source address is included)
4. Data frames are accepted under the following conditions:
   a) Data frames are enabled
   b) Length  >= 9
   c) If the device is a PAN coordinator (PANCOORD bit) and the source PAN identifier matches macPANId. This means that if the destination address is not present, the frame will be accepted only if the PANCOORD = 1 and the PANID in the source address field is equal to the receiving device macPANID. If the device is NOT a PAN coordinator, the destination address mode must be 2 or 3, and destination address-matching rules apply.

5. Acknowledgement Frames are accepted under the following conditions:
   a) Acknowledge Frames are enabled
   b) Length == 5
   c) Sequence number has to match.
6. Command Frames are accepted under the following conditions:
   a) Command Frames are enabled
   b) Length >= 9
   c) Source or  Destination Address are included, else it must be a coordinator (PANCOORD = 1) in which case the PANID must equal macPANID. If the device is NOT a PAN coordinator, the destination address mode must be 2 or 3, and destination address-matching rules apply.

Packet Rejection:

If a frame is rejected, the transceiver will start searching for a new frame after the rejected frame has been received.

**RX_FRAME_FILTER** **0x1856**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ▨ | ▨ | ▨ | NS_FT | CMD_FT | ACK_FT | DATA_FT | BEACON_FT |
| W | ▨ | ▨ | ▨ | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-37. Receive Frame Filter Register**

**Table 6-47. RX_FRAME_FILTER Field Description**

| Field | Description |
|---|---|
| NS_FT | **Not Specified FT** — Defines whether non-specified frames (4,5,6, and 7)  are rejected or not<br>0 Reject<br>1 Accept |
| CMD_FT | **MAC Command Frame Type** — Defines whether MAC command frames are accepted or not. MAC command frames have frame type = 011.<br>0 Reject<br>1 Accept |
| ACK_FT | **Acknowledge Frame Type** — Defines whether Ack frames are accepted or not. MAC acknowledge frames have frame type = 010.<br>0 Reject<br>1 Accept |

Table 6-47. RX_FRAME_FILTER Field Description (continued)

| Field | Description |
|---|---|
| DATA_FT | **Data Frame Type** — Defines whether Data frames are accepted or not. MAC data frames have frame type = 001.<br>0 Reject<br>1 Accept |
| BEACON_FT | **Beacon Frame Type** — Defines whether Beacon frames are accepted or not. MAC beacon frames have frame type = 000.<br>0 Reject<br>1 Accept |

## 6.13.29 MAC Frame Version Compatibility and Timer Prescale Register (FMR_REV_TMR)

The FMR_REV_TMR register provides the Event Timer prescaler control field as well as the Frame Version select which determines which version of IEEE 802.15.4 Standard frames that filtering will allow.

**FMR_REV_TMR** **0x1857**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TMR_PRESCALE2 | TMR_PRESCALE1 | TMR_PRESCALE0 | 0 | 0 | 0 | FRM_VER | |
| W | | | | ▨ | ▨ | ▨ | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

Figure 6-38. MAC Frame Version Compatibility and Timer Prescale Register

Table 6-48. FMR_REV_TMR Field Descriptions

| Field | Description |
|---|---|
| [2:0]<br>TMR_PRESCALE | **Timer Prescaler Select** — sets Event Timer pre scale ratio. See Section Table 6-7., "Event Timer Prescaler Settings" |
| FRM_VER<br>[1:0] | **802.15.4 Frame Version Selection**<br>00 All Frame Versions accepted (0,1,2 and 3)<br>01 Only accept FrameVersion 0 packets (2003 compliant)<br>10 Only accept FrameVersion 1 packets (2006 compliant)<br>11 Accept FrameVersion 0 and 1 packets, reject all others<br><br>Note: Frames received with FrameVersion 2 or 3 will be treated identically to FrameVersion 1, with respect to parsing of the Auxiliary Security Header. Other than this Header, all 4 frame versions will be treated identically |

## 6.13.30 CCA Energy Threshold (CCA_THRESHOLD)

The CCA_THRESHOLD Register provides the threshold value for comparison during a CCA cycle. See Section 6.5.1.3, “Reported Energy Values”

| CCA_THRESHOLD | | | | | 0x1858 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | CCA_THRESHOLD[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-39. CCA Energy Threshold**

**Table 6-49. CCA_THRESHOLD Field Description**

| Field | Description |
|---|---|
| [7:0]<br>CCA_THRESHOLD | **CCA Energy Threshold Detection**— Threshold value for the Clear Channel Assessment in dB-linear format |

## 6.13.31 CCA Power Compensation (CCA_OFFSET_CMP)

The CCA_OFFSET_CMP Register provides the offset value used to adjust/compensate the CCA and LQI calculation. See Section 6.5.1.3, “Reported Energy Values”

| CCA_OFFSET_CMP | | | | | 0x1859 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | CCA_OFFSET_CMP[7:0] | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

▨ = Unimplemented or Reserved

**Figure 6-40. CCA Power Compensation**

**Table 6-50. CCA_OFFSET_CMP Field description**

| Field | Description |
|---|---|
| [7:0]<br>CCA_OFFSET_CMP | **CCA Power Compensation** — Offset added to the calculated RSSI. |

### 6.13.32 Finite State Machine Register (FSM)

The FSM Register reports the state of the Sequence Manager Finite State Machine and can be used to monitor a sequence.

| | FSM | | | | 0x185A | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | FSM[5:0] | | | | | |
| W | ▨ | ▨ | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-41. Finite State Machine Register**

**Table 6-51. FSM Field Description**

| Field | Description |
|---|---|
| [5:0]<br>FSM | **Finite State Machine** — This register reflects the sequence manager internal state |

**Table 6-52. FSM State Codes**

| FSM[5:0] | State |
|---|---|
| 00 | IDLE |
| | |
| 21 | RX WARMUP |
| 22 | RX WARMUP |
| 23 | RX WARMUP |
| 24 | RX WARMUP |
| 25 | RX WARMUP |
| 26 | RX WARMUP |
| 27 | RX WARMUP |
| 28 | RX WARMUP |
| 29 | RX WARMUP |
| 2A | RX WARMUP |
| 2B | RX |
| 2C | RX |
| 2D | RX-TO-TX TRANSITION |
| | |
| 30 | CCA |

Table 6-52. FSM State Codes

| FSM[5:0] | State |
|---|---|
| 31 | CCA |
| 32 | WAIT BETWEEN 1ST & 2ND CCA |
| 33 | CCA |
| 34 | RX-TO-TX TRANSITION |
| | |
| 01 | TX WARMUP |
| 02 | TX WARMUP |
| 03 | TX WARMUP |
| 04 | TX WARMUP |
| 05 | TX WARMUP |
| 06 | TX WARMUP |
| 07 | TX WARMUP |
| 08 | TX WARMUP |
| 09 | TX WARMUP |
| 10 | TX |
| 14 | TX |
| 18 | CCA-TO-TX TRANSITION |
| 1C | TX WARMDOWN |
| 1D | TX WARMDOWN |
| 1E | TX WARMDOWN |
| 1F | TX-TO-RX TRANSITION |
| | |
| 3F | TRANSPARENT MODE |

### 6.13.33 Transceiver Indirect Access Registers (INDEX and DATA)

The transceiver Indirect Access Registers are normally reserved and are used to access a block of transceiver test radio functions. The only times this interface is used for application purposes are:

- Pseudo-random number generation
- Setting transmit power
- Controlling the analog voltage regulator for low power operation

**NOTE**

> Refer to Section 3.10, “Transceiver Indirect Access Register Application Functions” for information on using these functions.

The interface is composed of two registers:

- INDEX – writing this register provides an address to an internal register block that is used for read or write of the selected register
- DATA – writing data to this register causes a write of the same data to the internal register that is addressed by the INDEX register. A read access of the DATA register returns the contents of the internal register that is addressed by the INDEX register.

| | **INDEX** | | | | **0x185B** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | INDEX[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | **DATA** | | | | **0x185C** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | DATA[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 6-42. Transceiver Indirect Access Registers**

**Table 6-53. Indirect Access Register Field Descriptions**

| Field | Description |
|---|---|
| INDEX[7:0] | **Register Index** — provides a 7-bit register address index (Bit[6:0]) and an auto-increment control Bit 7. If Bit 7 is set, each access to the DATA register increments the index in the Bit [6:0] field. |
| DATA[7:0] | **Data** — writing this register causes a write of the same data to the indirect register addressed by the INDEX register. A read of this register accesses the data from the indirect register addressed by the INDEX register. |

# Chapter 7
# Advanced Security Module (ASM)

## 7.1 Introduction

The Advanced Security Module (ASM) is a hardware engine that encrypts using the 128-bit version of the Advanced Encryption Standard (AES). It can perform Counter (CTR), Cipher Block Chaining (CBC), and plain AES mode encryption. The combination of CTR and CBC modes of encryption is known as Counter with CBC-MAC (CCM) mode encryption. CCM is a generic authenticate and encrypt block cipher mode. CCM is only defined for use with 128-bit block ciphers, such as AES.

**NOTE**

- The encryption structures are briefly described in the modes below, however, users are directed to external resources for an in-depth understanding of the AES standard
- An in-depth definition of CCM mode encryption is documented in the NIST publication SP800-38C.
- The IEEE 802.15.4 Standard and the ZigBee specification deal only with encryption, not decryption. The ASM does not support all modes of decryption. The user is directed to the referenced standards and specifications for more information.
- The ASM block supports simple AES Mode encryption, but does NOT support simple AES decryption.

### 7.1.1 Features

The ASM has the following features:

- CTR encryption in 11 bus clock cycles.
- CBC encryption in 11 bus clock cycles
- AES encryption in 11 bus clock cycles.
- Encrypts 128 bits as a unit.
- All 128-bit data values accessed through a single block of 16 8-bit registers
- All accesses are on an 8-bit basis.

## 7.2 Block Diagram

A simple block diagram of the ASM module is shown in Figure 7-1.

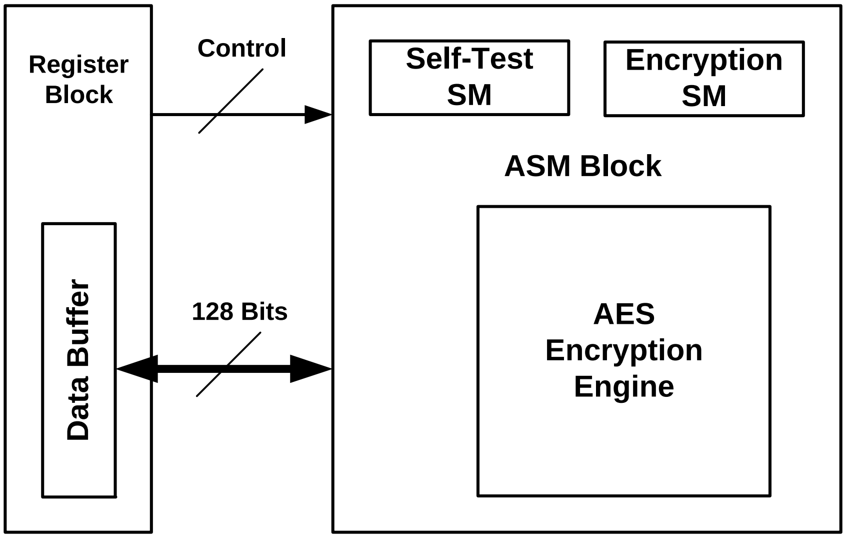


Figure 7-1. ASM Module Block Diagram

## 7.3 Functional Overview

The ASM block consists of an iterative encryption engine that is sequenced by an encryption state machine (SM) state machine or a self-test state machine. The block is controlled by mode bits that determine the encryption algorithm and data is exchanged between the MCU and the ASM Block via a single 128-bit buffer in the register block. Supporting the actual encryption engine is a register block that consists of two control/status registers and a bidirectional 16-byte data interface to provide data and constants. The ASM is clocked by the bus clock which always 1/2 the CPU clock rate (typically bus clock is 16 MHz).

### 7.3.1 Control Flow

The encryption or self-test process is sequenced by the appropriate SM. The general flow for the different encryption modes is:

1. Set desired encryption mode
2. Load required data components
3. Initiate the sequence
4. Wait for sequence completion
5. Read the resultant data component

### 7.3.1.1 Setting Encryption Mode

The encryption mode is based on the control bits listed in Figure 7-1. The defined mode of encryption is determined by setting the appropriate bit. Use of these bits during the encryption process is described in sub-sections of Section 7.4.

**Table 7-1. ASM Mode Control Bits**

| Bit | Description |
|---|---|
| SELFTEST | Enables self test |
| CTR | Enables Counter Mode encryption |
| CBC | Enables CBC-MAC Mode encryption/generation |
| AES | Enables simple AES Mode encryption |

### 7.3.1.2 Using the 128-Bit Data Interface

All required input data components and encryption results use a single 128-bit data interface buffer (16 byte-wide registers). The registers are described in Section 7.5.3, “AES 128 Data Registers (ASMDATA0 - ASMDATAF)” as to their organization and addresses. The use of the registers is controlled by the following fields:

- CLEAR Bit, AES Control Register 1 - writing a “1” to this bit clears all memory in the ASM, and the bit is self-clearing.
- Load_MAC Bit, AES Control Register 1 - writing a “1” to this bit loads the contents of the 128-bit data buffer into the ASM as the component selected by the “Data Register Type Select” field, and the bit is self-clearing.
- Data Register Type Select Field [2:0], AES Control Register 2 - this 3-bit field selects the target encryption component for a data write to the ASM (by writing a “1” to Load_MAC Bit), and also selects the encryption result component for an ASM data read. Table 7-2 lists use of data buffer versus the select field.

**Table 7-2. Data Buffer Use vs. Data_Reg_ype_Sel Field**

| Data_Reg_ Type_Sel[2:0] | Type | Mode | Description |
|---|---|---|---|
| 0 | Key | Write to ASM | Data buffer holds “Key” to be written for encryption |
| 1 | Data | Write to ASM | Data buffer holds “Data” to be written for encryption |
| 2 | CTR | Write to ASM | Data buffer holds “Counter” to be written for encryption |
| 3 | CTR Result | Read ASM | Reading buffer returns counter mode encryption result |
| 4 | CBC Result | Read ASM | Reading buffer returns CBC-MAC mode encryption result |
| 5 | MAC | Write ASM | Data buffer holds “Start Value” to be written for CBC-MAC encryption |
| 6 | AES Result | Read ASM | Reading buffer returns AES mode encryption result |
| 7 | N.A. | N.A. | Reserved (not used) |

For a write operation, the 128-bit buffer must first be loaded, the Data Register Type Select set, and finally the Load_MAC Bit written. For a read operation, the Data Register Type Select is first set, and a read of the 16 buffer registers yields the128-bit selected result.

#### 7.3.1.3 Initiating the Sequence and Getting Sequence Results

After the desired encryption mode is selected and the required encryption components are loaded, the sequence is initiated by writing the START Bit, AES Control Register 1. Other than self-test, all encryption sequences require 11 bus clock cycles. Completion of the encryption is signified by setting of the IRQ_FLAG status bit, AES Control Register 1.

The status flag can be enabled to generate an interrupt request or can be simply monitored by the CPU. The encryption results are read as described in Section 7.3.1.2. The ASM has been assigned Interrupt Vector No. 8.

### 7.3.2 Module Initialization

The ASM is unique in that it has a self-test mode, and it exits system reset disabled until the self-test mode has been run. After the self-test has been run and passes, the ASM is available for normal use. See Section 7.4.1 the description for self-test.

## 7.4 ASM Modes of Operation

The ASM performs Counter (CTR), Cipher Block Chaining (CBC), Counter with CBC-MAC (CCM), and plain AES encryption modes as well as self-test.

**NOTE**

In the following example flows for executing an encryption cycle:

- Writing a Key, Counter, MAC/Start, or Data value to the ASM implies a three step process; 1) Write the proper code to the Data_Reg_Type_Sel field, 2) Write the 128-bit value to the data interface registers, and 3) the Write Load_MAC bit to load the value from the buffer into the ASM.
- Reading an encryption result from the ASM implies a two step process; 1) Write the proper code to the Data_Reg_Type_Sel field, 2) Read the 128-bit return value from the data interface registers

### 7.4.1 Self-Test

As described, the ASM exits reset disabled and remains disabled until the self-test is run and is passed. Typical usage would require that this test be run only once after reset. This test can re-run at any time it becomes necessary to verify the operation of the encryption engine.

Self-test is run by:

1. Initialize - write SELFTEST bit to “1”
2. Start test - write START bit to “1” (bit is self-clearing)

3. Self-test runs - requires 3330 clocks to complete
4. Test completes:
   — Passes test: TSTPAS status flag is set and IRQ_FLAG status flag is set. An IRQ is asserted if it is enabled. The ASM is ready for normal use.
   — Fails test: TSTPAS remains low (clear) and IRQ_FLAG status flag is set. An IRQ is asserted if it is enabled. The ASM module is disabled and all outputs will be held to constant “0”. The self-test mode can be re-run.

**NOTE**

- Be sure to clear the SELFTEST bit before trying any other mode.
- It is also good practice to clear the ASM registers by writing to the CLEAR Bit of Control Register 1 immediately after self-test so that the ASM module is ready for any encryption mode.

### 7.4.2 Counter (CTR) Mode

Counter mode encryption (CTR) is done by using the AES engine as shown in the functional block diagram of Figure 7-2. Three 128-bit variables consisting of a Key, plain Text, and Counter values are each written to the encryption engine using the data buffer interface. The encryption cycle is initiated and requires 11 bus clocks to complete. The IRQ_FLAG status flag is set and an IRQ is asserted if it is enabled. The encrypted data can be then read from the data register.

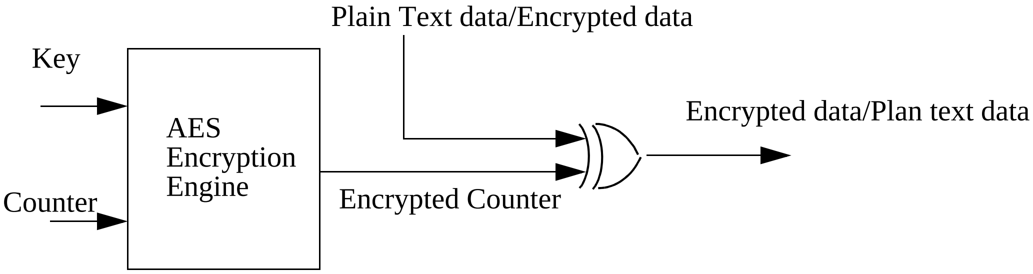


**Figure 7-2. CTR Mode Functional Block Diagram**

Decryption is possible for CTR mode and uses the same AES encryption engine configuration. The Key and Counter values remain the same, however, data to be decrypted is written to the data register. The decrypted result can be read from the data register similar to the encryption cycle.

A simple flow for CTR encryption consists of:

1. Write the Key value to the ASM.
2. Set the CTR bit in Control Register 1 - all other mode control bits must be clear
3. Write the Counter value to the ASM.
4. Write the plain Text value to the ASM.
5. Set the START bit in Control Register 1- this bit is a self clearing.
6. The Counter encryption runs to completion (11 bus clock cycles) -
   — The IRQ_FLAG status bit, AES Control Register 1 is set - an IRQ is generated is enabled.

- — The software may poll the status flag or wait for the interrupt

7. Read the encrypted CTR Result from the ASM

Steps 3-7 are repeated for each additional 128-bits plain text block. The Key value is re-used, and the Counter and plain Text values are new for each block. The IRQ_STATUS flag must be cleared after each iteration.

The decryption process is the same with the exception that the previously encoded 128-bit block is loaded in Step 4 and the results of Step 7 are now the plain text decrypted block.

### 7.4.3 Cipher Block Chaining (CBC) Mode

Cipher block chaining message authentication code (CBC or CBC-MAC) generation is done by using the AES engine as shown in the functional block diagram of Figure 7-3. Two 128-bit variables consisting of a Key and plain Text values are each written to the encryption engine using the data buffer interface. The encryption cycle is initiated and requires 11 bus clocks to complete. The IRQ_FLAG status flag is set and an IRQ is asserted if it is enabled. Additional blocks are encrypted as they are chained together. The final encrypted MAC result can be then read from the data register.

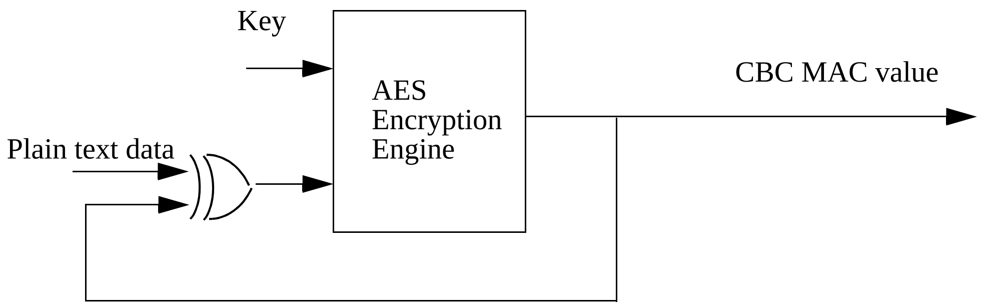


**Figure 7-3. CBC Mode Functional Block Diagram**

A simple flow for CBC-MAC encryption consists of:

1. Clear MAC accumulator as required - see NOTE below. This is required after self-test.
2. Write the Key value to the ASM.
3. Set the CBC bit in Control Register 1 - all other mode control bits must be clear
4. Write the plain Text value to the ASM.
5. Set the START bit in Control Register 1- this bit is a self clearing.
6. The CBC encryption runs to completion (11 bus clock cycles) -
   - — The IRQ_FLAG status bit, AES Control Register 1 is set - an IRQ is generated is enabled.
   - — The software may poll the status flag or wait for the interrupt
7. Repeat Steps 4-6 for each additional 128-bit plain text block - the Key value is re-used, and the plain Text value is new for each block. The IRQ_STATUS flag must be cleared before each iteration. This sequence accomplishes the block chaining.
8. Read the encrypted 128-bit MAC (authentication code) Result (Data_Reg_Type_Sel field to 0x4) from the ASM

**NOTE**

- The MAC accumulator must be cleared to prepare for a new encryption payload. Writing a 1 to the CLEAR bit will set the MAC accumulator back to all zeroes. The clear bit is self clearing.
- If for some reason the software needs to continue a MAC calculation from a previously saved value, the MAC accumulator can be pre-loaded with a initial value. Program Control register2[7:5] to 3’b101 now the value that must be loaded into the accumulator is written to data registers and then a 1 is written to the load_mac bit. The load_mac bit is self clearing.

### 7.4.4 Counter With CBC-MAC (CCM) Mode

CCM mode is the combination of using CTR mode for protecting the privacy of data, and using CBC mode to generate a MAC to protect the data from unauthorized modifications. In this mode both CTR and CBC mode bits are set and the ASM first performs a CTR encryption followed by a CBC conversion. The results of CTR and CBC conversion are stored in separate registers which can be read separately from another.

A simple flow for CCM encryption consists of:

1. Clear MAC accumulator as required - see NOTE below. This is required after self-test.
2. Write the Key value to the ASM.
3. Set the CTR and CBC bits in Control Register 1 - all other mode control bits must be clear
4. Write the Counter value to the ASM.
5. Write the plain Text value to the ASM.
6. Set the START bit in Control Register 1- this bit is a self clearing.
7. The CTR/CBC-MAC encryption runs to completion -
   — The IRQ_FLAG status bit, AES Control Register 1 is set - an IRQ is generated is enabled.
   — The software may poll the status flag or wait for the interrupt
8. Read the encrypted CTR Result from the ASM
9. Clear the IRQ_FLAG status bit.
10. Repeat Steps 4-9 for each additional 128-bit plain text block - the Key value is re-used, and the Counter and plain Text values are new for each block.
11. Read the encrypted 128-bit MAC (authentication code) Result (Data_Reg_Type_Sel field to 0x4) from the ASM

### 7.4.5 Plain AES Mode

AES mode is done by using the AES engine as shown in the functional block diagram of Figure 7-4. This mode is the most simple method of encryption in that there is no feedback of results back to modify the input in any form. Two 128-bit variables consisting of a Key and plain Text, values are each written to the encryption engine using the data buffer interface. The encryption cycle is initiated and requires 11 bus clocks to complete. The IRQ_FLAG status flag is set and an IRQ is asserted if it is enabled. The encrypted data can be then read from the data register.

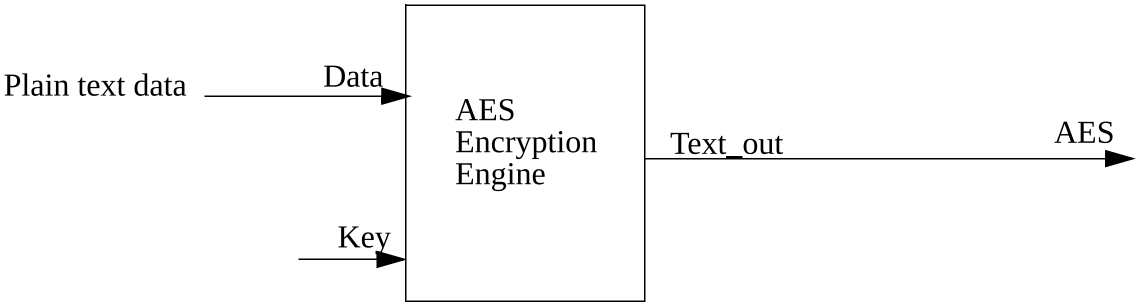


Figure 7-4. Plain AES Mode Functional Block Diagram

A simple flow for AES encryption consists of:

1. Write the Key value to the ASM.
2. Set the AES bit in Control Register 1 - all other mode control bits must be clear
3. Write the plain Text value to the ASM.
4. Set the START bit in Control Register 1- this bit is a self clearing.
5. The AES encryption runs to completion (11 bus clock cycles) -
    - The IRQ_FLAG status bit, AES Control Register 1 is set - an IRQ is generated is enabled.
    - The software may poll the status flag or wait for the interrupt
6. Read the encrypted AES Result from the ASM

Steps 3-6 are repeated for each additional 128-bit plain text block. The Key value is re-used, and the plain Text value is new for each block. The IRQ_STATUS flag must be cleared after each iteration.

## 7.5 ASM Register Definition

AES128 register block consists of two control/status registers and a 16-byte block of data registers for 128-bit data values. These are described below. The ASM registers are all located in the Direct-Page Register Map.

### 7.5.1 AES 128 Control 1 Register (CONTROL1)

Control Register 1 contains various control bits for the ASM.

CONTROL1 0x0020

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CLEAR | START | SELFTST | CTR | CBC | AES | LOAD_MAC | AES_MSK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| self clear | yes | yes | no | no | no | no | yes | no |

▨ = Unimplemented or Reserved

**Figure 7-5. AES Address Register CONTROL1**

**Table 7-3. AES128 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CLEAR | **Clear ASM —** Write 1 to this bit to clear all memory in ASM. Writing a 0 does nothing; the bit is self-clearing |
| 6<br>START | **Start —** Write 1 to start any encryption or self-test sequence. Writing a 0 does nothing; the bit is self-clearing |
| 5<br>SELFTST | **Self-Test Mode —** This bit enables self-test mode. Sequence is initiated by writing START bit.<br>1 = Self-test mode enabled<br>0 = Self-test mode not enabled |
| 4<br>CTR | **Counter Mode —** This bit enables Counter encryption mode. Sequence is initiated by writing START bit.<br>1 = Counter mode encryption enabled<br>0 = Counter mode encryption not enabled |
| 3<br>CBC | **CBC Mode —** This bit enables CBC-MAC encryption mode. Sequence is initiated by writing START bit<br>1 = CBC-MAC generation enabled.<br>0 = CBC-MAC generation not enabled |
| 2<br>AES | **AES Mode —** This bit enables simple AES encryption mode. Sequence is initiated by writing START bit.<br>1 = AES mode encryption enabled<br>0 = AES mode encryption not enabled |

Table 7-3. AES128 Field Descriptions (continued)

| Field | Description |
|---|---|
| 1<br>Load_Mac | **Load ASM Data —** Writing this bit to “1” loads the contents of the 128-bit data buffer into the destination selected by the Data_Reg_Type_Select[2:0] field. See Section 7.3.1.2, “Using the 128-Bit Data Interface”. Writing a “0” does nothing; the bit is self-clearing |
| 0<br>AES_MSK | **AES128 Interrupt Mask —** This bit enables generation of an ASM interrupt request if the IRQ_FLAG is set upon completion of an ASM operation<br>0 = IRQ enabled<br>1 = IRQ disabled |

## 7.5.2 AES 128 Control Register 2 (CONTROL2)

Control Register 2 contains the data register type select field and status flags for the ASM

CONTROL2 0x0021

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DATA_REG_TYPE_SELECT | | | 0 | 0 | IRQ_FLAG | TSTPAS | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 7-6. AES Control 2 Register

Table 7-4. AES Control 2 Field Descriptions

| Field | Description |
|---|---|
| 7:5<br>DATA_REG_<br>TYPE_SELECT<br>[2:0] | **Data Register Type Select —** This register selects the destination type of data to be written or the source of read data from the 16x8 Data register field. For coding see Section 7.3.1.2, “Using the 128-Bit Data Interface” and Table 7-2. |
| 2<br>IRQ_FLAG | **ASM Completion Status Flag —** A 1 indicates the completion of a ASM operation (self-test or encryption cycle). After it set, the status must be cleared by writing a 1 |
| 1<br>TSTPAS | **Self Test Pass —** This bit is set to 1 upon correct completion of the self test. If self-test fails then the module is not usable and the outputs will be set to all 0. The default state for this bit is 0. The software must initiate a self-test before the module can be used.<br>1 = Self-test has passed<br>0 = Self-test has failed |

### 7.5.3 AES 128 Data Registers (ASMDATA0 - ASMDATAF)

The 16x8-bit AES Data registers (128 bits) block serve as the interface between the ASM's internal memory segments and the MCU. The destination or source of the data in these registers is controlled through the Data_Reg_Type_Select field in AES Control 2 Register. See Section 7.3.1.2, "Using the 128-Bit Data Interface". The maps is as follows:

**Table 7-5. AES Data Registers**

| Register | Field Data |
|---|---|
| ASMDATA_0 | [127:120] most significant byte |
| ASMDATA_1 | [119:112] |
| ASMDATA_2 | [111:104] |
| ASMDATA_3 | [103:96] |
| ASMDATA_4 | [95:88] |
| ASMDATA_5 | [87:80] |
| ASMDATA_6 | [79:72] |
| ASMDATA_7 | [71:64] |
| ASMDATA_8 | [63:56] |
| ASMDATA_9 | [55:48] |
| ASMDATA_A | [47:40] |
| ASMDATA_B | [39:32] |
| ASMDATA_C | [31:24] |
| ASMDATA_D | [23:16] |
| ASMDATA_E | [15:8] |
| ASMDATA_F | [7:0] least significant byte |

| ADDRESS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x00**22** | DATA[127:120] | | | | | | | |
| 0x00**23** | DATA[119:112] | | | | | | | |
| 0x00**24** | DATA[111:104] | | | | | | | |
| 0x00**25** | DATA[103:96] | | | | | | | |
| 0x00**26** | DATA[95:88] | | | | | | | |
| 0x00**27** | DATA[87:80] | | | | | | | |
| 0x00**28** | DATA[79:72] | | | | | | | |
| 0x00**29** | DATA[70:64] | | | | | | | |
| 0x00**2A** | DATA[63:56] | | | | | | | |
| 0x00**2B** | DATA[55:48] | | | | | | | |
| 0x00**2C** | DATA[47:40] | | | | | | | |
| 0x00**2D** | DATA[39:32] | | | | | | | |
| 0x00**2E** | DATA[31:24] | | | | | | | |
| 0x00**2F** | DATA[23:16] | | | | | | | |
| 0x00**30** | DATA[15:8] | | | | | | | |
| 0x00**31** | DATA[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | | | | | | | | |

**Figure 7-7. AES 128 Data Registers [Data_0:Data_F]**

**Table 7-6. AES 128 Data Registers Field Description**

| Field | Description |
|---|---|
| DATA<br>[127:0] | **AES 128 Data** — 128-bit AES data interface. |

# Chapter 8
# CPU

## 8.1 Introduction

This chapter provides summary information about the registers, addressing modes, and instruction set of the CPU which executes a superset of the 68HC08 instruction set. It has an 8-bit data bus, a 16-bit address bus and a 2-stage instruction pipe that facilitates the overlapping of instruction fetching and execution. For a more detailed description, see the *HCS08 Family Reference Manual, volume 1,* Freescale Semiconductor document order number HCS08RMV1/D

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

### 8.1.1 Features

Features of the HCS08 CPU include:

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

- STOP and WAIT instructions to invoke low-power operating modes

### 8.1.2 Modes of Operation

The CPU is affected by the device mode of operation:

- Run mode - normal operation. The CPU clock is set by the prescaler, programmed by the **RDIV[2:0]** field of the System Oscillator Management and Control Register 1 (SOMC1)
- LPRun modes - CPU clock is reduce to 32 MHz reference oscillator divided by 64 (500 kHz). Reduced clock affects operation of time based peripherals.
- Wait and Stop modes - the CPU clock is disabled and the CPU does not run.
- Background Debug mode. The CPU is halted in BDM mode.
- CPU can be suspended by flash (reset, BIST mode and system security word access)
- DMA <> RAM accesses - DMA cycles work on a cycle steal basis and take bus clocks from the CPU.

## 8.2 Programmer’s Model and CPU Register Definition

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

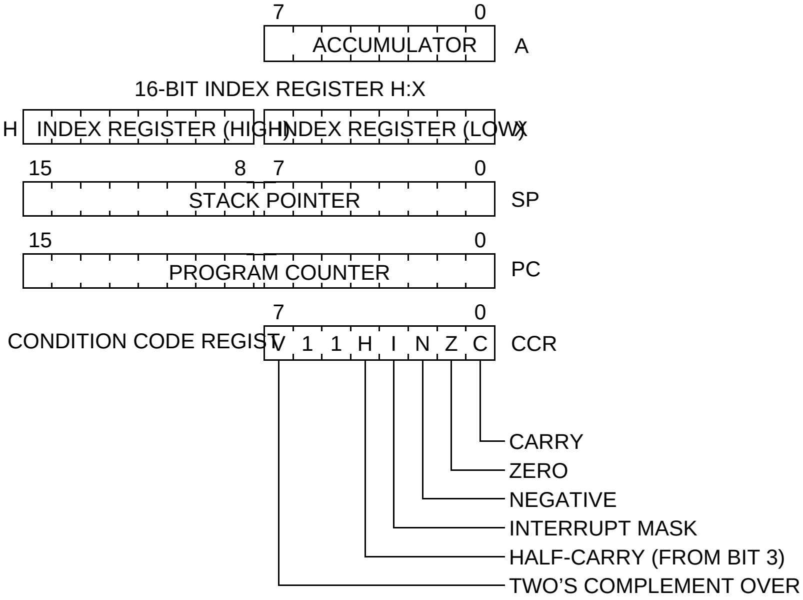


Figure 8-1. CPU Registers

### 8.2.1 Accumulator (A)

The A accumulator is a general-purpose 8-bit register. One operand input to the arithmetic logic unit (ALU) is connected to the accumulator and the ALU results are often stored into the A accumulator after arithmetic and logical operations. The accumulator can be loaded from memory using various addressing modes to specify the address where the loaded data comes from, or the contents of A can be stored to memory using various addressing modes to specify the address where data from A will be stored. Reset has no effect on the contents of the A accumulator.

### 8.2.2 Index Register (H:X)

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

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

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

### 8.2.3 Stack Pointer (SP)

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

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

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

### 8.2.4 Program Counter (PC)

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

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

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

### 8.2.5 Condition Code Register (CCR)

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

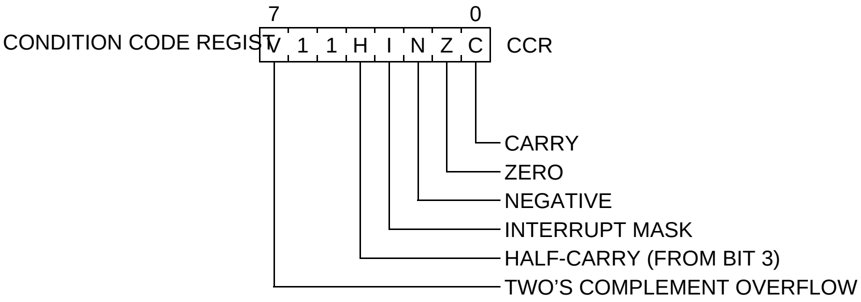


**Figure 8-2. Condition Code Register**

**Table 8-1. CCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>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.<br>0 No overflow<br>1 Overflow |
| 4<br>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.<br>0 No carry between bits 3 and 4<br>1 Carry between bits 3 and 4 |
| 3<br>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.<br>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.<br>0 Interrupts enabled<br>1 Interrupts disabled |
| 2<br>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.<br>0 Non-negative result<br>1 Negative result |

Table 8-1. CCR Register Field Descriptions (continued)

| Field | Description |
|---|---|
| 1<br>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.<br>0 Non-zero result<br>1 Zero result |
| 0<br>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.<br>0 No carry out of bit 7<br>1 Carry out of bit 7 |

## 8.3 Addressing Modes

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

The HCS08 has larger than 64-Kbytes of memory so it has 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.

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.

A summary of the addressing modes used by the core is given in Table 8-2.

Table 8-2. Addressing Mode Summary

| Addressing Mode | Abbreviation | Description |
|---|---|---|
| Inherent | INH | Operands (if any) are in CPU registers. |
| Immediate | IMM | Operand is included in byte(s) following the opcode; 8-bit or 16-bit size implied by context. |
| Direct | DIR | The operand is fetched from a direct page address given by the byte following the opcode. |

Table 8-2. Addressing Mode Summary (continued)

| Addressing Mode | Abbreviation | Description |
|---|---|---|
| Extended | EXT | The operand is fetched from an address given by the 2 bytes following the opcode (most significant first). |
| Relative | REL | Byte following the opcode is an 8-bit signed offset that is added to PC to obtain the effective address. |
| Indexed, no offset | IX | Effective address is contained in H:X. |
| Indexed, no offset, post increment | IX+ | Effective address is contained in H:X, which is incremented after the operand is accessed. |
| Indexed, 8-bit offset | IX1 | Byte following the opcode is an 8-bit offset that is added to H:X to obtain the effective address. |
| Indexed, 8-bit offset, post increment | IX1+ | Same as above, but H:X is incremented after the operand is accessed. |
| Indexed, 16 bit offset | IX2 | The 2 bytes following the opcode form a 16-bit offset (most significant first) that is added to H:X to obtain the effective address. |
| Direct to direct | DD | Used to move data inside direct page. First byte following the opcode is the source address. Second byte is the destination address. |
| Immediate to direct | IMD | Used to move an 8-bit constant to a direct page location. First byte following the opcode is the constant. Second byte is the destination address. |
| Indexed to direct, post increment | IX+D | Accesses a source operand addressed by H:X and a destination location within the direct page addressed by the byte following the opcode. H:X is then incremented. |
| Direct to indexed, post increment | DIX+ | Accesses a source operand addressed by the byte following the opcode and a destination location addressed by H:X, which is then incremented. |
| Stack Pointer, 8-bit offset | SP1 | Effective address is the value in SP added to the byte following the opcode. |
| Stack Pointer, 16-bit offset | SP2 | Effective address is the value in SP added to the word formed by the 2 bytes following the opcode (most significant first). |

### 8.3.1 Inherent Addressing Mode (INH)

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

### 8.3.2 Relative Addressing Mode (REL)

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

### 8.3.3 Immediate Addressing Mode (IMM)

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

### 8.3.4 Direct Addressing Mode (DIR)

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

### 8.3.5 Extended Addressing Mode (EXT)

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

### 8.3.6 Indexed Addressing Mode

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 8.4 Special Operations

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

### 8.4.1 Reset Sequence

Reset can be caused by a power-on-reset (POR) event, internal conditions such as the COP (computer operating properly) watchdog, or by assertion of an external active-low reset pin. When a reset event occurs, the CPU immediately stops whatever it is doing (the MCU does not wait for an instruction boundary before responding to a reset event).

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 0xFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

### 8.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

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

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

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

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

### 8.4.3 Wait Mode Operation

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

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

### 8.4.4 Stop Mode Operation

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

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

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode.

## 8.4.5 BGND Instruction

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

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

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.

Because 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. Because 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, because 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.

## 8.5 HCS08 Instruction Set Summary

### Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in Table 8-3.

### Operators

( ) = Contents of register or memory location shown inside parentheses

← = Is loaded with (read: “gets”)

& = Boolean AND

| = Boolean OR

⊕ = Boolean exclusive-OR

× = Multiply

÷ = Divide

: = Concatenate

+ = 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 |
| Þ | = | 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.

*oprx8* — Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing *oprx16* — Any label or expression that evaluates to a 16-bit value. Because the HCS08 has a 16-bit address bus, this can be either a signed or an unsigned value.

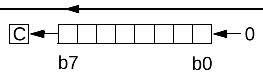
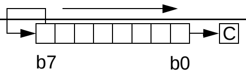
*page* — Any label or expression that evaluates to a valid bank number for the PPAGE register. For a 128-Kbyte derivative, any value between 0 and 7 is valid.

*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 8-3. HCS08 Instruction Set Summary (Sheet 1 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| ADC *#opr8i*<br>ADC *opr8a*<br>ADC *opr16a*<br>ADC *oprx16*,X<br>ADC *oprx8*,X<br>ADC ,X<br>ADC *oprx16*,SP<br>ADC *oprx8*,SP | Add with Carry | A ← (A) + (M) + (C) | Þ | Þ | – | Þ | Þ | Þ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A9<br>B9<br>C9<br>D9<br>E9<br>F9<br>9ED9<br>9EE9 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| ADD *#opr8i*<br>ADD *opr8a*<br>ADD *opr16a*<br>ADD *oprx16*,X<br>ADD *oprx8*,X<br>ADD ,X<br>ADD *oprx16*,SP<br>ADD *oprx8*,SP | Add without Carry | A ← (A) + (M) | Þ | Þ | – | Þ | Þ | Þ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AB<br>BB<br>CB<br>DB<br>EB<br>FB<br>9EDB<br>9EEB | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| AIS *#opr8i* | Add Immediate Value (Signed) to Stack Pointer | SP ← (SP) + (M)<br>M is sign extended to a 16-bit value | – | – | – | – | – | – | IMM | A7 | ii | 2 |
| AIX *#opr8i* | Add Immediate Value (Signed) to Index Register (H:X) | H:X ← (H:X) + (M)<br>M is sign extended to a 16-bit value | – | – | – | – | – | – | IMM | AF | ii | 2 |
| AND *#opr8i*<br>AND *opr8a*<br>AND *opr16a*<br>AND *oprx16*,X<br>AND *oprx8*,X<br>AND ,X<br>AND *oprx16*,SP<br>AND *oprx8*,SP | Logical AND | A ← (A) & (M) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A4<br>B4<br>C4<br>D4<br>E4<br>F4<br>9ED4<br>9EE4 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| ASL *opr8a*<br>ASLA<br>ASLX<br>ASL *oprx8*,X<br>ASL ,X<br>ASL *oprx8*,SP | Arithmetic Shift Left<br>(Same as LSL) |  | Þ | – | – | Þ | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| ASR *opr8a*<br>ASRA<br>ASRX<br>ASR *oprx8*,X<br>ASR ,X<br>ASR *oprx8*,SP | Arithmetic Shift Right |  | Þ | – | – | Þ | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 37<br>47<br>57<br>67<br>77<br>9E67 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 2 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BCC *rel* | Branch if Carry Bit Clear | Branch if (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |
| BCLR *n,opr8a* | Clear Bit n in Memory | Mn ← 0 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 11<br>13<br>15<br>17<br>19<br>1B<br>1D<br>1F | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BCS *rel* | Branch if Carry Bit Set (Same as BLO) | Branch if (C) = 1 | – | – | – | – | – | – | REL | 25 | rr | 3 |
| BEQ *rel* | Branch if Equal | Branch if (Z) = 1 | – | – | – | – | – | – | REL | 27 | rr | 3 |
| BGE *rel* | Branch if Greater Than or Equal To (Signed Operands) | Branch if (N ⊕ 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 *rel* | Branch if Greater Than (Signed Operands) | Branch if (Z) \| (N ⊕ V) = 0 | – | – | – | – | – | – | REL | 92 | rr | 3 |
| BHCC *rel* | Branch if Half Carry Bit Clear | Branch if (H) = 0 | – | – | – | – | – | – | REL | 28 | rr | 3 |
| BHCS *rel* | Branch if Half Carry Bit Set | Branch if (H) = 1 | – | – | – | – | – | – | REL | 29 | rr | 3 |
| BHI *rel* | Branch if Higher | Branch if (C) \| (Z) = 0 | – | – | – | – | – | – | REL | 22 | rr | 3 |
| BHS *rel* | Branch if Higher or Same (Same as BCC) | Branch if (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |
| BIH *rel* | Branch if IRQ Pin High | Branch if IRQ pin = 1 | – | – | – | – | – | – | REL | 2F | rr | 3 |
| BIL *rel* | Branch if IRQ Pin Low | Branch if IRQ pin = 0 | – | – | – | – | – | – | REL | 2E | rr | 3 |

**Table 8-3. HCS08 Instruction Set Summary (Sheet 3 of 11)**

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BIT *#opr8i*<br>BIT *opr8a*<br>BIT *opr16a*<br>BIT *oprx16*,X<br>BIT *oprx8*,X<br>BIT ,X<br>BIT *oprx16*,SP<br>BIT *oprx8*,SP | Bit Test | (A) & (M)<br>(CCR Updated but Operands Not Changed) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A5<br>B5<br>C5<br>D5<br>E5<br>F5<br>9ED5<br>9EE5 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| BLE *rel* | Branch if Less Than or Equal To (Signed Operands) | Branch if (Z) \| (N ⊕ V) = 1 | – | – | – | – | – | – | REL | 93 | rr | 3 |
| BLO *rel* | Branch if Lower (Same as BCS) | Branch if (C) = 1 | – | – | – | – | – | – | REL | 25 | rr | 3 |
| BLS *rel* | Branch if Lower or Same | Branch if (C) \| (Z) = 1 | – | – | – | – | – | – | REL | 23 | rr | 3 |
| BLT *rel* | Branch if Less Than (Signed Operands) | Branch if (N ⊕ V ) = 1 | – | – | – | – | – | – | REL | 91 | rr | 3 |
| BMC *rel* | Branch if Interrupt Mask Clear | Branch if (I) = 0 | – | – | – | – | – | – | REL | 2C | rr | 3 |
| BMI *rel* | Branch if Minus | Branch if (N) = 1 | – | – | – | – | – | – | REL | 2B | rr | 3 |
| BMS *rel* | Branch if Interrupt Mask Set | Branch if (I) = 1 | – | – | – | – | – | – | REL | 2D | rr | 3 |
| BNE *rel* | Branch if Not Equal | Branch if (Z) = 0 | – | – | – | – | – | – | REL | 26 | rr | 3 |
| BPL *rel* | Branch if Plus | Branch if (N) = 0 | – | – | – | – | – | – | REL | 2A | rr | 3 |
| BRA *rel* | Branch Always | No Test | – | – | – | – | – | – | REL | 20 | rr | 3 |
| BRCLR *n*,*opr8a*,*rel* | Branch if Bit *n* in Memory Clear | Branch if (Mn) = 0 | – | – | – | – | – | Þ | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 01<br>03<br>05<br>07<br>09<br>0B<br>0D<br>0F | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |

Table 8-3. HCS08 Instruction Set Summary (Sheet 4 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BRN *rel* | Branch Never | Uses 3 Bus Cycles | – | – | – | – | – | – | REL | 21 | rr | 3 |
| BRSET *n*,*opr8a*,*rel* | Branch if Bit *n* in Memory Set | Branch if (Mn) = 1 | – | – | – | – | – | Þ | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 00<br>02<br>04<br>06<br>08<br>0A<br>0C<br>0E | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSET *n*,*opr8a* | Set Bit *n* in Memory | Mn ← 1 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 10<br>12<br>14<br>16<br>18<br>1A<br>1C<br>1E | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSR *rel* | Branch to Subroutine | PC ← (PC) + 0x0002<br>push (PCL); SP ← (SP) – 0x0001<br>push (PCH); SP ← (SP) – 0x0001<br>PC ← (PC) + *rel* | – | – | – | – | – | – | REL | AD | rr | 5 |
| CALL page, opr16a | Call Subroutine | PC ← PC + 4<br>Push (PCL); SP ← (SP) – 0x0001<br>Push (PCH); SP ← (SP) – 0x0001<br>Push (PPAGE); SP ← (SP) – 0x0001<br>PPAGE ← page<br>PC ← Unconditional Address | – | – | – | – | – | – | EXT | AC | pghll | 8 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 5 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| CBEQ *opr8a*,*rel*<br>CBEQA *#opr8i*,*rel*<br>CBEQX *#opr8i*,*rel*<br>CBEQ *oprx8*,X+,*rel*<br>CBEQ ,X+,*rel*<br>CBEQ *oprx8*,SP,*rel* | Compare and Branch if Equal | Branch if (A) = (M)<br>Branch if (A) = (M)<br>Branch if (X) = (M)<br>Branch if (A) = (M)<br>Branch if (A) = (M)<br>Branch if (A) = (M) | – | – | – | – | – | – | DIR<br>IMM<br>IMM<br>IX1+<br>IX+<br>SP1 | 31<br>41<br>51<br>61<br>71<br>9E61 | dd rr<br>ii rr<br>ii rr<br>ff rr<br>rr<br>ff rr | 5<br>4<br>4<br>5<br>5<br>6 |
| CLC | Clear Carry Bit | C ← 0 | – | – | – | – | – | 0 | INH | 98 | | 1 |
| CLI | Clear Interrupt Mask Bit | I ← 0 | – | – | 0 | – | – | – | INH | 9A | | 1 |
| CLR *opr8a*<br>CLRA<br>CLRX<br>CLRH<br>CLR *oprx8*,X<br>CLR ,X<br>CLR *oprx8*,SP | Clear | M ← 0x00<br>A ← 0x00<br>X ← 0x00<br>H ← 0x00<br>M ← 0x00<br>M ← 0x00<br>M ← 0x00 | 0 | – | – | 0 | 1 | – | DIR<br>INH<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3F<br>4F<br>5F<br>8C<br>6F<br>7F<br>9E6F | dd<br><br><br><br>ff<br><br>ff | 5<br>1<br>1<br>1<br>5<br>4<br>6 |
| CMP *#opr8i*<br>CMP *opr8a*<br>CMP *opr16a*<br>CMP *oprx16*,X<br>CMP *oprx8*,X<br>CMP ,X<br>CMP *oprx16*,SP<br>CMP *oprx8*,SP | Compare Accumulator with Memory | (A) – (M)<br>(CCR Updated But Operands Not Changed) | Þ | – | – | Þ | Þ | Þ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A1<br>B1<br>C1<br>D1<br>E1<br>F1<br>9ED1<br>9EE1 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| COM *opr8a*<br>COMA<br>COMX<br>COM *oprx8*,X<br>COM ,X<br>COM *oprx8*,SP | Complement (One’s Complement) | M ← ($\overline{M}$)= 0xFF – (M)<br>A ← ($\overline{A}$) = 0xFF – (A)<br>X ← ($\overline{X}$) = 0xFF – (X)<br>M ← ($\overline{M}$) = 0xFF – (M)<br>M ← ($\overline{M}$) = 0xFF – (M)<br>M ← ($\overline{M}$) = 0xFF – (M) | 0 | – | – | Þ | Þ | 1 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 33<br>43<br>53<br>63<br>73<br>9E63 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| CPHX *opr16a*<br>CPHX *#opr16i*<br>CPHX *opr8a*<br>CPHX *oprx8*,SP | Compare Index Register (H:X) with Memory | (H:X) – (M:M + 0x0001)<br>(CCR Updated But Operands Not Changed) | Þ | – | – | Þ | Þ | Þ | EXT<br>IMM<br>DIR<br>SP1 | 3E<br>65<br>75<br>9EF3 | hh ll<br>jj kk<br>dd<br>ff | 6<br>3<br>5<br>6 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 6 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| CPX *#opr8i*<br>CPX *opr8a*<br>CPX *opr16a*<br>CPX *oprx16*,X<br>CPX *oprx8*,X<br>CPX ,X<br>CPX *oprx16*,SP<br>CPX *oprx8*,SP | Compare X (Index Register Low) with Memory | (X) – (M)<br>(CCR Updated But Operands Not Changed) | Þ | – | – | Þ | Þ | Þ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A3<br>B3<br>C3<br>D3<br>E3<br>F3<br>9ED3<br>9EE3 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| DAA | Decimal Adjust Accumulator After ADD or ADC of BCD Values | $(A)_{10}$ | U | – | – | Þ | Þ | Þ | INH | 72 | | 1 |
| DBNZ *opr8a*,*rel*<br>DBNZA *rel*<br>DBNZX *rel*<br>DBNZ *oprx8*,X,*rel*<br>DBNZ ,X,*rel*<br>DBNZ *oprx8*,SP,*rel* | Decrement and Branch if Not Zero | Decrement A, X, or M<br>Branch if (result) ≠ 0<br>DBNZX Affects X Not H | – | – | – | – | – | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3B<br>4B<br>5B<br>6B<br>7B<br>9E6B | dd rr<br>rr<br>rr<br>ff rr<br>rr<br>ff rr | 7<br>4<br>4<br>7<br>6<br>8 |
| DEC *opr8a*<br>DECA<br>DECX<br>DEC *oprx8*,X<br>DEC ,X<br>DEC *oprx8*,SP | Decrement | M ← (M) – 0x01<br>A ← (A) – 0x01<br>X ← (X) – 0x01<br>M ← (M) – 0x01<br>M ← (M) – 0x01<br>M ← (M) – 0x01 | Þ | – | – | Þ | Þ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3A<br>4A<br>5A<br>6A<br>7A<br>9E6A | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| DIV | Divide | A ← (H:A)÷(X)<br>H ← Remainder | – | – | – | – | Þ | Þ | INH | 52 | | 6 |
| EOR *#opr8i*<br>EOR *opr8a*<br>EOR *opr16a*<br>EOR *oprx16*,X<br>EOR *oprx8*,X<br>EOR ,X<br>EOR *oprx16*,SP<br>EOR *oprx8*,SP | Exclusive OR Memory with Accumulator | A ← (A ⊕ M) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A8<br>B8<br>C8<br>D8<br>E8<br>F8<br>9ED8<br>9EE8 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| INC *opr8a*<br>INCA<br>INCX<br>INC *oprx8*,X<br>INC ,X<br>INC *oprx8*,SP | Increment | M ← (M) + 0x01<br>A ← (A) + 0x01<br>X ← (X) + 0x01<br>M ← (M) + 0x01<br>M ← (M) + 0x01<br>M ← (M) + 0x01 | Þ | – | – | Þ | Þ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3C<br>4C<br>5C<br>6C<br>7C<br>9E6C | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 7 of 11)

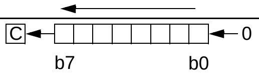
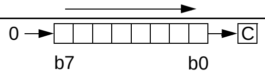
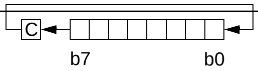
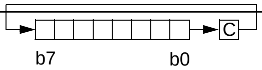
| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| JMP *opr8a*<br>JMP *opr16a*<br>JMP *oprx16*,X<br>JMP *oprx8*,X<br>JMP ,X | Jump | PC ← Jump Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BC<br>CC<br>DC<br>EC<br>FC | dd<br>hh ll<br>ee ff<br>ff | 3<br>4<br>4<br>3<br>3 |
| JSR *opr8a*<br>JSR *opr16a*<br>JSR *oprx16*,X<br>JSR *oprx8*,X<br>JSR ,X | Jump to Subroutine | PC ← (PC) + *n* (*n* = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – 0x0001<br>Push (PCH); SP ← (SP) – 0x0001<br>PC ← Unconditional Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BD<br>CD<br>DD<br>ED<br>FD | dd<br>hh ll<br>ee ff<br>ff | 5<br>6<br>6<br>5<br>5 |
| LDA *#opr8i*<br>LDA *opr8a*<br>LDA *opr16a*<br>LDA *oprx16*,X<br>LDA *oprx8*,X<br>LDA ,X<br>LDA *oprx16*,SP<br>LDA *oprx8*,SP | Load Accumulator from Memory | A ← (M) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A6<br>B6<br>C6<br>D6<br>E6<br>F6<br>9ED6<br>9EE6 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| LDHX *#opr16i*<br>LDHX *opr8a*<br>LDHX *opr16a*<br>LDHX ,X<br>LDHX *oprx16*,X<br>LDHX *oprx8*,X<br>LDHX *oprx8*,SP | Load Index Register (H:X) from Memory | H:X ← (M:M + 0x0001) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX<br>IX2<br>IX1<br>SP1 | 45<br>55<br>32<br>9EAE<br>9EBE<br>9ECE<br>9EFE | jj<br>kk<br>dd<br>hh ll<br><br>ee ff<br>ff<br>ff | 3<br>4<br>5<br>5<br>6<br>5<br>5 |
| LDX *#opr8i*<br>LDX *opr8a*<br>LDX *opr16a*<br>LDX *oprx16*,X<br>LDX *oprx8*,X<br>LDX ,X<br>LDX *oprx16*,SP<br>LDX *oprx8*,SP | Load X (Index Register Low) from Memory | X ← (M) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AE<br>BE<br>CE<br>DE<br>EE<br>FE<br>9EDE<br>9EEE | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| LSL *opr8a*<br>LSLA<br>LSLX<br>LSL *oprx8*,X<br>LSL ,X<br>LSL *oprx8*,SP | Logical Shift Left (Same as ASL) |  | Þ | – | – | Þ | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |

Table 8-3. HCS08 Instruction Set Summary (Sheet 8 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| LSR *opr8a*<br>LSRA<br>LSR*X*<br>LSR *oprx8*,X<br>LSR ,X<br>LSR *oprx8*,SP | Logical Shift Right |  | Þ | – | – | 0 | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 34<br>44<br>54<br>64<br>74<br>9E64 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| MOV *opr8a*,*opr8a*<br>MOV *opr8a*,X+<br>MOV *#opr8i*,*opr8a*<br>MOV ,X+,*opr8a* | Move | $(M)_{destination} \leftarrow (M)_{source}$<br><br>H:X ← (H:X) + 0x0001 in IX+/DIR and DIR/IX+ Modes | 0 | – | – | Þ | Þ | – | DIR/DIR<br>DIR/IX+<br>IMM/DIR<br>IX+/DIR | 4E<br>5E<br>6E<br>7E | dd dd<br>dd<br>ii dd<br>dd | 5<br>5<br>4<br>5 |
| MUL | Unsigned multiply | X:A ← (X) × (A) | – | 0 | – | – | – | 0 | INH | 42 | | 5 |
| NEG *opr8a*<br>NEGA<br>NEGX<br>NEG *oprx8*,X<br>NEG ,X<br>NEG *oprx8*,SP | Negate (Two's Complement) | M ← – (M) = 0x00 – (M)<br>A ← – (A) = 0x00 – (A)<br>X ← – (X) = 0x00 – (X)<br>M ← – (M) = 0x00 – (M)<br>M ← – (M) = 0x00 – (M)<br>M ← – (M) = 0x00 – (M) | Þ | – | – | Þ | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 30<br>40<br>50<br>60<br>70<br>9E60 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| NOP | No Operation | Uses 1 Bus Cycle | – | – | – | – | – | – | INH | 9D | | 1 |
| NSA | Nibble Swap Accumulator | A ← (A[3:0]:A[7:4]) | – | – | – | – | – | – | INH | 62 | | 1 |
| ORA *#opr8i*<br>ORA *opr8a*<br>ORA *opr16a*<br>ORA *oprx16*,X<br>ORA *oprx8*,X<br>ORA ,X<br>ORA *oprx16*,SP<br>ORA *oprx8*,SP | Inclusive OR Accumulator and Memory | A ← (A) \| (M) | 0 | – | – | Þ | Þ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AA<br>BA<br>CA<br>DA<br>EA<br>FA<br>9EDA<br>9EEA | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| PSHA | Push Accumulator onto Stack | Push (A); SP ← (SP) – 0x0001 | – | – | – | – | – | – | INH | 87 | | 2 |
| PSHH | Push H (Index Register High) onto Stack | Push (H); SP ← (SP) – 0x0001 | – | – | – | – | – | – | INH | 8B | | 2 |
| PSHX | Push X (Index Register Low) onto Stack | Push (X); SP ← (SP) – 0x0001 | – | – | – | – | – | – | INH | 89 | | 2 |
| PULA | Pull Accumulator from Stack | SP ← (SP + 0x0001); Pull (A) | – | – | – | – | – | – | INH | 86 | | 3 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 9 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| PULH | Pull H (Index Register High) from Stack | SP ← (SP + 0x0001); Pull (H) | – | – | – | – | – | – | INH | 8A | | 3 |
| PULX | Pull X (Index Register Low) from Stack | SP ← (SP + 0x0001); Pull (X) | – | – | – | – | – | – | INH | 88 | | 3 |
| ROL *opr8a*<br>ROLA<br>ROLX<br>ROL *oprx8*,X<br>ROL ,X<br>ROL *oprx8*,SP | Rotate Left through Carry |  | Þ | – | – | Þ | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 39<br>49<br>59<br>69<br>79<br>9E69 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| ROR *opr8a*<br>RORA<br>RORX<br>ROR *oprx8*,X<br>ROR ,X<br>ROR *oprx8*,SP | Rotate Right through Carry |  | Þ | – | – | Þ | Þ | Þ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 36<br>46<br>56<br>66<br>76<br>9E66 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| RSP | Reset Stack Pointer | SP ← 0xFF<br>(High Byte Not Affected) | – | – | – | – | – | – | INH | 9C | | 1 |
| RTC | Return fom CALL | SP ← (SP) + 0x0001; Pull (PPAGE)<br>SP ← (SP) + 0x0001; Pull (PCH)<br>SP ← (SP) + 0x0001; Pull (PCL) | – | – | – | – | – | – | INH | 8D | | 7 |
| RTI | Return from Interrupt | SP ← (SP) + 0x0001; Pull (CCR)<br>SP ← (SP) + 0x0001; Pull (A)<br>SP ← (SP) + 0x0001; Pull (X)<br>SP ← (SP) + 0x0001; Pull (PCH)<br>SP ← (SP) + 0x0001; Pull (PCL) | Þ | Þ | Þ | Þ | Þ | Þ | INH | 80 | | 9 |
| RTS | Return from Subroutine | SP ← SP + 0x0001; Pull (PCH)<br>SP ← SP + 0x0001; Pull (PCL) | – | – | – | – | – | – | INH | 81 | | 6 |
| SBC *#opr8i*<br>SBC *opr8a*<br>SBC *opr16a*<br>SBC *oprx16*,X<br>SBC *oprx8*,X<br>SBC ,X<br>SBC *oprx16*,SP<br>SBC *oprx8*,SP | Subtract with Carry | A ← (A) – (M) – (C) | Þ | – | – | Þ | Þ | Þ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A2<br>B2<br>C2<br>D2<br>E2<br>F2<br>9ED2<br>9EE2 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| SEC | Set Carry Bit | C ← 1 | – | – | – | – | – | 1 | INH | 99 | | 1 |
| SEI | Set Interrupt Mask Bit | I ← 1 | – | – | 1 | – | – | – | INH | 9B | | 1 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 10 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| STA *opr8a*<br>STA *opr16a*<br>STA *oprx16*,X<br>STA *oprx8*,X<br>STA ,X<br>STA *oprx16*,SP<br>STA *oprx8*,SP | Store Accumulator in Memory | M ← (A) | 0 | – | – | Þ | Þ | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | B7<br>C7<br>D7<br>E7<br>F7<br>9ED7<br>9EE7 | dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 3<br>4<br>4<br>3<br>2<br>5<br>4 |
| STHX *opr8a*<br>STHX *opr16a*<br>STHX *oprx8*,SP | Store H:X (Index Reg.) | (M:M + 0x0001) ← (H:X) | 0 | – | – | Þ | Þ | – | DIR<br>EXT<br>SP1 | 35<br>96<br>9EFF | dd<br>hh ll<br>ff | 4<br>5<br>5 |
| STOP | Enable Interrupts: Stop Processing see MCU Documentation | I bit ← 0; Stop Processing | – | – | 0 | – | – | – | INH | 8E | | 2+ |
| STX *opr8a*<br>STX *opr16a*<br>STX *oprx16*,X<br>STX *oprx8*,X<br>STX ,X<br>STX *oprx16*,SP<br>STX *oprx8*,SP | Store X (Low 8 Bits of Index Register) in Memory | M ← (X) | 0 | – | – | Þ | Þ | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | BF<br>CF<br>DF<br>EF<br>FF<br>9EDF<br>9EEF | dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 3<br>4<br>4<br>3<br>2<br>5<br>4 |
| SUB *#opr8i*<br>SUB *opr8a*<br>SUB *opr16a*<br>SUB *oprx16*,X<br>SUB *oprx8*,X<br>SUB ,X<br>SUB *oprx16*,SP<br>SUB *oprx8*,SP | Subtract | A ← (A) – (M) | Þ | – | – | Þ | Þ | Þ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A0<br>B0<br>C0<br>D0<br>E0<br>F0<br>9ED0<br>9EE0 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| SWI | Software Interrupt | PC ← (PC) + 0x0001<br>Push (PCL); SP ← (SP) – 0x0001<br>Push (PCH); SP ← (SP) – 0x0001<br>Push (X); SP ← (SP) – 0x0001<br>Push (A); SP ← (SP) – 0x0001<br>Push (CCR); SP ← (SP) – 0x0001<br>I ← 1;<br>PCH ← Interrupt Vector High Byte<br>PCL ← Interrupt Vector Low Byte | – | – | 1 | – | – | – | INH | 83 | | 11 |

### Table 8-3. HCS08 Instruction Set Summary (Sheet 11 of 11)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| TAP | Transfer Accumulator to CCR | CCR ← (A) | Þ | Þ | Þ | Þ | Þ | Þ | INH | 84 | | 1 |
| TAX | Transfer Accumulator to X (Index Register Low) | X ← (A) | – | – | – | – | – | – | INH | 97 | | 1 |
| TPA | Transfer CCR to Accumulator | A ← (CCR) | – | – | – | – | – | – | INH | 85 | | 1 |
| TST *opr8a*<br>TSTA<br>TSTX<br>TST *oprx8*,X<br>TST ,X<br>TST *oprx8*,SP | Test for Negative or Zero | (M) – 0x00<br>(A) – 0x00<br>(X) – 0x00<br>(M) – 0x00<br>(M) – 0x00<br>(M) – 0x00 | 0 | – | – | Þ | Þ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D<br>4D<br>5D<br>6D<br>7D<br>9E6D | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| TSX | Transfer SP to Index Reg. | H:X ← (SP) + 0x0001 | – | – | – | – | – | – | INH | 95 | | 2 |
| TXA | Transfer X (Index Reg. Low) to Accumulator | A ← (X) | – | – | – | – | – | – | INH | 9F | | 1 |
| TXS | Transfer Index Reg. to SP | SP ← (H:X) – 0x0001 | – | – | – | – | – | – | INH | 94 | | 2 |
| WAIT | Enable Interrupts; Wait for Interrupt | I bit ← 0; Halt CPU | – | – | 0 | – | – | – | INH | 8F | | 2+ |

[1] Bus clock frequency is one-half of the CPU clock frequency.

## Table 8-4. Opcode Map (Sheet 1 of 3)

| Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 005<br>BRSET0<br>3DIR | 105<br>BSET0<br>2DIR | 203<br>BRA<br>2REL | 305<br>NEG<br>2DIR | 401<br>NEGA<br>1INH | 501<br>NEGX<br>1INH | 605<br>NEG<br>2IX1 | 704<br>NEG<br>1IX | 809<br>RTI<br>1INH | 903<br>BGE<br>2REL | A02<br>SUB<br>2IMM | B03<br>SUB<br>2DIR | C04<br>SUB<br>3EXT | D04<br>SUB<br>3IX2 | E03<br>SUB<br>2IX1 | F03<br>SUB<br>1X |
| 015<br>BRCLR0<br>3DIR | 115<br>BCLR0<br>2DIR | 213<br>BRN<br>2REL | 315<br>CBEQ<br>3DIR | 414<br>CBEQA<br>3IMM | 514<br>CBEQX<br>3IMM | 615<br>CBEQ<br>3IX1+ | 715<br>CBEQ<br>2IX+ | 816<br>RTS<br>1INH | 913<br>BLT<br>2REL | A12<br>CMP<br>2IMM | B13<br>CMP<br>2DIR | C14<br>CMP<br>3EXT | D14<br>CMP<br>3IX2 | E13<br>CMP<br>2IX1 | F13<br>CMP<br>1IX |
| 025<br>BRSET1<br>3DIR | 125<br>BSET1<br>2DIR | 223<br>BHI<br>2REL | 325<br>LDHX<br>3EXT | 425<br>MUL<br>1INH | 526<br>DIV<br>1INH | 621<br>NSA<br>1INH | 721<br>DAA<br>1INH | 825+<br>BGND<br>1INH | 923<br>BGT<br>2REL | A22<br>SBC<br>2IMM | B23<br>SBC<br>2DIR | C24<br>SBC<br>3EXT | D24<br>SBC<br>3IX2 | E23<br>SBC<br>2IX1 | F23<br>SBC<br>1IX |
| 035<br>BRCLR1<br>3DIR | 135<br>BCLR1<br>2DIR | 233<br>BLS<br>2REL | 335<br>COM<br>2DIR | 431<br>COMA<br>1INH | 531<br>COMX<br>1INH | 635<br>COM<br>2IX1 | 734<br>COM<br>1IX | 8311<br>SWI<br>1INH | 933<br>BLE<br>2REL | A32<br>CPX<br>2IMM | B33<br>CPX<br>2DIR | C34<br>CPX<br>3EXT | D34<br>CPX<br>3IX2 | E33<br>CPX<br>2IX1 | F33<br>CPX<br>1IX |
| 045<br>BRSET2<br>3DIR | 145<br>BSET2<br>2DIR | 243<br>BCC<br>2REL | 345<br>LSR<br>2DIR | 441<br>LSRA<br>1INH | 541<br>LSRX<br>1INH | 645<br>LSR<br>2IX1 | 744<br>LSR<br>1IX | 841<br>TAP<br>1INH | 942<br>TXS<br>1INH | A42<br>AND<br>2IMM | B43<br>AND<br>2DIR | C44<br>AND<br>3EXT | D44<br>AND<br>3IX2 | E43<br>AND<br>2IX1 | F43<br>AND<br>1IX |
| 055<br>BRCLR2<br>3DIR | 155<br>BCLR2<br>2DIR | 253<br>BCS<br>2REL | 354<br>STHX<br>2DIR | 453<br>LDHX<br>3IMM | 554<br>LDHX<br>2DIR | 653<br>CPHX<br>3IMM | 755<br>CPHX<br>2DIR | 851<br>TPA<br>1INH | 952<br>TSX<br>1INH | A52<br>BIT<br>2IMM | B53<br>BIT<br>2DIR | C54<br>BIT<br>3EXT | D54<br>BIT<br>3IX2 | E53<br>BIT<br>2IX1 | F53<br>BIT<br>1IX |
| 065<br>BRSET3<br>3DIR | 165<br>BSET3<br>2DIR | 263<br>BNE<br>2REL | 365<br>ROR<br>2DIR | 461<br>RORA<br>1INH | 561<br>RORX<br>1INH | 665<br>ROR<br>2IX1 | 764<br>ROR<br>1IX | 863<br>PULA<br>1INH | 965<br>STHX<br>3EXT | A62<br>LDA<br>2IMM | B63<br>LDA<br>2DIR | C64<br>LDA<br>3EXT | D64<br>LDA<br>3IX2 | E63<br>LDA<br>2IX1 | F63<br>LDA<br>1IX |
| 075<br>BRCLR3<br>3DIR | 175<br>BCLR3<br>2DIR | 273<br>BEQ<br>2REL | 375<br>ASR<br>2DIR | 471<br>ASRA<br>1INH | 571<br>ASRX<br>1INH | 675<br>ASR<br>2IX1 | 774<br>ASR<br>1IX | 872<br>PSHA<br>1INH | 971<br>TAX<br>1INH | A72<br>AIS<br>2IMM | B73<br>STA<br>2DIR | C74<br>STA<br>3EXT | D74<br>STA<br>3IX2 | E73<br>STA<br>2IX1 | F72<br>STA<br>1IX |
| 085<br>BRSET4<br>3DIR | 185<br>BSET4<br>2DIR | 283<br>BHCC<br>2REL | 385<br>LSL<br>2DIR | 481<br>LSLA<br>1INH | 581<br>LSLX<br>1INH | 685<br>LSL<br>2IX1 | 784<br>LSL<br>1IX | 883<br>PULX<br>1INH | 981<br>CLC<br>1INH | A82<br>EOR<br>2IMM | B83<br>EOR<br>2DIR | C84<br>EOR<br>3EXT | D84<br>EOR<br>3IX2 | E83<br>EOR<br>2IX1 | F83<br>EOR<br>1IX |
| 095<br>BRCLR4<br>3DIR | 195<br>BCLR4<br>2DIR | 293<br>BHCS<br>2REL | 395<br>ROL<br>2DIR | 491<br>ROLA<br>1INH | 591<br>ROLX<br>1INH | 695<br>ROL<br>2IX1 | 794<br>ROL<br>1IX | 892<br>PSHX<br>1INH | 991<br>SEC<br>1INH | A92<br>ADC<br>2IMM | B93<br>ADC<br>2DIR | C94<br>ADC<br>3EXT | D94<br>ADC<br>3IX2 | E93<br>ADC<br>2IX1 | F93<br>ADC<br>1IX |
| 0A5<br>BRSET5<br>3DIR | 1A5<br>BSET5<br>2DIR | 2A3<br>BPL<br>2REL | 3A5<br>DEC<br>2DIR | 4A1<br>DECA<br>1INH | 5A1<br>DECX<br>1INH | 6A5<br>DEC<br>2IX1 | 7A4<br>DEC<br>1IX | 8A3<br>PULH<br>1INH | 9A1<br>CLI<br>1INH | AA2<br>ORA<br>2IMM | BA3<br>ORA<br>2DIR | CA4<br>ORA<br>3EXT | DA4<br>ORA<br>3IX2 | EA3<br>ORA<br>2IX1 | FA3<br>ORA<br>1IX |
| 0B5<br>BRCLR5<br>3DIR | 1B5<br>BCLR5<br>2DIR | 2B3<br>BMI<br>2REL | 3B7<br>DBNZ<br>3DIR | 4B4<br>DBNZA<br>2INH | 5B4<br>DBNZX<br>2INH | 6B7<br>DBNZ<br>3IX1 | 7B6<br>DBNZ<br>2IX | 8B2<br>PSHH<br>1INH | 9B1<br>SEI<br>1INH | AB2<br>ADD<br>2IMM | BB3<br>ADD<br>2DIR | CB4<br>ADD<br>3EXT | DB4<br>ADD<br>3IX2 | EB3<br>ADD<br>2IX1 | FB3<br>ADD<br>1IX |
| 0C5<br>BRSET6<br>3DIR | 1C5<br>BSET6<br>2DIR | 2C3<br>BMC<br>2REL | 3C5<br>INC<br>2DIR | 4C1<br>INCA<br>1INH | 5C1<br>INCX<br>1INH | 6C5<br>INC<br>2IX1 | 7C4<br>INC<br>1IX | 8C1<br>CLRH<br>1INH | 9C1<br>RSP<br>1INH | AC8<br>CALL<br>4EXT | BC3<br>JMP<br>2DIR | CC4<br>JMP<br>3EXT | DC4<br>JMP<br>3IX2 | EC3<br>JMP<br>2IX1 | FC3<br>JMP<br>1IX |
| 0D5<br>BRCLR6<br>3DIR | 1D5<br>BCLR6<br>2DIR | 2D3<br>BMS<br>2REL | 3D4<br>TST<br>2DIR | 4D1<br>TSTA<br>1INH | 5D1<br>TSTX<br>1INH | 6D4<br>TST<br>2IX1 | 7D3<br>TST<br>1IX | 8D7<br>RTC<br>1INH | 9D1<br>NOP<br>1INH | AD5<br>BSR<br>2REL | BD5<br>JSR<br>2DIR | CD6<br>JSR<br>3EXT | DD6<br>JSR<br>3IX2 | ED5<br>JSR<br>2IX1 | FD5<br>JSR<br>1IX |

## Table 8-4. Opcode Map (Sheet 2 of 3)

| Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0E5<br>BRSET7<br>3DIR | 1E5<br>BSET7<br>2DIR | 2E3<br>BIL<br>2REL | 3E6<br>CPHX<br>3EXT | 4E5<br>MOV<br>3DD | 5E5<br>MOV<br>2DIX+ | 6E4<br>MOV<br>3IMD | 7E5<br>MOV<br>2IX+D | 8E2+<br>STOP<br>1INH | 9E<br>Page<br>2 | AE2<br>LDX<br>2IMM | BE3<br>LDX<br>2DIR | CE4<br>LDX<br>3EXT | DE4<br>LDX<br>3IX2 | EE3<br>LDX<br>2IX1 | FE3<br>LDX<br>1IX |
| 0F5<br>BRCLR7<br>3DIR | 1F5<br>BCLR7<br>2DIR | 2F3<br>BIH<br>2REL | 3F5<br>CLR<br>2DIR | 4F1<br>CLRA<br>1INH | 5F1<br>CLRX<br>1INH | 6F5<br>CLR<br>2IX1 | 7F4<br>CLR<br>1IX | 8F2+<br>WAIT<br>1INH | 9F1<br>TXA<br>1INH | AF2<br>AIX<br>2IMM | BF3<br>STX<br>2DIR | CF4<br>STX<br>3EXT | DF4<br>STX<br>3IX2 | EF3<br>STX<br>2IX1 | FF2<br>STX<br>1IX |
| | | | | | | | | | | | | | | | |
| INHInherentRELRelativeSP1Stack Pointer, 8-Bit Offset<br>IMMImmediateIXIndexed, No OffsetSP2Stack Pointer, 16-Bit Offset<br>DIRDirectIX1Indexed, 8-Bit OffsetIX+Indexed, No Offset with<br>EXTExtendedIX2Indexed, 16-Bit OffsetPost Increment<br>DDDIR to DIRIMDIMM to DIRIX1+Indexed, 1-Byte Offset with<br>IX+DIX+ to DIR DIX+DIR to IX+Post Increment | | | | | | | | | Opcode in<br>Hexadecimal<br><br>Number of Bytes | | | | | | |
| | | | | | | | | | | | | F03<br>SUB<br>1IX | HCS08 Cycles<br>Instruction Mnemonic<br>Addressing Mode | | |
| | | | | | | 9E606<br>NEG<br>3SP1 | | | | | | | 9ED05<br>SUB<br>4SP2 | 9EE04<br>SUB<br>3SP1 | |
| | | | | | | 9E616<br>CBEQ<br>4SP1 | | | | | | | 9ED15<br>CMP<br>4SP2 | 9EE14<br>CMP<br>3SP1 | |
| | | | | | | | | | | | | | 9ED25<br>SBC<br>4SP2 | 9EE24<br>SBC<br>3SP1 | |
| | | | | | | 9E636<br>COM<br>3SP1 | | | | | | | 9ED35<br>CPX<br>4SP2 | 9EE34<br>CPX<br>3SP1 | 9EF36<br>CPHX<br>3SP1 |
| | | | | | | 9E646<br>LSR<br>3SP1 | | | | | | | 9ED45<br>AND<br>4SP2 | 9EE44<br>AND<br>3SP1 | |
| | | | | | | | | | | | | | 9ED55<br>BIT<br>4SP2 | 9EE54<br>BIT<br>3SP1 | |
| | | | | | | 9E666<br>ROR<br>3SP1 | | | | | | | 9ED65<br>LDA<br>4SP2 | 9EE64<br>LDA<br>3SP1 | |
| | | | | | | 9E676<br>ASR<br>3SP1 | | | | | | | 9ED75<br>STA<br>4SP2 | 9EE74<br>STA<br>3SP1 | |
| | | | | | | 9E686<br>LSL<br>3SP1 | | | | | | | 9ED85<br>EOR<br>4SP2 | 9EE84<br>EOR<br>3SP1 | |

### Table 8-4. Opcode Map (Sheet 3 of 3)

| Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 9E696<br>ROL<br>3 SP1 | | | | | | | 9ED9<br>5<br>ADC<br>4<br>SP2 | 9EE9<br>4<br>ADC<br>3<br>SP1 | |
| | | | | | | 9E6A6<br>DEC<br>3 SP1 | | | | | | | 9EDA<br>5<br>ORA<br>4<br>SP2 | 9EEA<br>4<br>ORA<br>3<br>SP1 | |
| | | | | | | 9E6B8<br>DBNZ<br>4 SP1 | | | | | | | 9EDB<br>5<br>ADD<br>4<br>SP2 | 9EEB<br>4<br>ADD<br>3<br>SP1 | |
| | | | | | | 9E6C6<br>INC<br>3 SP1 | | | | | | | | | |
| | | | | | | 9E6D5<br>TST<br>3 SP1 | | | | | | | | | |
| | | | | | | | | | | 9EAE<br>5<br>LDHX<br>2 IX | 9EBE<br>6<br>LDHX<br>4 IX2 | 9ECE<br>5<br>LDHX<br>3 IX1 | 9EDE<br>5<br>LDX<br>4<br>SP2 | 9EEE<br>4<br>LDX<br>3<br>SP1 | 9EFE<br>5<br>LDHX<br>3<br>SP1 |
| | | | | | | 9E6F6<br>CLR<br>3 SP1 | | | | | | | 9EDF<br>5<br>STX<br>4<br>SP2 | 9EEF<br>4<br>STX<br>3<br>SP1 | 9EFF<br>5<br>STHX<br>3<br>SP1 |

INHInherentRELRelativeSP1Stack Pointer, 8-Bit Offset
IMMImmediateIXIndexed, No OffsetSP2Stack Pointer, 16-Bit Offset
DIRDirectIX1Indexed, 8-Bit OffsetIX+Indexed, No Offset with
EXTExtendedIX2Indexed, 16-Bit OffsetPost Increment
DDDIR to DIRIMDIMM to DIRIX1+Indexed, 1-Byte Offset with
IX+DIX+ to DIRDIX+DIR to IX+Post Increment

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

Prebyte (9E) and Opcode in Hexadecimal

Number of Bytes

| | |
|---|---|
| 9E606<br>NEG<br>3<br>SP1 | HCS08 Cycles<br>Instruction Mnemonic<br>Addressing Mode |

# Chapter 9
# Parallel Input and Output

## 9.1 Introduction

This section explains software controls related to parallel input/output (I/O) and pin control. The MC13237 has 3 parallel 8-bit and 1 parallel 4-bit I/O ports, which include a total of 28 I/O pins. The MC13234 has four parallel 8-bit I/O ports, which include a total of 32 I/O pins. See Chapter 2, “Pins and Connections” for more information about pin assignments.

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 are disabled.

After reset, the shared peripheral functions are disabled and the pins default to:

- Inputs
- Slew rate control enabled
- Low drive strength selected
- Internal pullups disabled.

**NOTE**

- To avoid extra current drain from floating input pins, the user should initialize any unused/unconnected GPIO as either the default inputs with on-chip pullups enabled or to outputs set to low. See Chapter 3, “System Considerations” for more details.
- I/O signal PTA2 is a special pin that can enable a special reserved factory test mode. See Section 3.3.2, “Signal PTA2 (Factory Test Mode Enable)” for more information.

## 9.2 Features

Parallel I/O features include:

- A total of 28/32 general-purpose I/O pins in four ports
- Hysteresis input buffers
- Software-controlled pullups on each input pin
- Software-controlled slew rate output buffers
- Port A pins shared with IIC Bus and 32.768 kHz oscillator
- PTA2 enables a reserved factory test mode
- PTA7 is shared as serial debug port pin

- Eight Port B pins shared with KBI1
- MC13234: Four Port C pins shared with KBI2
- MC13234: Four Port C pins shared with SPI port
- MC13234: Eight Port D pins share with four TPM I/O pins, CMT high current output, and two SCI pins
- MC13237: Four Port C pins shared with SS_B, SPICLK/AD7, MISO, and MOSI
- MC13237: Eight Port D pins shared with four TPM I/O pins, CMT high current output, two SCI pins
- GPIO have no interrupt request capability other than that provided by the shared functions

## 9.3 IO Port Pin Functional Description

The following description provides an overview of an individual port pin. Figure 9-1 shows a simple block diagram of a port pin.

Most pins reset to the conditions stated in Section 9.1, “Introduction”, however, there are the following exceptions:

- PTA2 (factory test enable) — This signal should not be forced low while VDD is ramping-up.
  - As VDD ramps, the POR circuit holds a reset condition until the voltage exceeds the POR threshold. This pin is an input and must be in the low state when the POR exits reset or a factory chip test mode is enabled.
  - PTA2 is otherwise a standard GPIO port pin, however, best practice is to tie the pin low through a resistor to assure a low state when exiting POR
  - A reset initiated by the RESET pin or any operational condition other than POR does not affect PTA2 or factory test.
- PTA7/BKGD/MS (shared debug port) — upon exiting reset, BKGD/MS is enabled and therefore is not usable as an output pin until BKGDPE in SOPT Register is cleared.

Each pin has the programmable attributes of input mode versus output mode:

- Input mode with optional pullup
- Output mode
  - Slew rate control
  - Drive strength control
  - Output state - as determined by output register

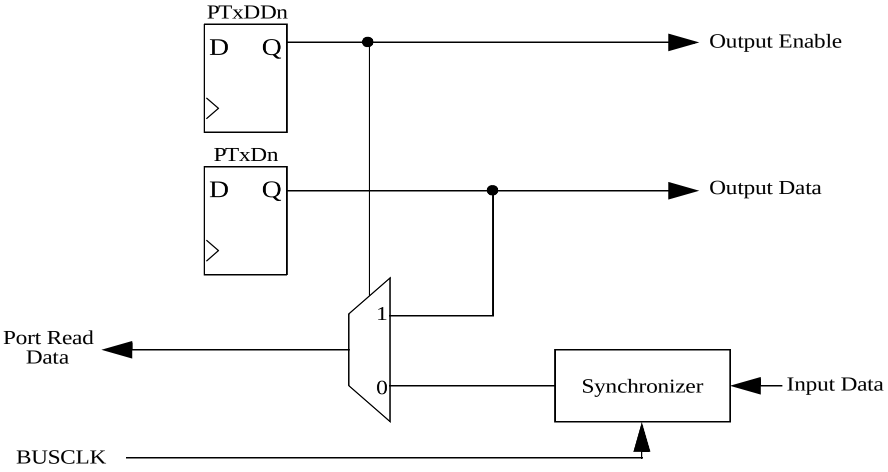


Figure 9-1. Parallel Port Pin Block Diagram

### 9.3.1 Port Data and Data Direction

Reading and writing of the parallel I/O are performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers.

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 or is an output-only pin.

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

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

## 9.4 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.

### 9.4.1 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 is also disabled for an analog function.

### 9.4.2 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 to reduce EMC emissions. Slew rate control has no effect on pins that are configured as inputs.

### 9.4.3 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 is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the MCU are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load.

## 9.5 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the stop mode that is entered. An explanation of pin behavior for the Stop3 mode follows:

- In Stop3 mode, all I/O is maintained because internal logic circuity stays powered up. Upon recovery, normal I/O function is available to the user.

Pins behavior is unaffected by LPRun, Wait, and Background mode operation.

## 9.6 Interrupt Request Capability

The parallel IO ports have no interrupt request generation capability.

## 9.7 Parallel I/O and Pin Control Registers

Each port has an independent set of control registers. These registers are located both in the Direct-Page and High-Page Register Maps:

- The data and data direction registers are located in the Direct-Page Map.
- The pullup, slew rate, and drive strength registers are located in the High-Page Map.

### 9.7.1 Port A Registers

Port A is controlled by the registers listed below.

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

PTAD is the Port A data register. The register contents drive the Port A pads when they are configured as outputs. A read of this register returns the register contents for a programmed output and returns the state of the pad for a programmed input.

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

**NOTE**

PTA2 is a special pin. If the VDD supply voltage is applied (or recycled), the POR circuit is activated and tests the PTA2 input state as the POR releases. If PTA2 is at a high state the device enters a factory test mode, and if PTA2 is at a low state, normal operation ensues.

| PTAD | | | | | 0x0000 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTAD7 | PTAD6 | PTAD5 | PTAD4 | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-2. Port A Data Register (PTAD)**

**Table 9-1. PTAD Register Field Descriptions**

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

### 9.7.1.2 Port A Data Direction Register (PTADD)

PTADD is the read/write Port A data direction register. The register controls the direction of the Port A pins and also controls the read function of PTAD.

| PTADD | | | | | 0x0001 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTADD7 | PTADD6 | PTADD5 | PTADD4 | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-3. Port A Data Direction Register (PTADD)**

Table 9-2. PTADD Register Field Descriptions

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

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

PTAPE is the read/write Port A pullup enable register. The register enables onboard pullups for pins that are enabled as inputs. These bits have no effect on Port A pins programmed as outputs.

When pins are controlled by peripheral functions, this bits can also enable a pullup on input pins. The pullups are disabled for an analog function such as the 32 kHz oscillator.

**PTAPE** **0x1830**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAPE7 | PTAPE6 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 9-4. Port A Pull Enable Register (PTAPE)

Table 9-3. PTAPE Register Field Descriptions

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

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

PTASE is the read/write Port A output slew rate enable register. The register controls the slew rate of Port A outputs.

**PTASE** **0x1831**

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

Figure 9-5. Port A Slew Rate Enable Register (PTASE)

Table 9-4. PTASE Register Field Descriptions

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

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

PTADS is the read/write Port A output drive strength control register. The register controls the drive strength of Port A outputs.

**PTADS** **0x1832**

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

Figure 9-6. Port A Drive Strength Selection Register (PTADS)

Table 9-5. PTADS Register Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>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 that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for Port A bit n.<br>1 High output drive strength selected for Port A bit n. |

## 9.7.2 Port B Registers

Port B is controlled by the registers listed below.

### 9.7.2.1 Port B Data Register (PTBD)

PTBD is the Port Bdata register. The register contents drive the PORT Bpads when they are configured as outputs. A read of this register returns the register contents for a programmed output and returns the state of the pad for a programmed input.

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

| PTBD | | | | | 0x0002 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTBD7 | PTBD6 | PTBD5 | PTBD4 | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-7. Port B Data Register (PTBD)**

**Table 9-6. PTBD Register Field Descriptions**

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

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

PTBDD is the read/write Port B data direction register. The register controls the direction of the Port B pins and also controls the read function of PTBD.

| PTBDD | | | | | 0x0003 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTBDD7 | PTBDD6 | PTBDD5 | PTBDD4 | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-8. Port B Data Direction Register (PTBDD)**

**Table 9-7. PTBDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBDD[7:0] | **Data Direction for Port B Bits** — These read/write bits control the direction of Port B pins and what is read for PTBD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for Port B bit n and PTBD reads return the contents of PTBDn. |

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

PTBPE is the read/write Port B pullup enable register. The register enables onboard pullups for pins that are enabled as inputs. These bits have no effect on Port B pins programmed as outputs.

When pins are controlled by peripheral functions, this bits can also enable a pullup on input pins.

**PTBPE** **0x1834**

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

**Figure 9-9. Port B Pull Enable Register (PTBPE)**

**Table 9-8. PTBPE Register Field Descriptions**

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

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

PTBSE is the read/write Port B output slew rate enable register. The register controls the slew rate of Port B outputs.

**PTBSE** **0x1835**

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

**Figure 9-10. Port B Slew Rate Enable Register (PTBSE)**

**Table 9-9. PTBSE Register Field Descriptions**

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

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

PTBDS is the read/write Port B output drive strength control register. The register controls the drive strength of Port B outputs.

| | PTBDS | | | | 0x1836 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTBDS7 | PTBDS6 | PTBDS5 | PTBDS4 | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-11. Port B Drive Strength Selection Register (PTBDS)**

**Table 9-10. PTBDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>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 that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for Port B bit n.<br>1 High output drive strength selected for Port B bit n. |

## 9.7.3 Port C Registers

Port C is controlled by the registers listed below.

### 9.7.3.1 Port C Data Register (PTCD)

PTCD is the Port C data register. The register contents drive the PORT C pads when they are configured as outputs. A read of this register returns the register contents for a programmed output and returns the state of the pad for a programmed input.

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

| | PTCD | | | | 0x0004 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTCD7 | PTCD6 | PTCD5 | PTCD4 | PTCD3 | PTCD2 | PTCD1 | PTCD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-12. Port C Data Register (PTCD)**

**Table 9-11. PTCD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCD[7:0] | **Port C Data Register Bits** — For Port C pins:<br>• For pins configured as inputs, reads return the logic level on the pin.<br>• For pins that are configured as outputs, reads return the last value written to this register.<br>• Writes are latched into all bits of this register. For port A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.<br>• Reset forces PTCD to all 0s, but these 0s are not driven out to the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.<br>• In the MC13237, only PTC4, PTC5, PTC6, and PTC7 are available. |

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

PTCDD is the read/write Port A data direction register. The register controls the direction of the Port C pins and also controls the read function of PTCD.

| | PTCDD | | | | 0x0005 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTCDD7 | PTCDD6 | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-13. Port C Data Direction Register (PTCDD)**

**Table 9-12. PTCDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCDD[7:0] | **Data Direction for Port C Bits** — These read/write bits control the direction of Port C pins and what is read for PTCD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for Port C bit n and PTCD reads return the contents of PTCDn.<br>In the MC13237, only PTC4, PTC5, PTC6, and PTC7 are available. |

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

PTAPE is the read/write Port A pullup enable register. The register enables onboard pullups for pins that are enabled as inputs. These bits have no effect on Port A pins programmed as outputs.

When pins are controlled by peripheral functions, this bits can also enable a pullup on input pins.

| | PTCPE | | | | 0x1838 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTCPE7 | PTCPE6 | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-14. Port C Pull Enable Register (PTCPE)**

**Table 9-13. PTCPE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCPE[7:0] | **Internal Pull Enable for Port C Bits** — Each of these control bits determines whether the internal pullup device is enabled for the associated PTC pin. For Port C pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled.<br>0 Internal pullup device disabled for Port C bit n.<br>1 Internal pullup device enabled for Port C bit n.<br>In the MC13237, only PTC4, PTC5, PTC6, and PTC7 are available. |

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

PTCSE is the read/write Port C output slew rate enable register. The register controls the slew rate of Port C outputs.

| | PTCSE | | | | 0x1839 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTCSE7 | PTCSE6 | PTCSE5 | PTCSE4 | PTCSE3 | PTCSE2 | PTCSE1 | PTCSE0 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 9-15. Port C Slew Rate Enable Register (PTCSE)**

**Table 9-14. PTCSE Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCSE[7:0] | **Output Slew Rate Enable for Port C Bits** — Each of these control bits determines whether the output slew rate control is enabled for the associated PTC pin. For Port C pins that are configured as inputs, these bits have no effect.<br>0 Output slew rate control disabled for Port C bit n.<br>1 Output slew rate control enabled for Port C bit n.<br>In the MC13237, only PTC4, PTC5, PTC6, and PTC7 are available. |

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

PTCDS is the read/write Port C output drive strength control register. The register controls the drive strength of Port C outputs.

| | PTCDS | | | | 0x183A | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTCDS7 | PTCDS6 | PTCDS5 | PTCDS4 | PTCDS3 | PTCDS2 | PTCDS1 | PTCDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-16. Port C Drive Strength Selection Register (PTCDS)**

**Table 9-15. PTCDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>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 that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for Port C bit n.<br>1 High output drive strength selected for Port C bit n.<br>In the MC13237, only PTC4, PTC5, PTC6, and PTC7 are available. |

## 9.7.4 Port D Registers

Port D is controlled by the registers listed below.

### 9.7.4.1 Port D Data Register (PTDD)

PTDD is the Port D data register. The register contents drive the PORT D pads when they are configured as outputs. A read of this register returns the register contents for a programmed output and returns the state of the pad for a programmed input.

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

| | PTDD | | | | 0x0006 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTDD7 | PTDD6 | PTDD5 | PTDD4 | PTDD3 | PTDD2 | PTDD1 | PTDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-17. Port D Data Register (PTDD)**

**Table 9-16. PTDD Register Field Descriptions**

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

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

PTDDD is the read/write Port D data direction register. The register controls the direction of the Port D pins and also controls the read function of PTDD.

| PTDDD | | | | | 0x0007 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTDDD7 | PTDDD6 | PTDDD5 | PTDDD4 | PTDDD3 | PTDDD2 | PTDDD1 | PTDDD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-18. Port D Data Direction Register (PTDDD)**

**Table 9-17. PTDDD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>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.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for Port D bit n and PTDD reads return the contents of PTDDn. |

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

PTDPE is the read/write Port D pullup enable register. The register enables onboard pullups for pins that are enabled as inputs. These bits have no effect on Port D pins programmed as outputs.

When pins are controlled by peripheral functions, this bits can also enable a pullup on input pins.

| PTDPE | | | | | 0x183C | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTDPE7 | PTDPE6 | PTDPE5 | PTDPE4 | PTDPE3 | PTDPE2 | PTDPE1 | PTDPE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-19. Port D Pull Enable Register (PTDPE)**

**Table 9-18. PTDPE Register Field Descriptions**

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

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

PTDSE is the read/write Port D output slew rate enable register. The register controls the slew rate of Port D outputs.

| | PTDSE | | | | 0x183D | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTDSE7 | PTDSE6 | PTDSE5 | PTDSE4 | PTDSE3 | PTDSE2 | PTDSE1 | PTDSE0 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 9-20. Port D Slew Rate Enable Register (PTDSE)**

**Table 9-19. PTDSE Register Field Descriptions**

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

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

PTDDS is the read/write Port D output drive strength control register. The register controls the drive strength of Port D outputs.

| | PTDDS | | | | 0x183E | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PTDDS7 | PTDDS6 | PTDDS5 | PTDDS4 | PTDDS3 | PTDDS2 | PTDDS1 | PTDDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-21. Port D Drive Strength Selection Register (PTDDS)**

Table 9-20. PTDDS Register Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>PTDDS[7:0] | **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 that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for Port D bit n.<br>1 High output drive strength selected for Port D bit n. |

# Chapter 10
# Real-Time Counter (RTC)

## 10.1 Introduction

The real-time counter (RTC) consists of a 16-bit counter, a 16-bit comparator, several binary-based and decimal-based prescaler dividers, three clock sources, and a programmable periodic interrupt request. This module can be used for time-of-day, calendar or task scheduling functions. It can also serve as a cyclic wake-up from low power modes without the need of external components.

RTC can be clocked from the 32 MHz reference clock, the optional 32.768 kHz clock or the on board 1 kHz RC clock.

## 10.2 Features

Features of the RTC module include:

- 16-bit up-counter
  - 16-bit modulo match limit
  - Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler
  - Optional 32.768 kHz oscillator (XOSC1).
  - 32MHz reference oscillator (XOSC2).
  - 1 kHz RC oscillator (default)
- Wide range of prescale options include binary and decimal divide-by ratios
- For lowest operating power, the RTC can be disabled via system clock gating control (Register SCGC2, Bit 1)

## 10.3 Block Diagram

The block diagram for the RTC module is shown in Figure 10-1.

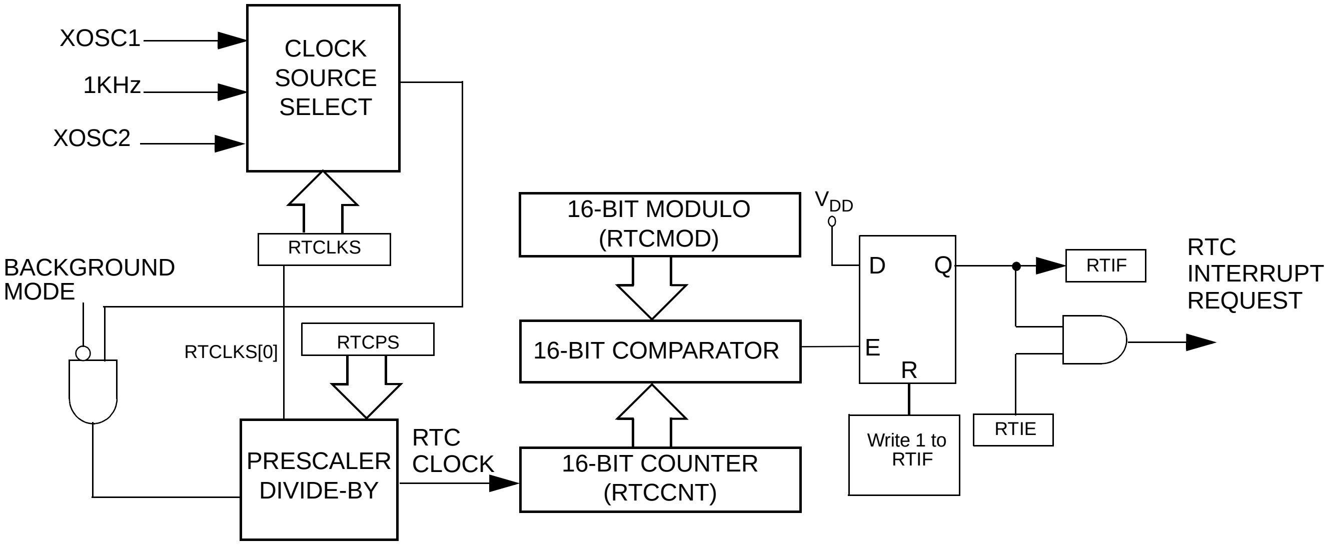


Figure 10-1. Real-Time Counter (RTC) Block Diagram

## 10.4 Functional Description

The RTC is built around a main 16-bit up-counter (RTCCNT) driven by the internal RTC Clock that is derived from a clock select multiplexer that, in turn, feeds a programmable prescaler with binary and decimal divide-by options. The up-counter value is then compared to a 16-bit modulo register (RTCMOD) that provides status (RTIF) and resets the up-counter upon a valid compare. The RTIF can be enabled to provide an interrupt request.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1kHz internal oscillator clock is selected as the default clock source. To start the prescaler, any value other than zero is written to the prescaler select bits (RTCPS).

### 10.4.1 Clock Source and Prescale Value

Three clock sources are software selectable:

- The optional 32.768 kHz oscillator (XOSC1)
- 1 kHz RC oscillator (default)
- 32 MHz reference oscillator (XOSC2)

The RTC clock select bits (RTCLKS[1:0]) are used to select the desired clock source to the prescaler dividers. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

**NOTE**

The selected RTC clock source impacts operation of the 1 kHz oscillator and power down of the RTC:

- The RTC must be enabled by the Register SCGC2, Bit 1 to switch the RTC clock source to one other than the 1 kHz oscillator
- The 1 kHz oscillator must not be the selected RTC clock source to allow the 1 kHz oscillator to be disabled. See Section 5.4.1, “Clock Sources”.
- The 1 kHz oscillator CANNOT be the selected RTC clock source if it is desired to disable the RTC block via Register SCGC2, Bit 1. See Section 10.5.2, “Stop3 Mode”

The RTC prescaler select bits (RTCPS[3:0]) in combination with the RTCLKS[0] bit select the desired prescaler divide-by value. If a new value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. Table 10-1 shows available prescaler period values.

**Table 10-1. RTC Prescaler Period Values**

| RTCPS[3:0] | 1 kHz RC Osc Source Prescale Ratio / Prescaler Period (RTCLKS = 00) | 32-MHz Osc Source Prescale Ratio / Prescaler Period (RTCLKS = 01) | 32.768-kHz Osc Source Prescale Ratio / Prescaler Period (RTCLKS = 10) | 32.768-kHz Osc Source Prescale Ratio / Prescaler Period (RTCLKS = 11) |
|---|---|---|---|---|
| 0000 | Off | Off | Off | Off |
| 0001 | $2^{3}$ / 8 ms | $2^{10}$ / 0.032 ms | $2^{3}$ / 0.244 ms | $2^{10}$ / 31.25 ms |
| 0010 | $2^{5}$ / 32 ms | $2^{11}$ / 0.064 ms | $2^{5}$ / 0.976 ms | $2^{11}$ / 62.5 ms |
| 0011 | $2^{6}$ / 64 ms | $2^{12}$ / 0.128 ms | $2^{6}$ / 1.95 ms | $2^{12}$ / 125 ms |
| 0100 | $2^{7}$ / 128 ms | $2^{13}$ / 0.256 ms | $2^{7}$ / 3.9 ms | $2^{13}$ / 250 ms |
| 0101 | $2^{8}$ / 256 ms | $2^{14}$ / 0.512 ms | $2^{8}$ / 7.8125 ms | $2^{14}$ / 500 ms |
| 0110 | $2^{9}$ / 512 ms | $2^{15}$ / 1.024 ms | $2^{9}$ / 15.625 ms | $2^{15}$ / 1 S |
| 0111 | $2^{10}$ / 1024 ms | $2^{16}$ / 2.048 ms | $2^{10}$ / 31.25 ms | $2^{16}$ / 2 S |
| 1000 | 1 / 1 ms | $10^{3}$ / 0.03125 ms | 1 / 0.030517 ms | $10^{3}$ / 30.518 ms |
| 1001 | 2 / 2 ms | 2x$10^{3}$ / 0.0625 ms | 2 / 0.06103 ms | 2x$10^{3}$ / 61.035 ms |
| 1010 | $2^{2}$ / 4 ms | 5x$10^{3}$ / 0.15625 ms | $2^{2}$ / 0.12207 ms | 5x$10^{3}$ / 152.59 ms |
| 1011 | 10 / 10 ms | $10^{4}$ / 0.3125 ms | 10 / 0.30517 ms | $10^{4}$ / 305.18 ms |
| 1100 | $2^{4}$ / 16 ms | 2x$10^{4}$ / 0.625 ms | $2^{4}$ / 0.48828 ms | 2x$10^{4}$ / 610.35 ms |
| 1101 | $10^{2}$ / 100 ms | 5x$10^{4}$ / 1.5625 ms | $10^{2}$ / 3.0517 ms | 5x$10^{4}$ / 1.5259 s |
| 1110 | 5x$10^{2}$ / 500 ms | $10^{5}$ / 3.125 ms | 5x$10^{2}$ / 15.2588 ms | $10^{5}$ / 3.0518 s |
| 1111 | $10^{3}$ / 1 s | 2x$10^{5}$ / 6.25 ms | $10^{3}$ / 30.517 ms | 2x$10^{5}$ / 6.1035 s |

## 10.4.2 Compare Function

The compare function is used to set a time base for generating a periodic interrupt that can be used for walk-up or as a timetick. The time period is determined by:

- The frequency of the selected clock source
- The prescaler setting

- The RTC modulo register

#### 10.4.2.1 RTC Modulo Register

The RTC modulo register (RTCMOD) provides the compare value which can be set to any value from 0x0000 to 0xFFFF. When the counter is active, the counter increments at the selected prescale rate until the count matches the modulo value. When the values match, on the next clock the counter resets to 0x0000 and then continues counting. The real-time interrupt flag (RTIF) is set whenever a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counter to 0x0000.

#### 10.4.2.2 RTC Accuracy with 1 kHz RC Oscillator

The RTC time period is only as accurate as the selected clock source. The 32.768 kHz and 32 MHz sources are crystal-based and as such are very accurate. The 1 kHz RC oscillator frequency is typically accurate to within ± 40% at room temperate and can vary even more with temperature.

It is possible to do a simple form of trimming for a 1 kHz oscillator based RTC application:

- Select an RTC prescale option that allows for the largest possible modulo register setting for the desired timebase
- In software, provide a utility that creates a TPM timer function with a delay long enough to loosely calibrate the RTC count.
  - Use an interrupt from the TPM to read the RTC Counter Registers.
  - This counter value can then be used to calculate a difference from an expected value.
  - The difference can then be used to modify the Modulo register value to trim the RTC counter period closer to an accurate value.

### 10.4.3 Interrupt Request

The RTC allows for an interrupt request to be generated whenever RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

### 10.4.4 Module Clock Disable

The RTC compare logic and registers are driven by the bus clock. To disable the module for lowest operating power, the module clock can be disabled by the RTC Bit of Register SCGC2.

## 10.5 Modes of Operation

The operational mode of the RTC is affected by the present mode of the MC13234/MC13237. The RTC is unaffected in Run or LPRun modes.

### 10.5.1 Wait Mode

The RTC continues to run in Wait mode if it was enabled before executing the WAIT instruction. Therefore, the RTC can be used to bring the MCU out of Wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during Wait mode.

### 10.5.2 Stop3 Mode

The RTC continues to run in Stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can be used to bring the MCU out of stop modes, if the real-time interrupt is enabled.

- For lowest power, the RTC block can be disabled before entering Stop mode -
  - The RTC clock source must first be programmed as the 32 MHz or the 32.768 kHz (not using the 1 kHz source) before disabling the RTC block. The RTC clock must be enabled to allow this operation.
  - The RTC block is then disabled by writing RTC Clock Date Control (Bit 1, Register SCGC2) to a zero.
- Power consumption is lower when the RTC is disabled, but in that case the real-time interrupt cannot wake up the MCU from Stop modes.

### 10.5.3 Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

## 10.6 External Signal Description

The only external signals which affect the RTC are the 32.768 kHz oscillator crystal pins XTAL_32K and EXTAL_32K when the oscillator is in use.

## 10.7 Register Definitions

The RTC includes a status and control register, an 16-bit counter register, and an 16-bit modulo register. The register addresses are located in the High-Page Register map, and Table 10-2 lists address versus register mnemonic.

**Table 10-2. RTC Register Address Map**

| Register | Address |
|---|---|
| RTCCSC | 0x1828 |
| RTCCNTL | 0x1829 |

**Table 10-2. RTC Register Address Map (continued)**

| Register | Address |
|---|---|
| RTCMODL | 0x182A |
| RTCCNTH | 0x182B |
| RTCMODH | 0x182C |

**NOTE**

The RTC register map for the MC13234/MC13237 is an anomaly to 68HC08 standard address mapping. Normally for two-byte wide registers such as the RTCCNT and RTCMOD, the most significant byte of the variable is located at the lower address of two adjacent locations and the least significant byte is at the higher address. Both the RTCCNT and RTCMOD violate this model as observed in Table 10-2.

### 10.7.1 RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).

RTCSC 0x1828

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | RTIF | RTCLKS | | RTIE | RTCPS | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-2. RTC Status and Control Register (RTCSC)**

**Table 10-3. RTCSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIF | **Real-Time Interrupt Flag**— This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request.<br>0 RTC counter has not reached the value in the RTC modulo register<br>1 RTC counter has reached the value in the RTC modulo register. |
| 6:5<br>RTCLKS[1:0] | **Real-Time Clock Source Select** — These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters.<br>00 Real-time clock source is 1kHz<br>01 Real-time clock source is XOSC2 (32.MHz)<br>10 Real-time clock source is XOSC1 (32.768kHz)<br>11 Real-time clock source is XOSC1 (32.768kHz) |

Table 10-3. RTCSC Field Descriptions (continued)

| Field | Description |
|---|---|
| 4<br>RTIE | **Real-Time Interrupt Enable** — This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set.<br>0 Real-time interrupt requests are disabled.<br>1 Real-time interrupt requests are enabled. |
| 3:0<br>RTCPS[3:0] | **Real-Time Clock Prescaler Select** — These four read/write bits and RTCLKS[0] select binary-based or decimal-based divide-by values for the clock source. See Table 10-4. Changing the prescaler value clears the prescaler and RTCCNT counters. |

Table 10-4. RTC Prescaler Divide-By Values

| RTCLKS[0] | RTCPS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | OFF | $2^3$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ | $2^{10}$ | 1 | 2 | $2^2$ | 10 | $2^4$ | $10^2$ | $5x10^2$ | $10^3$ |
| 1 | OFF | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{13}$ | $2^{14}$ | $2^{15}$ | $2^{16}$ | $10^3$ | $2x10^3$ | $5x10^3$ | $10^4$ | $2x10^4$ | $5x10^4$ | $10^5$ | $2x10^5$ |

## 10.7.2 RTC Counter Registers (RTCCNTH and RTCCNTL)

RTCCNT[15:0], which is equal to {RTCCNTH[7:0], RTCCNTL[7:0]}, is the read-only value of the current count of the 16-bit RTC counter.

Figure 10-3. RTC Counter Register (RTCCNT)

| | RTCCNTH | | | | 0x182B | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | RTCCNTH[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | RTCCNTL | | | | 0x1829 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | RTCCNTL[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 10-4. RTC Counter Register (RTCCNT)

Table 10-5. RTCCNT Field Description

| Field | Description |
|---|---|
| RTCCNT[15:0] | **RTC Count** — These sixteen read-only bits contain the current value of the 16-bit RTC counter. Writes have no effect to this register. Writing to RTCMOD or writing different values to RTCLKS and RTCPS clear the count to 0x0000. |

## 10.7.3 RTC Modulo Registers (RTCMODH and RTCMODL)

RTCMOD[15:0], which is equal to {RTCMODH[7:0], RTCMODL[7:0]}, is the modulo value against which the RTC counter is compared. The RTCMOD is loaded as a 16-bit value and requires that both registers be written before the 16-bit value is updated. The RTCMODH (high byte) should be written first and it gets latched. The RTCMODL (low byte) should then be written and causes the 2-byte value to be written to the RTCMOD register.

| | **RTCMODH** | | | | **0x182C** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | RTCMODH | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | **RTCMODL** | | | | **0x182A** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | RTCMODL | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 10-5. RTC Modulo Register (RTCMOD)

Table 10-6. RTCMOD Field Descriptions

| Field | Description |
|---|---|
| RTCMOD[15:0] | **RTC Modulo** — These sixteen read/write bits contain the modulo value used to reset the count to 0x0000 upon a compare match and set the RTIF status bit. A value of 0x0000 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x0000. |

# Chapter 11
# Modules (KBIx)

## 11.1 Introduction

There are two KBI modules. One has eight keyboard interrupt inputs that share port B pins. The other KBI module has four inputs that are shared on the four (4) port C pins. See the Pins and Connections chapter for more information about the logic and hardware aspects of these pins.

Port B is an 8-bit port that is shared between the KBI1 keyboard interrupt inputs and general-purpose I/O. The eight KBI1PEn control bits in the KBI1PE register allow selection of any combination of port B pins to be assigned as KBI1 inputs. Any pins that are enabled as KBI1 inputs will be forced to act as inputs and the remaining port B pins are available as general-purpose I/O pins controlled by the port B data (PTBD), data direction (PTBDD) and pullup enable (PTBPE) registers. The eight PTBPEn control bits in the PTBPE register allow the user to select whether an internal pullup device is enabled on each port B pin that is configured as a port input or a KBI1 input.

KBI1 inputs can be configured for edge-only sensitivity or edge-and-level sensitivity. Bits 3 through 0 of port B are falling-edge/low-level sensitive and bits 7 through 4 can be configured for rising-edge / high-level or for falling-edge/low-level sensitivity.

Port C is an 8-bit port with its four pins shared between the KBI2 keyboard interrupt inputs and general-purpose I/O. The four KBI2PEn control bits in the KBI2PE register allow selection of any combination of the lower four port C pins to be assigned as KBI2 inputs. Any pins that are enabled as KBI2 inputs will be forced to act as inputs and the remaining port C pins are available as general-purpose I/O pins controlled by the port C data (PTCD), data direction (PTCDD) and pullup enable (PTCPE) registers. The eight PTCPEn control bits in the PTCPE register allow the user to select whether an internal pullup device is enabled on each port C pin that is configured as a port input or a KBI2 input.

Any enabled keyboard interrupt can be used to wake the MCU from wait or standby (STOP3). Either KBI1 or KBI2 can be used to wake the MCU from wait or standby (STOP3). The KBF bits for both KBI modules must be cleared before entering Stop3 mode, regardless of whether the interrupt is enabled.

Note that MC13237 contains only the KBI1 module. MC13234 contains both KBI1 and KBI2.

### 11.1.1 KBI Clock Gating

The bus clock to the KBI can be gated on and off using the KBI1/KBI2 bits in SCGC2. These bits are set after any reset, which enables the bus clock to these modules. To conserve power, the KBI1/KBI2 bits can be cleared to disable the clock to these modules when not in use.

### 11.1.2 Features

The KBI features include:

- KBI1 with up to eight keyboard interrupt pins with individual pin enable bits.
- KBI2 with four 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.
- Independent software enabled keyboard interrupt for both KBI1 and KBI2.
- Exit from low-power modes.

### 11.1.3 Modes of Operation

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

#### 11.1.3.1 KBI in Wait Mode

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

#### 11.1.3.2 KBI in Stop Modes

The KBIx operates asynchronously in STOP3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of STOP3 mode if the KBI interrupt is enabled (KBIE = 1).

#### 11.1.3.3 KBI in Active Background Mode

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

### 11.1.4 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 11-1.

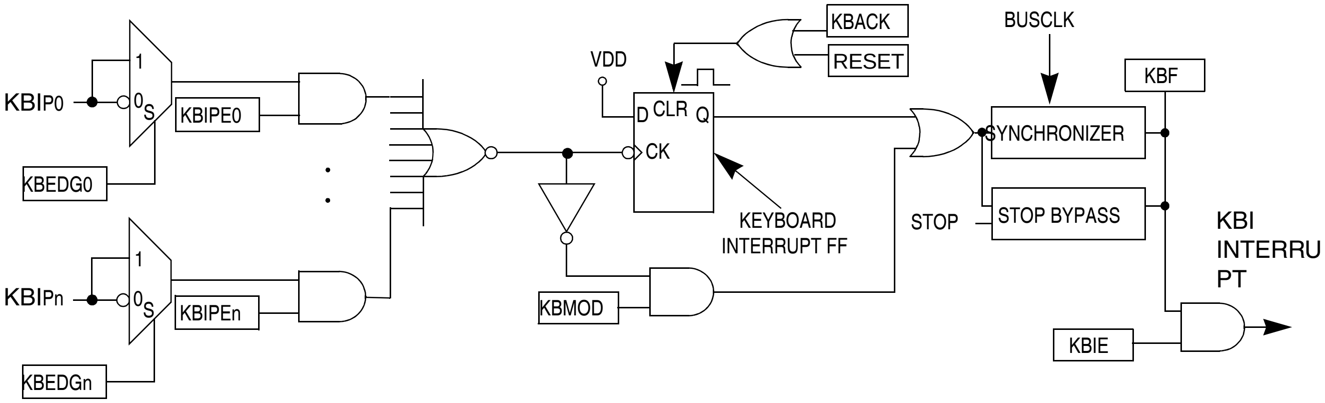


Figure 11-1. Havasu USB Architectural Definition Document Block Diagram

## 11.1.5 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 11-1. KBI1 Pin Mapping

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

Table 11-2. KBI2 Pin Mapping

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Port pin | | | | | PTC3 | PTC2 | PTC1 | PTC0 |
| KBI2 pin | | | | | KBI2P3 | KBI2P2 | KBI2P1 | KBI2P0 |

## 11.1.6 Register Definition

Each KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

KBI2 has four ports (KBI2P0–KBI2P3). These are available only in MC13234.

Refer to the direct-page register summary in Chapter 4, “Memory,” for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names and relative address offsets.

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

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

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

**Figure 11-2. KBI Interrupt Status and Control Register (KBIxSC)**

**Table 11-3. KBIxSC Register Field Descriptions**

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

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

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

**Figure 11-3. KBI Interrupt Pin Select Register (KBIxPE)**

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

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

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

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

**Figure 11-4. KBI Edge Select Register (KBIxES)**

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

| Field | Description |
|---|---|
| 7:0<br>KBEDG[7:0] | **KBI Edge Selects** — Each of the KBEDGn bits serves as selecting a pull-up or pull-down device if enabled.<br>0 A pull-up device is connected to the associated pin.<br>1 A pull-down device is connected to the associated pin. |

## 11.1.7 Functional Description

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

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.

### 11.1.7.1 Edge Only Sensitivity

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

### 11.1.7.2 Edge and Level Sensitivity

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

### 11.1.7.3 Pull-up/Pull-down Resistors

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

### 11.1.7.4 Keyboard Interrupt Initialization

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

1. Mask interrupts by clearing KBIE in KBIxSC.
2. If using internal pull-up/pull-down device, configure the associated pull enable bits in KBIxPE.
3. Enable the interrupt pins by setting the appropriate KBIPEn bits in KBIxPE.
4. Write to KBACK in KBIxSC to clear any false interrupts.

Set KBIE in KBIxSC to enable interrupts.

# Chapter 12
# Analog-to-Digital Converter (ADC)

## 12.1 Introduction

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip. The ADC module is available only in MC13237.

**NOTE**

The ADCCG bit in SOPT2 must be set for the ADC module to be enabled. See Section 5.8.5, “System Options Register 2 (SOPT2).”

### 12.1.1 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bit resolution
- Up to 8 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
- Converter subsystem shut-down capability

### 12.1.2 ADC Module Block Diagram

Figure 12-1 provides a block diagram of the ADC module.

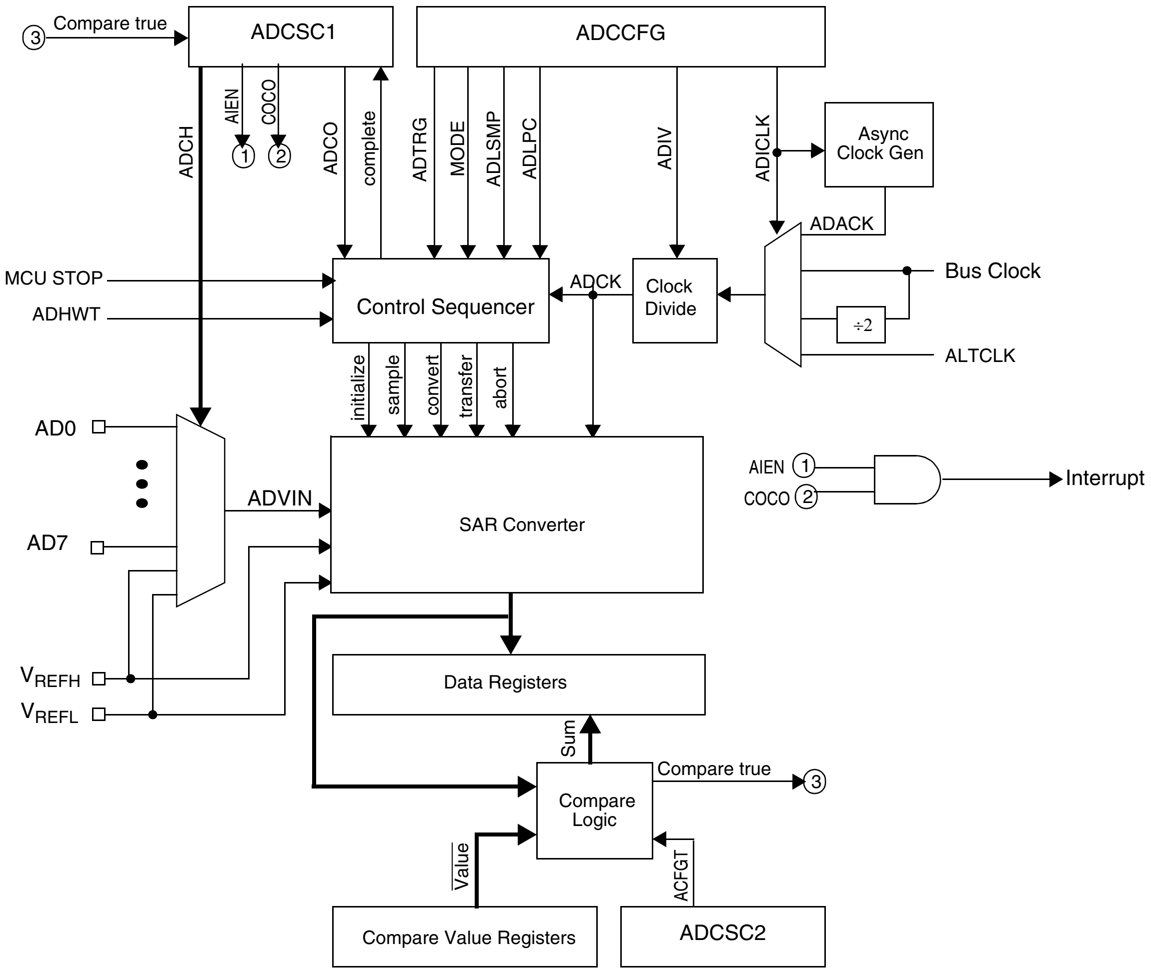


Figure 12-1. ADC Block Diagram

## 12.2 External Signal Description

The ADC module supports up to 8 separate analog inputs. It also requires four supply/reference/ground connections.

Table 12-1. Signal Properties

| Name | Function |
|---|---|
| AD7–AD0 | Analog Channel inputs |
| $V_{REFH}$ | High reference voltage |
| $V_{REFL}$ | Low reference voltage |
| $V_{DDA\_ADC}$ | Analog power supply |
| $V_{SSA\_ADC}$ | Analog ground |

### 12.2.1 Analog Power ($V_{DDA\_ADC}$)

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 better results.

### 12.2.2 Analog Ground ($V_{SSA\_ADC}$)

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 $V_{SS}$.

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

$V_{REFH}$ is the high reference voltage for the converter. In some packages, $V_{REFH}$ is connected internally to $V_{DDA}$. 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 $V_{DDA}$).

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

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

### 12.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 8 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

## 12.3 Register Definition

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

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin control registers, APCTLx[1]

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

### 12.3.1 Status and Control Register 1 (ADCSC1)

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

| | ADCSC1 | | | | 0x1870 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | COCO | AIEN | ADCO | ADCH | | | | |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-2. Status and Control Register (ADCSC1)**

**Table 12-2. ADCSC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>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.<br>0 Conversion not completed<br>1 Conversion completed |
| 6<br>AIEN | **Interrupt Enable** — AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted.<br>0 Conversion complete interrupt disabled<br>1 Conversion complete interrupt enabled |
| 5<br>ADCO | **Continuous Conversion Enable** — ADCO enables continuous conversions.<br>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.<br>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<br>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 12-3.<br>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 12-3. ADC Channel Assignment

| ADCH | Channel | Input | Pin Control |
|---|---|---|---|
| 00000 | AD0 | PTD1 | ADPC0 |
| 00001 | AD1 | PTD2 | ADPC1 |
| 00010 | AD2 | PTD3 | ADPC2 |
| 00011 | AD3 | PTD4 | ADPC3 |
| 00100 | AD4 | PTD5 | ADPC4 |
| 00101 | AD5 | PTD6 | ADPC5 |
| 00110 | AD6 | PTD7 | ADPC6 |
| 00111 | AD7 | PTC5 | ADPC7 |
| 01000 | AD8 | Reserved | N/A |
| 01001 | AD9 | Reserved | N/A |
| 01010 | AD10 | Reserved | N/A |
| 01011 | AD11 | Reserved | N/A |
| 01100 | AD12 | Reserved | N/A |
| 01101 | AD13 | Reserved | N/A |
| 01110 | AD14 | Reserved | N/A |
| 01111 | AD15 | Reserved | N/A |
| 10000 | AD16 | Reserved | N/A |
| 10001 | AD17 | Reserved | N/A |
| 10010 | AD18 | Reserved | N/A |
| 10011 | AD19 | Reserved | N/A |
| 10100 | AD20 | Reserved | N/A |
| 10101 | AD21 | Reserved | N/A |
| 10110 | AD22 | Reserved | N/A |
| 10111 | AD23 | Reserved | N/A |
| 11000 | AD24 | Reserved | N/A |
| 11001 | AD25 | Reserved | N/A |
| 11010 | AD26 | Temperature Sensor[1] | N/A |
| 11011 | AD27 | Internal Bandgap | N/A |
| 11100 | — | Reserved | N/A |
| 11101 | $V_{REFH}$ | $V_{REFH}$ | N/A |
| 11110 | $V_{REFL}$ | $V_{REFL}$ | N/A |
| 11111 | Module Disabled | None | N/A |

[1] For information, see Section 12.4.8, “Temperature Sensor.”

## 12.3.2 Status and Control Register 2 (ADCSC2)

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

ADCSC2 0x1871

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ADACT | ADTRG | ACFE | ACFGT | 0 | 0 | 0 | 0 |
| W | | | | | | | | |

Figure 12-3. Status and Control Register 2 (ADCSC2)

**Table 12-4. ADCSC2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>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.<br>0 Conversion not in progress<br>1 Conversion in progress |
| 6<br>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 (PTC[6]) input.<br>0 Software trigger selected<br>1 Hardware trigger selected |
| 5<br>ACFE | **Compare Function Enable** — Enables the compare function.<br>0 Compare function disabled<br>1 Compare function enabled |
| 4<br>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.<br>0 Compare triggers when input is less than compare value<br>1 Compare triggers when input is greater than or equal to compare value |

### 12.3.3 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 12.4.7, "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.

| | ADCRH | | | | 0x1872 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | 0 | 0 | ADR11 | ADR10 | ADR9 | ADR8 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-4. Data Result High Register (ADCRH)**

### 12.3.4 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 12.4.7, “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.

| | ADCRL | | | | 0x1873 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-5. Data Result Low Register (ADCRL)**

### 12.3.5 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.

| | ADCCVH | | | | 0x1874 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | 0 | 0 | ADCV11 | ADCV10 | ADCV9 | ADCV8 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-6. 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.

### 12.3.6 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.

| | ADCCVL | | | | 0x1875 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | ADCV7 | ADCV6 | ADCV5 | ADCV4 | ADCV3 | ADCV2 | ADCV1 | ADCV0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-7. Compare Value Low Register (ADCCVL)**

### 12.3.7 Configuration Register (ADCCFG)

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

| | ADCCFG | | | | 0x1876 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | ADLPC | ADIV | | ADLSMP | MODE | | ADICLK | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-8. Configuration Register (ADCCFG)**

**Table 12-5. ADCCFG Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>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.<br>0 High speed configuration<br>1 Low power configuration: The power is reduced at the expense of maximum clock speed. |
| 6:5<br>ADIV | **Clock Divide Select** — ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 12-6 shows the available clock configurations. |

Table 12-5. ADCCFG Register Field Descriptions (continued)

| Field | Description |
|---|---|
| 4<br>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.<br>0 Short sample time<br>1 Long sample time |
| 3:2<br>MODE | **Conversion Mode Selection** — MODE bits select between 12-, 10-, or 8-bit operation. See Table 12-7. |
| 1:0<br>ADICLK | **Input Clock Select** — ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 12-8. |

Table 12-6. Clock Divide Select

| ADIV | Divide Ratio | Clock Rate |
|---|---|---|
| 00 | 1 | Input clock |
| 01 | 2 | Input clock ÷ 2 |
| 10 | 4 | Input clock ÷ 4 |
| 11 | 8 | Input clock ÷ 8 |

Table 12-7. Conversion Modes

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

Table 12-8. Input Clock Select

| ADICLK | Selected Clock Source |
|---|---|
| 00 | Bus clock |
| 01 | Bus clock divided by 2 |
| 10 | Alternate clock (32 kHz oscillator) |
| 11 | Asynchronous clock (ADACK) |

### 12.3.8 Pin Control 1 Register (APCTL1)

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

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

| | APCTL1 | | | | 0x1877 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | ADPC7 | ADPC6 | ADPC5 | ADPC4 | ADPC3 | ADPC2 | ADPC1 | ADPC0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-9. Pin Control 1 Register (APCTL1)**

**Table 12-9. APCTL1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC7 | **ADC Pin Control 7** — ADPC7 controls the pin associated with channel AD7.<br>0 AD7 pin I/O control enabled<br>1 AD7 pin I/O control disabled |
| 6<br>ADPC6 | **ADC Pin Control 6** — ADPC6 controls the pin associated with channel AD6.<br>0 AD6 pin I/O control enabled<br>1 AD6 pin I/O control disabled |
| 5<br>ADPC5 | **ADC Pin Control 5** — ADPC5 controls the pin associated with channel AD5.<br>0 AD5 pin I/O control enabled<br>1 AD5 pin I/O control disabled |
| 4<br>ADPC4 | **ADC Pin Control 4** — ADPC4 controls the pin associated with channel AD4.<br>0 AD4 pin I/O control enabled<br>1 AD4 pin I/O control disabled |
| 3<br>ADPC3 | **ADC Pin Control 3** — ADPC3 controls the pin associated with channel AD3.<br>0 AD3 pin I/O control enabled<br>1 AD3 pin I/O control disabled |
| 2<br>ADPC2 | **ADC Pin Control 2** — ADPC2 controls the pin associated with channel AD2.<br>0 AD2 pin I/O control enabled<br>1 AD2 pin I/O control disabled |
| 1<br>ADPC1 | **ADC Pin Control 1** — ADPC1 controls the pin associated with channel AD1.<br>0 AD1 pin I/O control enabled<br>1 AD1 pin I/O control disabled |
| 0<br>ADPC0 | **ADC Pin Control 0** — ADPC0 controls the pin associated with channel AD0.<br>0 AD0 pin I/O control enabled<br>1 AD0 pin I/O control disabled |

## 12.4 Functional Description

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

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. 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.

### 12.4.1 Clock Select and Divide Control

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

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by 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.

### 12.4.2 Input Select and Pin Control

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

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

### 12.4.3 Asynchronous and Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, or the local asynchronous clock (ADACK) within the module. The alternate clock, a 32 kHz clock, can be used for ADC conversion. It has to be enabled in SOMC1 before performing ADC conversion.

### 12.4.4 Software Trigger

The ADC conversion can be initiated by software trigger. To select the software trigger, ADTRG in the ADCSC2 register must be set to 0. When software trigger is selected, writes to ADCSC1 will start a new ADC conversion.

### 12.4.5 Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. ADHWT is connected to PTC[6] or RTC. The selection between PTC[6] and RTC for ADHWT is in SOPT1[ADCHT].

When the 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.

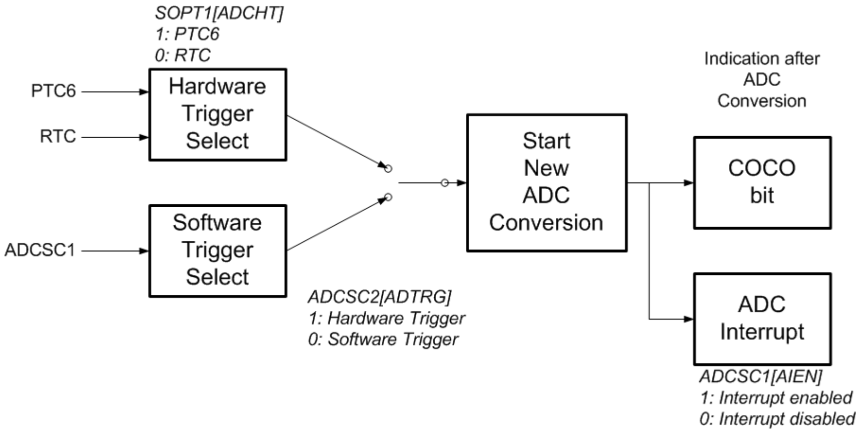


**Figure 12-10. ADC Triggering**

**Table 12-10.**

| Trigger Source | ADCSC2[ADTRG] | SOPT1[ADCHT] |
|---|---|---|
| Software (ADCSC1) | 0 | 0 |
| Software (ADCSC1) | 0 | 1 |
| RTC | 1 | 0 |
| PTC[6] | 1 | 1 |

### 12.4.6 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.

#### 12.4.6.1 Initiating Conversions

A conversion is initiated:

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

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

#### 12.4.6.2 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.

#### 12.4.6.3 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.

#### 12.4.6.4 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 $f_{ADCK}$.

#### 12.4.6.5 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 and long sample times.When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

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

The maximum total conversion time for different conditions is summarized in Table 12-11.

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

**Table 12-11. Total Conversion Time vs. Control Conditions**

| Conversion Type | ADICLK | ADLSMP | Max Total Conversion Time |
|---|---|---|---|
| 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:

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

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

**NOTE**

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

### 12.4.7 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.

```
   %101 1111 1111         <= 1’s complement of 0x200 compare value
 +            %1
   ---------------
   %110 0000 0000         <= 2’s complement of 0x200 compare value
```

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

```
     %000 1000 0000
  +  %110 0000 0000
     ---------------
      %110 1000 0000       <= Subtraction result is –0x180 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 ACFGT=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 (ACFGT=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 (ACFGT=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)
```

### 12.4.8 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 12-1 provides an approximate transfer function of the temperature sensor.

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

where:

- — $V_{TEMP}$ is the voltage of the temperature sensor channel at the ambient temperature.
- — $V_{TEMP25}$ is the voltage of the temperature sensor channel at 25°C.

— m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the $V_{TEMP25}$ and m values from the ADC Electricals table.

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 12-1. If $V_{TEMP}$ is less than $V_{TEMP25}$ the hot slope value is applied in Equation 12-1.

### 12.4.9 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, asynchronous clock, and alternate clock are available as conversion clock sources while in Wait mode.

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

### 12.4.10 MCU Stop3 Mode Operation

Stop3 mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

#### 12.4.10.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by Stop3 mode. After exiting from Stop3 mode, a software or hardware trigger is required to resume conversions.

#### 12.4.10.2 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 12.4.6.2, “Completing Conversions) is cleared when entering Stop3 and continuing ADC conversions.

## 12.5 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 12-6, Table 12-7, and Table 12-8 for information used in this example.

**NOTE**

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

### 12.5.1 ADC Module Initialization Example

#### 12.5.1.1 Initialization Sequence

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

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

#### 12.5.1.2 Pseudo-Code Example

In this example, the ADC module 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

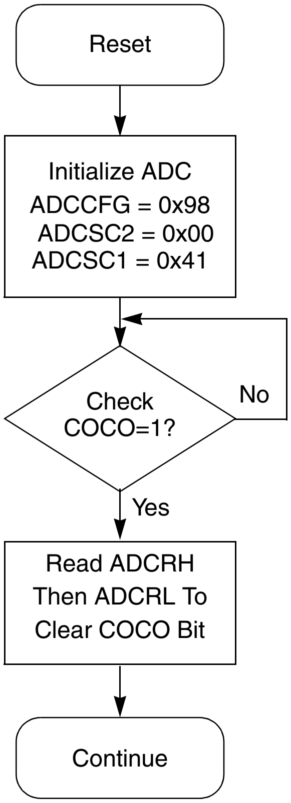


**Figure 12-11. Initialization Flowchart for Example**

## 12.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

### 12.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

#### 12.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies ($V_{DDA\_ADC}$ and $V_{SSA\_ADC}$) available as separate pins on some devices. $V_{SSA\_ADC}$ is shared on the same pin as the MCU digital $V_{SS}$ on some devices. On other devices, $V_{SSA\_ADC}$ and $V_{DDA\_ADC}$ 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 $V_{DDA\_ADC}$ and $V_{SSA\_ADC}$ 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 $V_{SSA\_ADC}$ pin. This should be the only ground connection between these supplies if possible. The $V_{SSA\_ADC}$ pin makes a good single point ground location.

### 12.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is $V_{REFH}$, which may be shared on the same pin as $V_{DDA\_ADC}$ on some devices. The low reference is $V_{REFL}$, which may be shared on the same pin as $V_{SSA\_ADC}$ on some devices.

When available on a separate pin, $V_{REFH}$ may be connected to the same potential as $V_{DDA\_ADC}$, or may be driven by an external source between the minimum $V_{DDA\_ADC}$ spec and the $V_{DDA\_ADC}$ potential ($V_{REFH}$ must never exceed $V_{DDA\_ADC}$). When available on a separate pin, $V_{REFL}$ must be connected to the same voltage potential as $V_{SSA\_ADC}$. $V_{REFH}$ 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).

### 12.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer 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 $V_{SSA\_ADC}$.

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.

## 12.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

### 12.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 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 kΩ.

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

### 12.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance ($R_{AS}$) is high. If this error cannot be tolerated by the application, keep $R_{AS}$ lower than $V_{DDA\_ADC}$ / ($2^N$*$I_{LEAK}$) for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

### 12.6.2.3 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 μF low-ESR capacitor from $V_{REFH}$ to $V_{REFL}$.
- There is a 0.1 μF low-ESR capacitor from $V_{DDA}$ to $V_{SSA}$.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from $V_{DDA\_ADC}$ to $V_{SSA\_ADC}$.
- $V_{SSA\_ADC}$ (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 $V_{SSA\_ADC}$ (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.

### 12.6.2.4 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:

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

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code 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 ± 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.

### 12.6.2.5 Linearity Errors

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

- Zero-scale error ($E_{ZS}$) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 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.

#### 12.6.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

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

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb 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 12.6.2.3 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 13
# Inter-integrate Circuit (IIC)

## 13.1 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 800 kps (max bus 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**

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

### 13.1.1 Features

The IIC includes these distinctive features:

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

### 13.1.2 Block Diagram

Figure 13-1 is a block diagram of the IIC

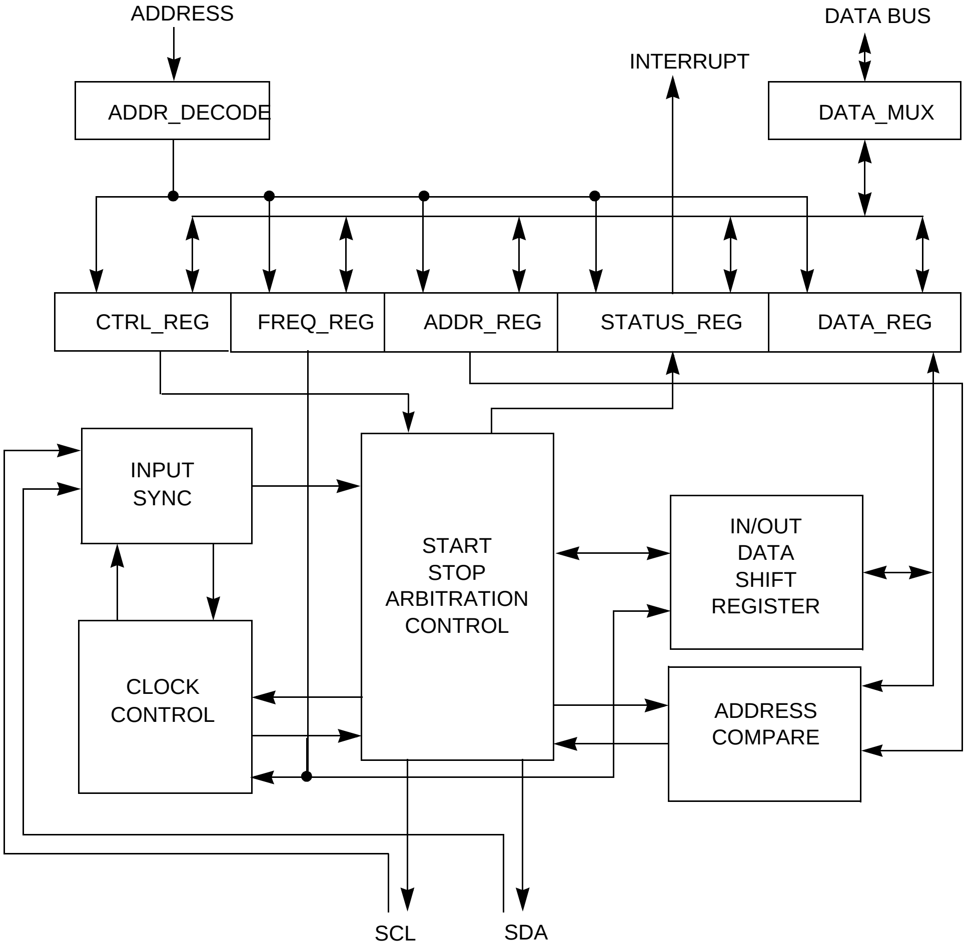


**Figure 13-1. IIC Functional Block Diagram**

## 13.2 External Signal Description

This section describes the two IIC external signals.

1. SCL — Serial Clock Line; the bidirectional SCL is the serial clock line of the IIC system.
2. SDA — Serial Data Line; the bidirectional SDA is the serial data line of the IIC system..

## 13.3 Register Definitions

This section describes the IIC control and status registers. The register addresses are located in the Direct-Page Register map.

**Table 13-1. Module Memory Map**

| Address | Use | Access |
|---|---|---|
| 0x006A | IIC Address Register (IIC1A) | read/write |
| 0x006B | IIC Frequency Divider Register (IIC1F) | read/write |
| 0x006C | IIC Control Register (IIC1C1) | read/write |
| 0x006D | IIC Status Register (IIC1S) | read |
| 0x006E | IIC Data IO Register (IIC1D) | read/write |
| 0x006F | IIC Control Register 2 (IIC1C2) | read/write |

### 13.3.1 IIC Address Register (IIC1A)

IIC1A is the slave address register for the module.

| | IIC1A | | | | 0x006A | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | 0 |
| W | | | | | | | | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 13-2. IIC Address Register (IIC1A)**

**Table 13-2. IIC1A Field Descriptions**

| Field | Description |
|---|---|
| 7:1<br>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 in the 7-bit address scheme and as the lower seven bits of the 10-bit address scheme. |

## 13.3.2 IIC Frequency Divider Register (IIC1F)

IIC1F register controls generation of the IIC bus baud clock and bus timing. The IIC module clock is driven from the bus clock (always 1/2 CPU clock) which is typically 16 MHz for the MC13234/MC13237. As may be required for variable bus loading, it is possible to “tune” the bus hold times. The register contains two fields used for varying the bus baud rate and hold timing:

1. IIC Multiplier Factor (MULT[1:0]) - this 2-bit field selects a “**Mul**” multiplication factor as shown in Table 13-3.

**Table 13-3. IIC Multiplication Factor**

| MULT[1:0] | Mul |
|---|---|
| 00 | 1 Times |
| 01 | 2 Times |
| 10 | 4 Times |
| 11 | Reserved |

2. IIC Clock Rate (ICR[5:0]) - this 6-bit field programs and selects four hardware factors (see Table 13-6) that along with the **MUL** factor determine IIC baud rate and bus hold times. These factors are:
   - **SCL_Divider -** multiplied by multiplier factor **Mul,** is used to generate IIC baud rate:

     IIC baud rate = bus speed (Hz) / (**Mul * SCL_Divider**) ***Eqn. 13-1***

   - **SDA_Hold_Value** - SDA hold time is the delay from the falling edge of SDA (IIC data) to the changing of SDA (IIC data) and is calculated as follows:

     SDA hold time = bus period (s) *** (Mul * SDA_Hold_Value)** *Eqn. 13-2*

   - **SCL_Start_Hold_Value** - 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) and is calculated as follows:

     SCL Start hold time = bus period (s) *** (Mul * SCL_Start_Hold_Value)** ***Eqn. 13-3***

   - **SCL_Stop_Hold_Value** - 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) and is calculated as follows:

     SCL Stop hold time = bus period (s) *** (Mul * SCL_Stop_Hold_Value)** ***Eqn. 13-4***

Table 13-4 shows the possible hold time values with different ICR and MULT selections to achieve an IIC baud rate of 100 kbps using a 16 MHz bus clock.

### Table 13-4. IIC Hold Times for 100 kbps Baud Rate and 16 MHz Bus Clock

| MULT[1:0] | ICR | Hold times (µs) | | |
|---|---|---|---|---|
| | | SDA | SCL Start | SCL Stop |
| 0x2 (Mul = 4) | 0x07 | 2.50 | 4.00 | 5.25 |
| 0x2 (Mul = 4) | 0x0B | 2.25 | 4.00 | 5.25 |
| 0x1 (Mul = 2) | 0x14 | 2.12 | 4.25 | 5.12 |
| 0x1 (Mul = 2) | 0x18 | 1.12 | 4.75 | 5.12 |
| 0x0 (Mul = 1) | 0x1D | 1.56 | 4.88 | 5.06 |
| 0x0 (Mul = 1) | 0x20 | 1.06 | 4.88 | 5.06 |

| IIC1F | | | | | 0x006B | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | MULT | | ICR | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 13-3. IIC Frequency Divider Register (IIC1F)

### Table 13-5. IIC1F Field Descriptions

| Field | Description |
|---|---|
| 7:6<br>MULT[1:0] | **IIC Multiplier Factor** — The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate the IIC baud rate. See Table 13-3 |
| 5:0<br>ICR[5:0] | **IIC Clock Rate** — The ICR bits are used to pre scale the bus clock for bit rate selection. These bits and the MULT bits are used to determine the IIC baud rate, the SDA hold time, the SCL Start hold time and the SCL Stop hold time. Table 13-6 provides the SCL divider and hold values for corresponding values of the ICR. |

**Table 13-6. IIC Divider and Hold Values versus ICR Field Setting**

| ICR (hex) | SCL_ Divider | SDA_ Hold_ Value | SCL_ Start_ Hold_ Value | SDA_ Stop_ Hold_ 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 |
| 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 |

### 13.3.3 IIC Control Register (IIC1C1)

IIC1C1 if the primary of two control registers for the IIC module.

IIC1C1 0x006C

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

▨ = Unimplemented or Reserved

**Figure 13-4. IIC Control Register (IIC1C1)**

**Table 13-7. IIC1C1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IICEN | **IIC Enable** — The IICEN bit determines whether the IIC module is enabled.<br>0 IIC is not enabled.<br>1 IIC is enabled. |
| 6<br>IICIE | **IIC Interrupt Enable** — The IICIE bit determines whether an IIC interrupt is requested.<br>0 IIC interrupt request not enabled.<br>1 IIC interrupt request enabled. |
| 5<br>MST | **Master Mode Select** — The MST bit is changed from a 0 to a 1 when a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a STOP signal is generated and the mode of operation changes from master to slave.<br>0 Slave mode.<br>1 Master mode. |
| 4<br>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 will always be high. When addressed as a slave this bit should be set by software according to the SRW bit in the status register.<br>0 Receive.<br>1 Transmit. |
| 3<br>TXAK | **Transmit Acknowledge Enable** — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers.<br>0 An acknowledge signal will be sent out to the bus after receiving one data byte.<br>1 No acknowledge signal response is sent. |
| 2<br>RSTA | **Repeat START** — Writing a 1 to this bit will generate a repeated START condition provided it is the current master. This bit will always be read as a low{iic_reg.asm}. Attempting a repeat at the wrong time will result in loss of arbitration. |

### 13.3.4 IIC Status Register (IIC1S)

IIC1S is the status register for the IIC module.

| IIC1S | | | | | 0x006D | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| W | ▨ | | ▨ | | ▨ | ▨ | | ▨ |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 13-5. IIC Status Register (IIC1S)**

**Table 13-8. IIC1S Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TCF | **Transfer Complete Flag** — This bit is set on the completion of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IIC1D register in receive mode or writing to the IIC1D in transmit mode.<br>0 Transfer in progress.<br>1 Transfer complete. |
| 6<br>IAAS | **Addressed as a Slave** — The IAAS bit is set when the calling address matches the programmed slave address, or when the GCAEN bit is set and a general call is received. Writing the IIC1C register clears this bit.<br>0 Not addressed.<br>1 Addressed as a slave. |
| 5<br>BUSY | **Bus Busy** — The BUSY bit indicates the status of the bus regardless of slave or master mode{iic_bus_busy.asm}. The BUSY bit is set when a START signal is detected and cleared when a STOP signal is detected{iic_bus_busy.asm}.<br>0 Bus is idle{iic_bus_busy.asm}.<br>1 Bus is busy{iic_bus_busy.asm}. |
| 4<br>ARBL | **Arbitration Lost** — This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing a 1 to it.<br>0 Standard bus operation.<br>1 Loss of arbitration. |
| 2<br>SRW | **Slave Read Only**— When addressed as a slave the SRW bit indicates the value of the R/$\overline{W}$ command bit of the calling address sent to the master.<br>0 Slave receive, master writing to slave.<br>1 Slave transmit, master reading from slave. |

Table 13-8. IIC1S Field Descriptions (continued)

| Field | Description |
|---|---|
| 1<br>IICIF | **IIC Interrupt Flag** — The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit:<br>• One byte transfer completes<br>• Match of slave address to calling address<br>• Arbitration lost<br>0 No interrupt pending.<br>1 Interrupt pending. |
| 0<br>RXAK | **Receive Acknowledge** — When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected.<br>0 Acknowledge received.<br>1 No acknowledge received. |

## 13.3.5 IIC Data I/O Register (IIC1D)

IIC1D is the data transfer register for the IIC module.

| | IIC1D | | | | 0x006E | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | DATA | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 13-6. IIC Data I/O Register (IIC1D)

Table 13-9. IIC1D Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>DATA[7:0] | **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 in-advertent initiation of a master receive data transfer.

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

Note that the TX bit in IIC1C must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, then reading the IIC1D will not initiate the receive.

Reading the IIC1D will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IIC1D does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IIC1D correctly by reading it back.

In master transmit mode, the first byte of data written to IIC1D following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{\text{W}}$ bit (in position bit 0).

### 13.3.6 IIC Control Register 2 (IIC1C2)

IIC1C2 is the second of two IIC module control registers.

IIC1C2 0x06F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | GCAEN | ADEXT | 0 | 0 | 0 | AD[10:8] | | |
| W | | | ▨ | ▨ | ▨ | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 13-7. IIC Control Register 2 (IIC1C2)**

**Table 13-10. IIC1C2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>GCAEN | **General Call Address Enable** — The GCAEN bit enables or disables general call address.<br>0 General call address is disabled<br>1 General call address is enabled. |
| 6<br>ADEXT | **Address Extension** — The ADEXT bit controls the number of bits used for the slave address.<br>0 7-bit address scheme<br>1 10-bit address scheme |
| 2:0<br>AD[10:8] | **Slave Address[10:8]** — 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. |

## 13.4 Functional Description

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

### 13.4.1 IIC Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pullup 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 shown in Figure 13-8.

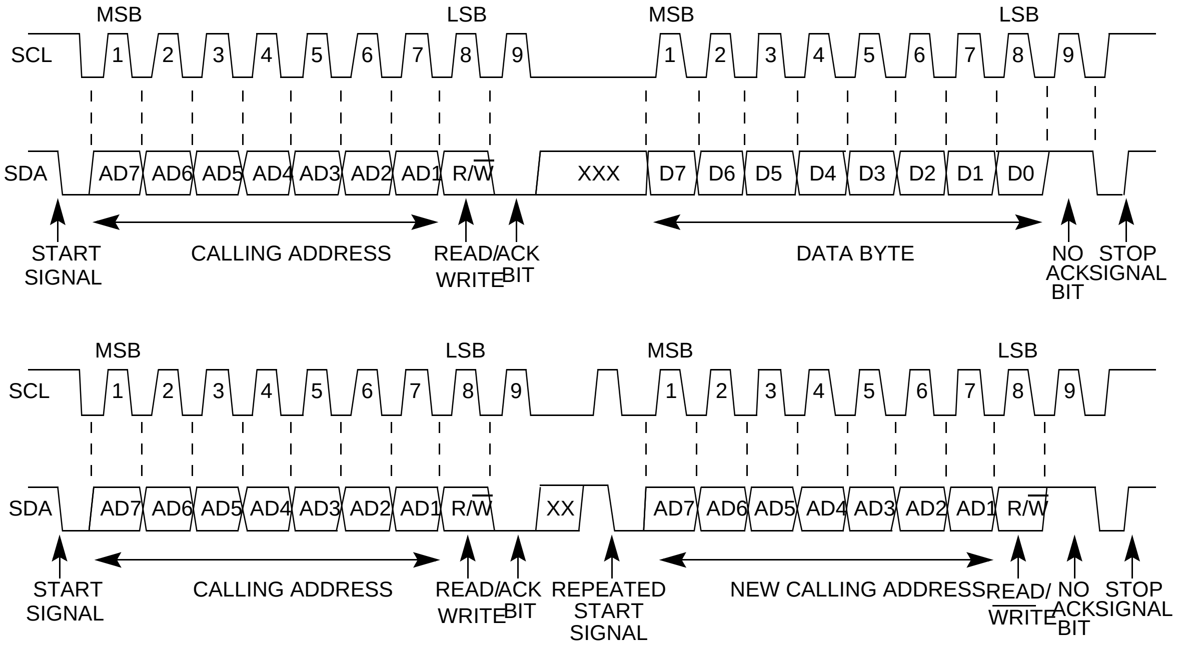


**Figure 13-8. IIC Bus Transmission Signals**

### 13.4.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logic high), a master may initiate communication by sending a START signal. As shown in Figure 13-8, 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.

### 13.4.1.2 Slave Address Transmission

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

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

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 13-8).

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

### 13.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{\text{W}}$ 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 13-8. 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 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

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

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

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.

### 13.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logic 1 (see Figure 13-8).

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

### 13.4.1.5 Repeated START Signal

As shown in Figure 13-8, 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.

### 13.4.1.6 Arbitration Procedure

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

### 13.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 13-9). 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.

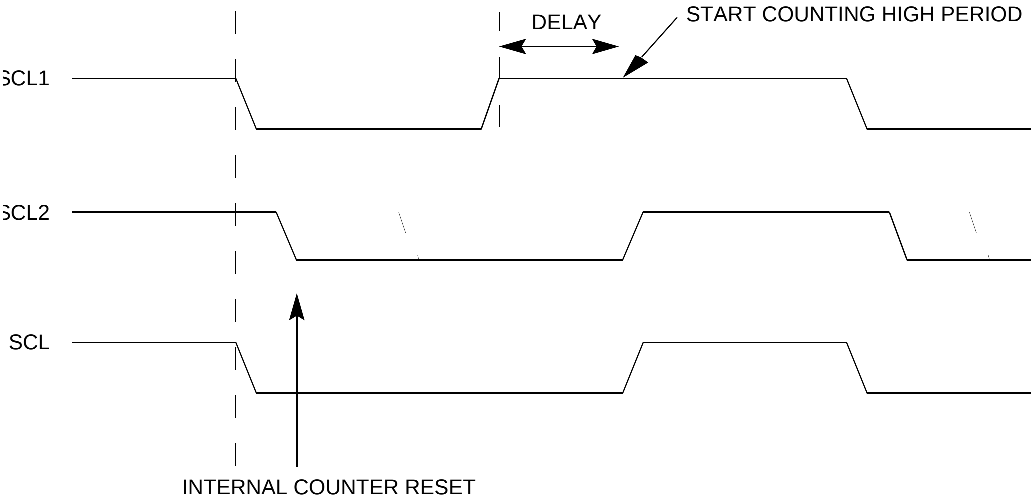


Figure 13-9. IIC Clock Synchronization

### 13.4.1.8 Handshaking

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

### 13.4.1.9 Clock Stretching

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

## 13.4.2 10-bit Address

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

### 13.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 13-11). When a 10-bit address follows a START condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit ($R/\overline{W}$ direction bit) is 0. It is possible that more than one device will find a match and generate an acknowledge (A1). Each slave that finds a match will compare the eight bits of the second byte of the slave address with its own address, but only one slave will find a match and generate an acknowledge (A2). The matching slave will remain addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

**Table 13-11. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address**

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

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

### 13.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second $R/\overline{W}$ bit (see Table 13-12). 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 will 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 will be addressed because $R/\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

**Table 13-12. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address**

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

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

### 13.4.3 General Call Address

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

### 13.4.4 Resets

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

### 13.4.5 Interrupts

The IIC generates a single interrupt request. The IIC has Interrupt Vector No. 24.

An interrupt from the IIC is generated when any of the events in Table 13-13 occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. The user can determine the interrupt type by reading the status register.

**Table 13-13. 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 |

#### 13.4.5.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

#### 13.4.5.2 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.

### 13.4.5.3 Arbitration Lost Interrupt

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

Arbitration is lost in the following circumstances:

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

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

## 13.5 Modes of operation

The IIC is affected by the device mode of operation and the module clock can be disabled:

- For lower power, the IIC module clock can be disabled via the IIC control bit of the SCGC1 Register
- LPRun - the reference oscillator is divided by 64 and all clocks are scaled by this factor. It is recommended to disable the IIC during this mode as all frequencies and timing changes accordingly.
- Wait modes - there are two wait modes
  - Wait - when entered, the CPU clock is disabled and the bus clock still runs at normal speed . The IIC, if enabled, will continue to operate normally until an interrupt is required. However, there will be no new codes or changes of operation while in wait mode, because the CPU is not operating.
  - LP Wait - when entered, the CPU clock is disabled and the bus clock still runs but at a reduced speed. The IIC, if enabled, will continue to operate, however, all timing would be incorrect. Operation is not recommended.
- The TPM is disabled in Stop3 mode, regardless of the settings before executing the STOP instruction.
  - During Stop3 mode, clocks to the IIC module are halted. No registers are affected. If Stop3 is exited with a reset, the IIC will be put into its reset state. If Stop3 is exited with an interrupt, the IIC will resume operation. It is recommended that the IIC be halted before entering Stop3.
- Background Mode Operation -When the microcontroller is in active background mode, the IIC temporarily suspends until the microcontroller returns to normal user mode.

# Chapter 14
# Serial Communications Interface (SCI)

## 14.1 Introduction

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.

## 14.2 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Fractional baud rates (5-bit 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 - can be used for device wake-up
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wake up by idle-line or address-mark
- MC13234/MC13237 wake-up capability from Stop3 on an RXD input transition
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

## 14.3 Block Diagrams

Figure 14-2 shows the baud rate generator of the SCI.

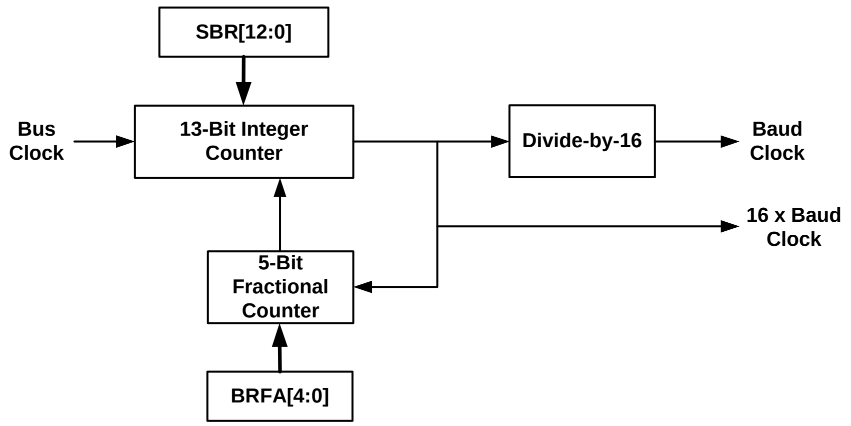


**Figure 14-1. SCI Baud Rate Generator**

Figure 14-2 shows the transmitter portion of the SCI.

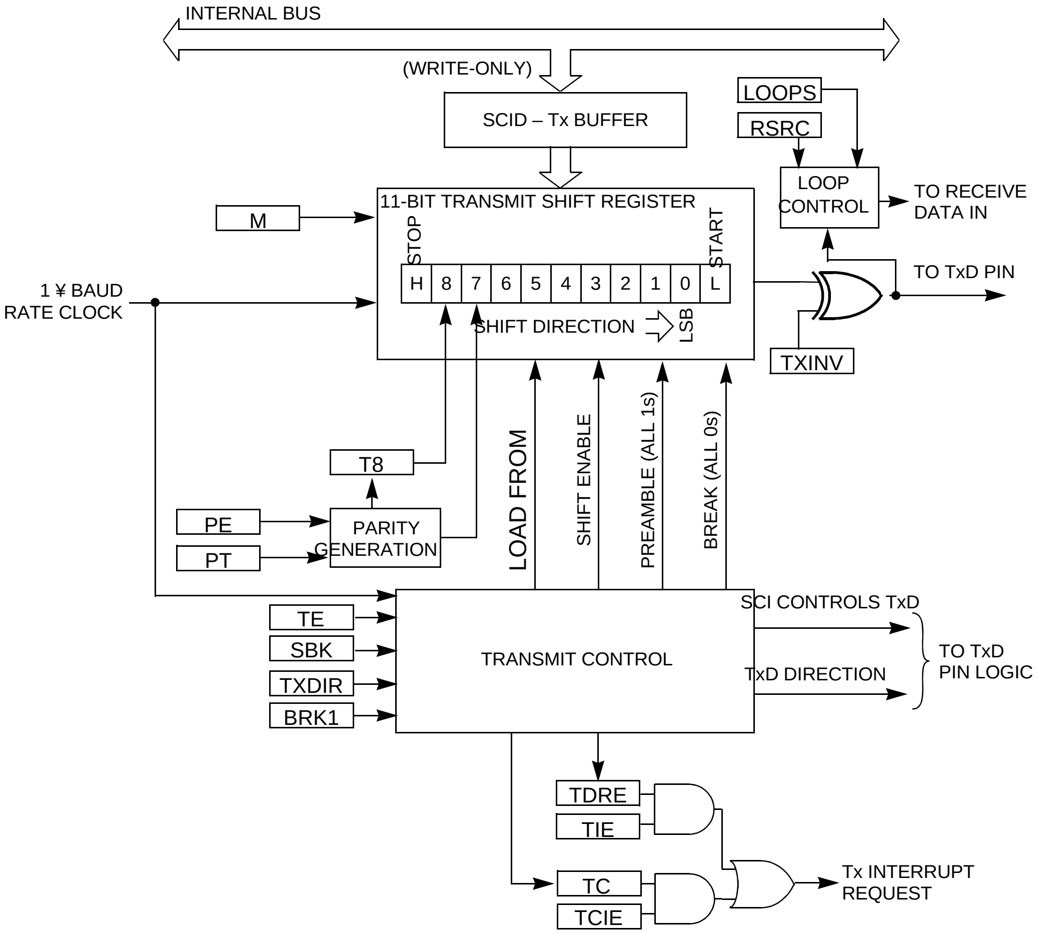


**Figure 14-2. SCI Transmitter Block Diagram**

Figure 14-3 shows the receiver portion of the SCI.

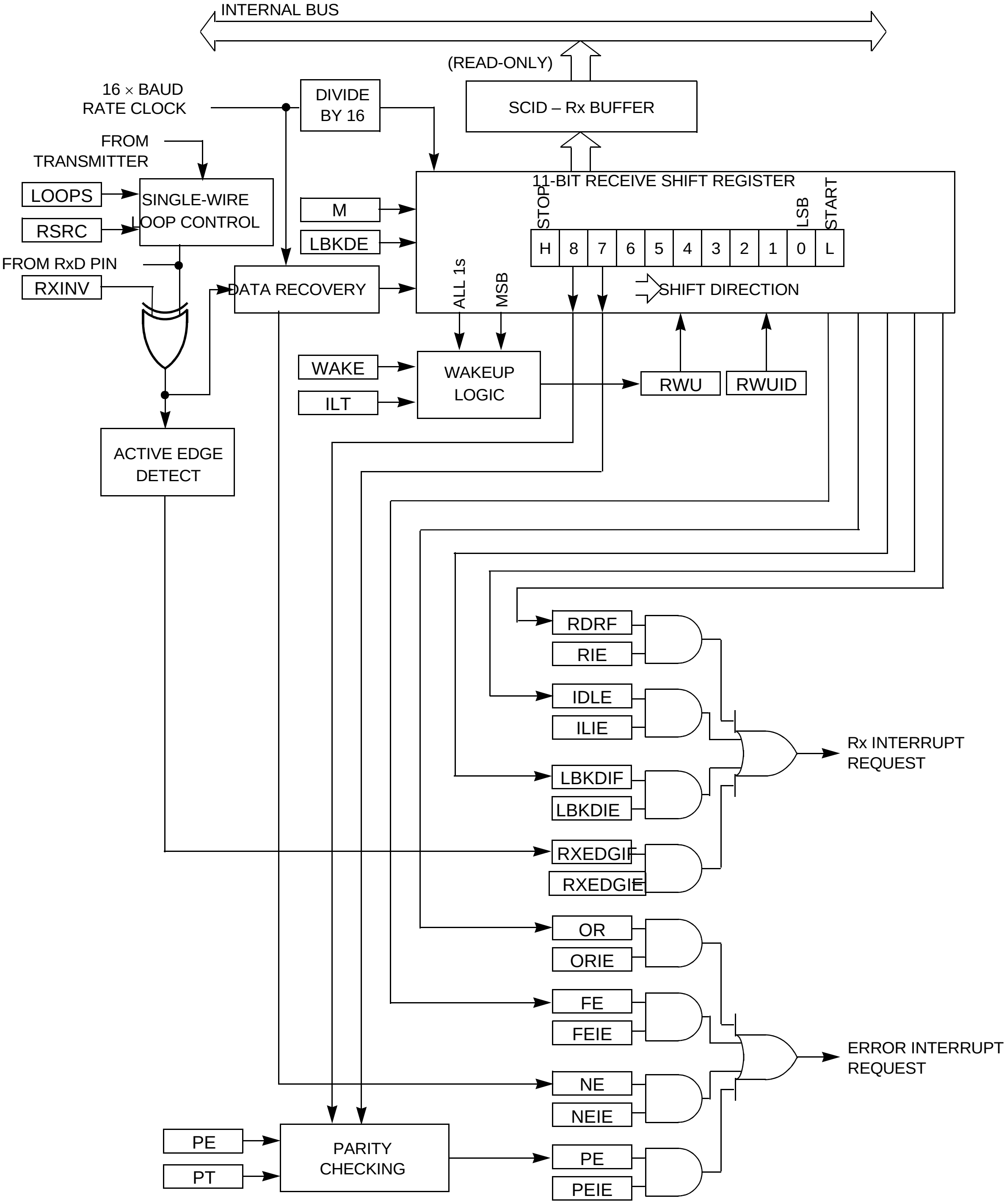


**Figure 14-3. SCI Receiver Block Diagram**

## 14.4 External Signal Description

The UART shares two port pins.

### 14.4.1 TXD — Transmit Data Output

When the SCI is enabled, this pin is the asynchronous transmit data output.

### 14.4.2 RXD — Receive Data Input

When the SCI is enabled, this pin is the asynchronous receive data input.

## 14.5 Functional Description

The SCI has three basic blocks consisting of the baud rate generator, the transmitter, and the receiver.

### 14.5.1 Baud Rate Generator

As shown in Figure 14-1, the baud rate generator is driven from the Bus Clock and generates a 1 x baud clock and a 16 x baud clock. The 16 x baud clock is derived from a 13-bit modulus counter and a 5-bit fractional fine-adjust counter. The 13-bit counter determines the primary integer clock divisor, and it is programmed via the SBR[12:0] field of the SCI1BDH and SCI1BDL Registers (see Section 14.6.1, “SCI Baud Rate Registers (SCI1BDH, SCI1BDL)”). Valid divisor values are from 1 to 8191; a value of zero disables the counter.

The fractional fine-adjust counter adds fractional delays to the baud rate clock to allow a much higher accuracy baud rate clock than can be derived from a simple prescaler. The 5-bit fractional counter is driven from the 16 x baud clock and causes the primary counter to increase the divide divisor by one on a periodic basis as determined by its programming. The BRFA[4:0] field of the SCI1C4 Register (see Section 14.6.8, “SCI Control Register 4 (SCI1C4)”) allows selection of 32 different fractional divisors that increase the integer divisor by the amount shown in Table 14-2.

The 16 x baud rate clock is used by the receiver to sample (at a 16x rate) and synchronize the incoming RX data. The 1 x baud rate clock is used the the transmitter to send TX data at the selected baud rate.

The equation for determining the baud rate is:

$$\text{SCI baud rate} = \text{Bus Clock} / (16 * (\text{SBR}[12:0]_{\text{dec}} + \text{BRFD}_{\text{dec}})) \qquad \textit{Eqn. 14-1}$$

As an example, the typical bus clock for the MC13234/MC13237 is 16 MHz and a standard rate is 115,200 baud. For this case, the integer is first determined assuming no fractional divide:

SBR[12:0]dec = 16 MHz / (115,200 * 16) = 8.68 = 8 (integer)

If no fractional value is added, the actual baud rate is:

Baud rate = 16 MHz / (16 * 8) = 125,000

which gives an error of +8.5% which is undesirable. The closest BRFD value is 0.6875 (BRFA[4:0] = $22_{\text{dec}}$) and if the fractional option is added:

Baud rate = 16 MHz / (16 * 8.6875) = 115,108

The final error is -0.08% which is very good.

Table 14-1 lists common standard baud rates with a bus clock frequency of 16 MHz using fractional fine adjustment.

**Table 14-1. SCI Baud Rates Using 16 MHz Bus Clock**

| Standard Baud Rate | SBR[12:0] (dec) | BRFA[4:0] (dec) | BRFD (dec) | Actual Baud Rate | Error |
|---|---|---|---|---|---|
| 1200 | 833 | 11 | 0.34375 | 1199.98 | -0.09% |
| 2400 | 416 | 25 | 0.78125 | 2399.34 | -0.03% |
| 4800 | 208 | 11 | 0.34375 | 4799.76 | -0.01% |
| 9600 | 104 | 5 | 0.15625 | 9600.96 | +0.01% |
| 19200 | 52 | 3 | 0.09375 | 19196.16 | +0.02% |
| 28800 | 34 | 23 | 0.71875 | 28802.88 | +0.01% |
| 38400 | 26 | 0 | 0 | 38461.53 | +0.16% |
| 57600 | 17 | 12 | 0.375 | 57553.96 | -0.08% |
| 115200 | 8 | 22 | 0.6875 | 115107.91 | -0.08% |
| 230400 | 4 | 11 | 0.34375 | 230215.83 | -0.08% |
| 460800 | 2 | 5 | 0.15625 | 463768.12 | +0.64% |

Table 14-2 lists the available Baud Rate Fine Adjust values (BRFA[4:0]) versus the Baud Rate Fractional Divisor (BRFD)

**Table 14-2. SCI Baud Rate Fine Adjust (BRFA)**

| BRFA[4:0] (dec) | Baud Rate Fractional Divisor (BRFD) |
|---|---|
| 0 | 0/32 = 0 |
| 1 | 1/32 = 0.03125 |
| 2 | 2/32 = 0.0625 |
| 3 | 3/32 = 0.09375 |
| 4 | 4/32 = 0.125 |
| 5 | 5/32 = 0.15625 |
| 6 | 6/32 = 0.1875 |
| 7 | 7/32 = 0.21875 |
| 8 | 8/32 = 0.25 |
| 9 | 9/32 = 0.28125 |
| 10 | 10/32 = 0.3125 |
| 11 | 11/32 = 0.34375 |

Table 14-2. SCI Baud Rate Fine Adjust (BRFA) (continued)

| BRFA[4:0] (dec) | Baud Rate Fractional Divisor (BRFD) |
|---|---|
| 12 | 12/32 = 0.375 |
| 13 | 13/32 = 0.40625 |
| 14 | 14/32 = 0.4375 |
| 15 | 15/32 = 0.46875 |
| 16 | 16/32 = 0.5 |
| 17 | 17/32 = 0.53125 |
| 18 | 18/32 = 0.5625 |
| 19 | 19/32 = 0.59375 |
| 20 | 20/32 = 0.625 |
| 21 | 21/32 = 0.65625 |
| 22 | 22/32 = 0.6875 |
| 23 | 23/32 = 0.71875 |
| 24 | 24/32 = 0.75 |
| 25 | 25/32 = 0.78125 |
| 26 | 26/32 = 0.8125 |
| 27 | 27/32 = 0.84375 |
| 28 | 28/32 = 0.875 |
| 29 | 29/32 = 0.90625 |
| 30 | 30/32 = 0.9375 |
| 31 | 31/32 = 0.96875 |

### 14.5.2 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 14-2.

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCI1C2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

#### 14.5.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wake up is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

**Table 14-3. 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 |

## 14.5.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 14-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wake up function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, see Section 14.5.5.1, “8- and 9-Bit Data Modes”. For the remainder of this topic, assume that the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user’s program that handles receive data. See Section 14.5.4, “Interrupts and Status Flags” for more details about flag clearing.

### 14.5.3.1 Data Sampling Technique

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

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

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

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

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

### 14.5.3.2 Receiver Wake up Operation

Receiver wake up is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

#### 14.5.3.2.1 Idle-Line Wake up

When WAKE = 0, the receiver is configured for idle-line wake up. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

#### 14.5.3.2.2 Address-Mark Wake up

When WAKE = 1, the receiver is configured for address-mark wake up. 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 wake up allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

## 14.5.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1. Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while RDRF = 1 and then reading SCIxD.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while IDLE = 1 and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled (RE = 1), and as a result can be used as a wake-up function from low power mode or to enable the UART if it has been disabled.

### 14.5.5 Additional SCI Functions

The following sections describe additional SCI functions.

#### 14.5.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCIxD.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wake up so the ninth data bit can serve as the wake up bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

#### 14.5.5.2 Stop3 Mode Operation

In Stop3 mode, clocks to the SCI module are halted.

No SCI module registers are affected in Stop3 mode.

The receive input active edge detect circuit is still active in Stop3 mode. An active edge on the receive input brings the CPU out of Stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from Stop3 mode. Software should ensure Stop3 mode is not entered while there is a character being transmitted out of or received into the SCI module.

#### 14.5.5.3 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.

#### 14.5.5.4 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.

### 14.5.6 Modes of Operation

The SCI is affected by the device mode of operation and the module clock can be disabled:

- For lower power, the SPI module clock can be disabled via the SCI control bit of the SCGC1 register
- LPRun - the reference oscillator is divided by 64 and all clocks are scaled by this factor. It is recommended to disable the SCI during this mode as all frequencies and timing changes accordingly. If not disabled, the SCI baud rate changes and normal baud rate and accuracy may not be possible.
- Wait modes - there are 2 wait modes
  - Wait - when entered, the CPU clock is disabled and the peripheral clock still runs at normal speed. In addition, the SCISWAI control bit in the SCI control SCIC1 register must be enabled to allow the SCI clock to run during Wait. If the SCI is a slave, it can still respond to a transfer and enable an interrupt request. No transmit transfers would occur as the CPU is disabled
    - Setting SCISWAI does not affect the state of the receiver enable bit RE, or the transmitter enable bit TE.
    - If SCISWAI is set, any transmission or reception in progress stops at wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of wait mode. Exiting wait mode by reset aborts any transmission or reception in progress and resets the SCI.
  - LPWait - when entered the the the CPU clock is disabled and the peripheral clock still runs but at a reduced speed. This mode should be disabled as the baud rate would be altered.
- The SPI is disabled in Stop3 mode, regardless of the settings before executing the STOP instruction.
  - During Stop3 mode, clocks to the SCI module are halted. No registers are affected. If Stop3 is exited with a reset, the SCI will be put into its reset state. If Stop3 is exited with an interrupt, the SCI continues from the state it was in when Stop3 was entered.
  - The RXEDGIF flag and related interrupt request is available for wake-up from low power.

## 14.6 Register Definition

This section provides a detailed description of all SCI registers.

The memory map for the SCI module is given below in Table 14-4. Only the SCIC3 register is located in the High-Page Register Map.

**Table 14-4. Module Memory Map**

| Address | Name | Access |
|---|---|---|
| 0x0038 | SCI Baud Rate Register High (SCI1BDH) | Read/Write |
| 0x0039 | SCI Baud Rate Register Low (SCI1BDL) | Read/Write |
| 0x003A | SCI Control Register1 (SCI1C1) | Read/Write |
| 0x003B | SCI Control Register 2 (SCI1C2) | Read/Write |
| 0x003C | SCI Status Register 1 (SCI1S1) | Read |
| 0x003D | SCI Status Register 2(SCI1S2) | Read/Write |
| 0x003E | SCI Control Register 3 (SCI1C3) | Read/Write |
| 0x003F | SCI Data Register (SCI1D) | Read/Write |
| 0x1804 | SCI Control Register 3 (SCI1C3) | Read/Write |

### 14.6.1 SCI Baud Rate Registers (SCI1BDH, SCI1BDL)

This pair of registers (SCI1BDH, SCI1BDL) controls the pre scale integer divisor SBR[12:0] for SCI baud rate generation. This divisor works in conjunction of the baud rate Fraction Adjust Counter (see Section 14.6.8, “SCI Control Register 4 (SCI1C4)”) to establish the SCI baud rate.

To update the 13-bit baud rate setting SBR[12:0], first write to SCI1BDH to buffer the high half of the new value and then write to SCI1BDL. The working value in SCI1BDH does not change until SCI1BDL is written. To calculate SBR[12:0] for a given baud rate, see Section 14.5.1, “Baud Rate Generator”.

SCI1BDL 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 SCI1C2 are written to 1).

**SCI1BDH** **0x0038**

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

▨ = Unimplemented or Reserved

**Figure 14-4. SCI Baud Rate Register High (SCI1BDH)**

Table 14-5. SCI1BDH Field Descriptions

| Field | Description |
|---|---|
| 7<br>LBKDIE | **LIN Break Detect Interrupt Enable (for LBKDIF)**<br>0 Hardware interrupts from LBKDIF disabled (use polling).<br>1 Hardware interrupt requested when LBKDIF flag is 1. |
| 6<br>RXEDGIE | **RxD Input Active Edge Interrupt Enable (for RXEDGIF)**<br>0 Hardware interrupts from RXEDGIF disabled (use polling).<br>1 Hardware interrupt requested when RXEDGIF flag is 1. |
| 4:0<br>SBR[12:8] | **Baud Rate Modulo Divisor** — The 13 bits in SBR[12:0] set the integer divisor for the baud rate generator.<br>• When SBR = 0, the SCI baud rate generator is disabled<br>• Valid SBR values are 1 to 8191 |

**SCI1BDL** **0x0x0039**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Figure 14-5. SCI Baud Rate Register Low (SCI1BDL)

Table 14-6. SCIxBDL Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>SBR[7:0] | **Baud Rate Modulo Divisor** — The 13 bits in SBR[12:0] set the integer divisor for the baud rate generator.<br>• When SBR = 0, the SCI baud rate generator is disabled<br>• Valid SBR values are 1 to 8191 |

## 14.6.2 SCI Control Register 1 (SCI1C1)

The SCI1C1 read/write register is used to control various optional features of the SCI system.

**SCI1C1** **0x003A**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 14-6. SCI Control Register 1 (SCI1C1)

### Table 14-7. SCI1C1 Field Descriptions

| Field | Description |
|---|---|
| 7<br>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.<br>0 Normal operation — RxD and TxD use separate pins.<br>1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See <st-blue>RSRC bit.) RxD pin is not used by SCI. |
| 6<br>SCISWAI | **SCI Stops in Wait Mode**<br>0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU.<br>1 SCI clocks freeze while CPU is in wait mode. |
| 5<br>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.<br>0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins.<br>1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input. |
| 4<br>M | **9-Bit or 8-Bit Mode Select**<br>0 Normal — start + 8 data bits (LSB first) + stop.<br>1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop. |
| 3<br>WAKE | **Receiver Wake up Method Select** — See Section 14.5.3.2, “Receiver Wake up Operation” for more information.<br>0 Idle-line wakeup.<br>1 Address-mark wakeup. |
| 2<br>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. See Section 14.5.3.2.1, “Idle-Line Wake up” for more information.<br>0 Idle character bit count starts after start bit.<br>1 Idle character bit count starts after stop bit. |
| 1<br>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.<br>0 No hardware parity generation or checking.<br>1 Parity enabled. |
| 0<br>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.<br>0 Even parity.<br>1 Odd parity. |

### 14.6.3 SCI Control Register 2 (SCI1C2)

The SCI1C2 read/write register is also used to control optional features of the SCI system.

| | SCI1C2 | | | | 0x003B | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-7. SCI Control Register 2 (SCI1C2)**

**Table 14-8. SCI1C2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TIE | **Transmit Interrupt Enable (for TDRE)**<br>0 Hardware interrupts from TDRE disabled (use polling).<br>1 Hardware interrupt requested when TDRE flag is 1. |
| 6<br>TCIE | **Transmission Complete Interrupt Enable (for TC)**<br>0 Hardware interrupts from TC disabled (use polling).<br>1 Hardware interrupt requested when TC flag is 1. |
| 5<br>RIE | **Receiver Interrupt Enable (for RDRF)**<br>0 Hardware interrupts from RDRF disabled (use polling).<br>1 Hardware interrupt requested when RDRF flag is 1. |
| 4<br>ILIE | **Idle Line Interrupt Enable (for IDLE)**<br>0 Hardware interrupts from IDLE disabled (use polling).<br>1 Hardware interrupt requested when IDLE flag is 1. |
| 3<br>TE | **Transmitter Enable**<br>0 Transmitter off.<br>1 Transmitter on.<br>TE must be 1 to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system.<br>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).<br>TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. See Section 14.5.2.1, “Send Break and Queued Idle” for more details.<br>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<br>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.<br>0 Receiver off.<br>1 Receiver on. |

Table 14-8. SCI1C2 Field Descriptions (continued)

| Field | Description |
|---|---|
| 1<br>RWU | **Receiver Wake up 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 wake up condition. The wake up condition is either an idle line between messages (WAKE = 0, idle-line wake up), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wake up). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. See Section 14.5.3.2, “Receiver Wake up Operation” for more details.<br>0 Normal SCI receiver operation.<br>1 SCI receiver in standby waiting for wakeup condition. |
| 0<br>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. See Section 14.5.2.1, “Send Break and Queued Idle” for more details.<br>0 Normal transmitter operation.<br>1 Queue break character(s) to be sent. |

## 14.6.4 SCI Status Register 1 (SCI1S1)

The read-only SCIS1 register has eight status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

SCI1S1 0x003C

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

▨ = Unimplemented or Reserved

Figure 14-8. SCI Status Register 1 (SCI1S1)

Table 14-9. SCI1S1 Field Descriptions

| Field | Description |
|---|---|
| 7<br>TDRE | **Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE = 1 and then write to the SCI data register (SCIxD).<br>0 Transmit data register (buffer) full.<br>1 Transmit data register (buffer) empty. |
| 6<br>TC | **Transmission Complete Flag** — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.<br>0 Transmitter active (sending data, a preamble, or a break).<br>1 Transmitter idle (transmission activity complete).<br>TC is cleared automatically by reading SCIxS1 with TC = 1 and then doing one of the following three things:<br>• Write to the SCI data register (SCIxD) to transmit new data<br>• Queue a preamble by changing TE from 0 to 1<br>• Queue a break character by writing 1 to SBK in SCIxC2 |

Table 14-9. SCI1S1 Field Descriptions (continued)

| Field | Description |
|---|---|
| 5<br>RDRF | **Receive Data Register Full Flag** — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). To clear RDRF, read SCIxS1 with RDRF = 1 and then read the SCI data register (SCIxD).<br>0 Receive data register empty.<br>1 Receive data register full. |
| 4<br>IDLE | **Idle Line Flag** — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn’t start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, read SCIxS1 with IDLE = 1 and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period.<br>0 No idle line detected.<br>1 Idle line was detected. |
| 3<br>OR | **Receiver Overrun Flag** — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with OR = 1 and then read the SCI data register (SCIxD).<br>0 No overrun.<br>1 Receive overrun (new SCI data lost). |
| 2<br>NF | **Noise Flag** — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD).<br>0 No noise detected.<br>1 Noise detected in the received character in SCIxD. |
| 1<br>FE | **Framing Error Flag** — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIxS1 with FE = 1 and then read the SCI data register (SCIxD).<br>0 No framing error detected. This does not guarantee the framing is correct.<br>1 Framing error. |
| 0<br>PF | **Parity Error Flag** — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIxS1 and then read the SCI data register (SCIxD).<br>0 No parity error.<br>1 Parity error. |

### 14.6.5 SCI Status Register 2 (SCI1S2)

The SCIS2 register has one read-only status flag and several control bits.

SCI1S2 0x003D

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

▨ = Unimplemented or Reserved

**Figure 14-9. SCI Status Register 2 (SCI1S2)**

**Table 14-10. SCI1S2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>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.<br>0 No LIN break character has been detected.<br>1 LIN break character has been detected. |
| 6<br>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.<br>0 No active edge on the receive pin has occurred.<br>1 An active edge on the receive pin has occurred. |
| 4<br>RXINV[1] | **Receive Data Inversion** — Setting this bit reverses the polarity of the received data input.<br>0 Receive data not inverted<br>1 Receive data inverted |
| 3<br>RWUID | **Receive Wake Up Idle Detect**— RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit.<br>0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character.<br>1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. |
| 2<br>BRK13 | **Break Character Generation Length** — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit.<br>0 Break character is transmitted with length of 10 bit times (11 if M = 1)<br>1 Break character is transmitted with length of 13 bit times (14 if M = 1) |
| 1<br>LBKDE | **LIN Break Detection Enable**— LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting.<br>0 Break character is detected at length of 10 bit times (11 if M = 1).<br>1 Break character is detected at length of 11 bit times (12 if M = 1). |
| 0<br>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.<br>0 SCI receiver idle waiting for a start bit.<br>1 SCI receiver active (RxD input not idle). |

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

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

### 14.6.6 SCI Control Register 3 (SCI1C3)

The SCI1C3 read/write register is also used to control optional features of the SCI system as well as provide the 9th receiver bit when 9-bit operation is enabled.

**SCI1C3** **0x003E**

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

▨ = Unimplemented or Reserved

**Figure 14-10. SCI Control Register 3 (SCI1C3)**

**Table 14-11. SCI1C3 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>R8 | **Ninth Data Bit for Receiver** — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCIxD register. When reading 9-bit data, read R8 before reading SCIxD because reading SCIxD completes automatic flag clearing sequences which could allow R8 and SCIxD to be overwritten with new data. |
| 6<br>T8 | **Ninth Data Bit for Transmitter** — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written. |
| 5<br>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.<br>0 TxD pin is an input in single-wire mode.<br>1 TxD pin is an output in single-wire mode. |
| 4<br>TXINV[1] | **Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output.<br>0 Transmit data not inverted<br>1 Transmit data inverted |
| 3<br>ORIE | **Overrun Interrupt Enable** — This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0 OR interrupts disabled (use polling).<br>1 Hardware interrupt requested when OR = 1. |
| 2<br>NEIE | **Noise Error Interrupt Enable** — This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0 NF interrupts disabled (use polling).<br>1 Hardware interrupt requested when NF = 1. |

**Table 14-11. SCI1C3 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FEIE | **Framing Error Interrupt Enable** — This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0 FE interrupts disabled (use polling).<br>1 Hardware interrupt requested when FE = 1. |
| 0<br>PEIE | **Parity Error Interrupt Enable** — This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0 PF interrupts disabled (use polling).<br>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.

## 14.6.7 SCI Data Register (SCI1D)

This register location actually accesses two separate registers:

- Reads return the contents of the read-only receive data buffer
- 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.

SCI1D 0x003F

| | 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 14-11. SCI Data Register (SCI1D)**

**Table 14-12. SCI1D Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>R[7:0] | **Read Data -** This is the read byte from the receiver buffer. When 9-bit operation is enabled, the 9th MSB bit is read from SCIC3, Bit 7. |
| 7:0<br>W[7:0] | **Write Data -** This is the write byte to the transmit buffer. When 9-bit operation is enabled, the 9th MSB bit is written to SCIC3, Bit 6. |

## 14.6.8 SCI Control Register 4 (SCI1C4)

SCIC4 is the only SCI control register located in the High-Page Map and controls the fractional adjust counter of the SCI baud rate generator. This register works in conjunction of the baud rate integer counter (see Section 14.6.1, “SCI Baud Rate Registers (SCI1BDH, SCI1BDL)”) to establish the SCI baud rate.

To calculate BRFA[4:0] for a given baud rate, see Section 14.5.1, “Baud Rate Generator”.

| SCI1C4 | | | | | 0x1804 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | 0 | BRFA[4:0] | | | | |
| W | ▨ | ▨ | ▨ | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 14-12. SCI Control Register 4 (SCI1C4)**

**Table 14-13. SCI1C4 Field Description**

| Field | Description |
|---|---|
| 4:0<br>BRFA[4:0] | **Baud Rate Fraction Adjust** — The 5 bits in BRFA[4:0] program the fractional adjust of the SCI baud rate generator. See Table 14-2 for field value versus fractional divisor |

# Chapter 15
# Serial Peripheral Interface (SPI) Module

## 15.1 Introduction

The SPI module is a synchronous serial data input/output port used for interfacing with serial memories, peripheral devices, or other processors. The SPI allows an 8-bit serial bit stream to be shifted simultaneously into and out of the device at a programmed bit-transfer rate (called 4-wire mode). There are four pins associated with the SPI port; SPCLK (SPSCK), MOSI, MISO, and SS.

The SPI module can be programmed for master or slave operation. It also supports a 3-wire mode where for master mode the MOSI becomes MOMI, a bidirectional data pin, and for slave mode the MISO becomes SISO, a bidirectional data pin. In 3-wire mode, data is only transferred in one direction at a time.

The SPI bit clock is derived from the bus clock with a maximum 16 MHz operation. A programmable prescaler (maximum divide-by-8) drives a second baud rate programmable divider (maximum divide-by-256) to develop the bit clock. The maximum SPI transfer rate is 8 MHz.

## 15.2 Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- 8-Bit only transfer size
- Programmable transmit bit rate (8 MHz max)
- Double-buffered transmit and receive
- Serial clock phase and polarity options (supports all 4 options)
- Optional slave select output
- Selectable MSB-first or LSB-first shifting
- Selectable MSB-first or LSB-first shifting

# 15.3 Block Diagrams

## 15.3.1 SPI System Block Diagram

This section shows a typical simple SPI system level 4-wire block diagram.

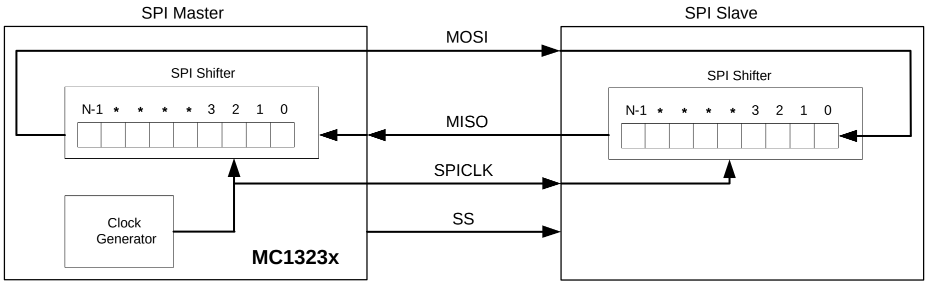


**Figure 15-1. SPI System Block Diagram**

Figure 15-1 shows the MC13234/MC13237 as the SPI master and an external device as a SPI slave. 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. The SPI interface supports simultaneous data exchange between master and slave, although some devices may be read or write only. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be enabled by a low level on the slave select input.

## 15.3.2 SPI Module Block Diagram

Figure 15-2 shows the block diagram of SPI module.

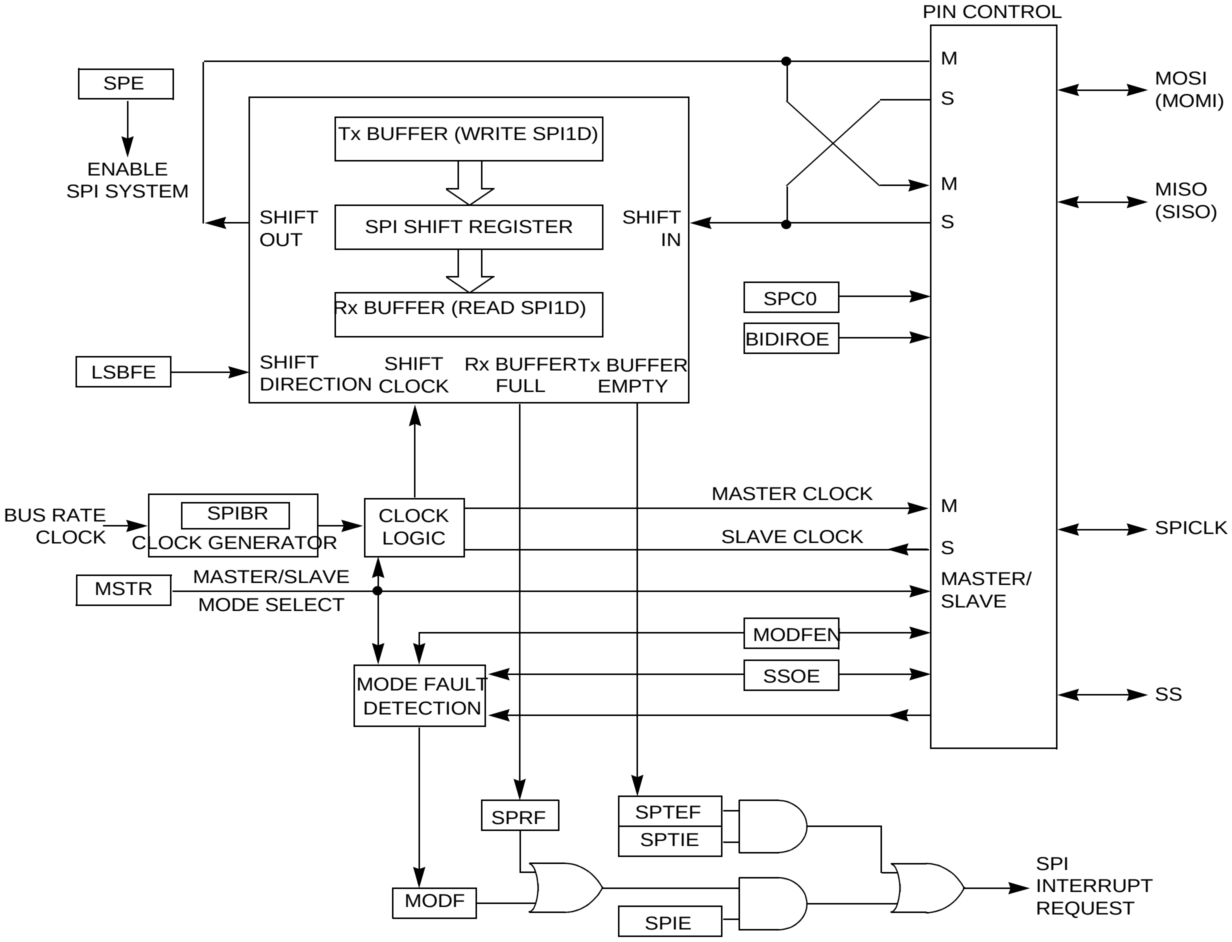


Figure 15-2. SPI Module Block Diagram

## 15.4 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 15.4.1 SPICLK — SPI Serial Clock (SPSCK)

When the SPI is enabled as a slave, this pin, also known as SPSCK, is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 15.4.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 15.4.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 15.4.4 $\overline{\text{SS}}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

## 15.5 Functional Description

The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPI1D) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPI1D). Pin multiplexing logic controls connections between MCU pins and the SPI module.

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

When the SPI is configured as a slave, the SPICLK 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 MOSI1 pin.

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

### 15.5.1 Bit Rate Generation

As shown in Figure 15-3, the clock source for the SPI bitrate generator is the bus clock. The three prescale bits (SPPR[2:0]) select a pre scale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR[2:0]) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI Master Mode bitrate clock. With a maximum bus clock rate of 16 MHz and prescaler divisor = 1 and clock rate divider = 2, the SPI maximum bit rate is 8 MHz.

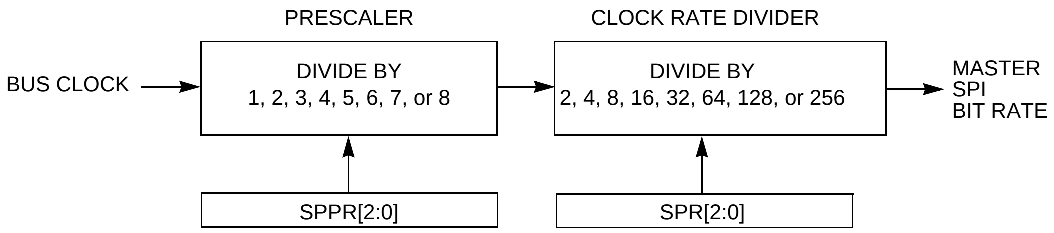


Figure 15-3. MCU SPI Bit Rate Generation

## 15.5.2 Basic Transfer

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 SPSCK edge and shifted, changing the bit value on the MOSI pin, one-half SPSCK cycle later. After eight SPSCK 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.

When the SPI is configured as a slave, its $\overline{\text{SS}}$ pin must be driven low before a transfer starts and $\overline{\text{SS}}$ must stay low throughout the transfer. If a clock format where CPHA = 0 is selected, $\overline{\text{SS}}$ must be driven to a logic 1 between successive transfers. If CPHA = 1, $\overline{\text{SS}}$ may remain low between successive transfers. See Section 15.5.3, “SPI Clock Formats” for more details.

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.

## 15.5.3 SPI Clock Formats

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

Figure 15-4 shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{\text{SS}}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{\text{SS}}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{\text{SS}}$ IN waveform applies to the slave select input of a slave.

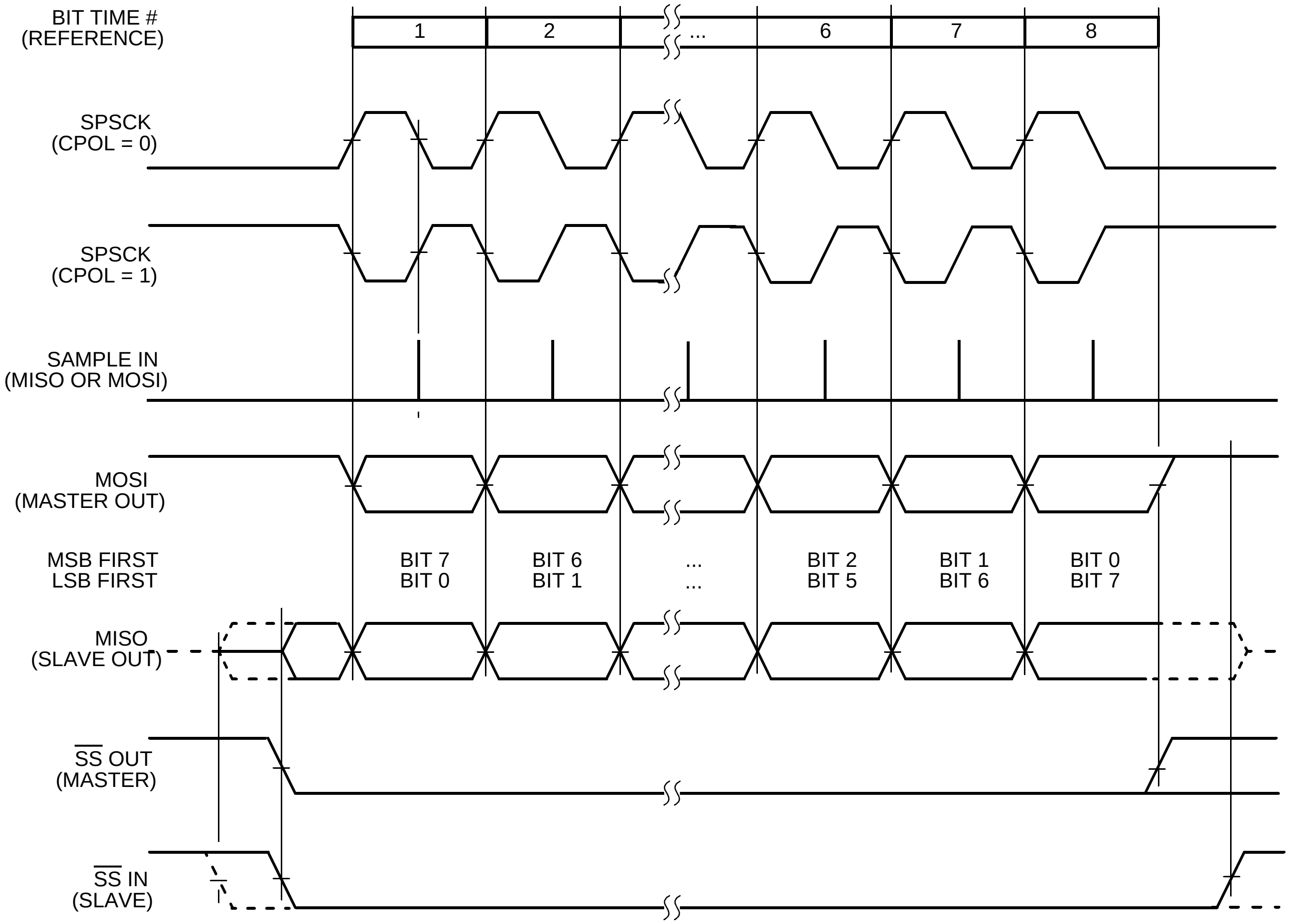


**Figure 15-4. SPI Clock Formats (CPHA = 1)**

When CPHA = 1, the slave begins to drive its MISO output when $\overline{\text{SS}}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's SS input is not required to go to its inactive high level between transfers.

Figure 15-5 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected (SS IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The SS OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master SS output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The SS IN waveform applies to the slave select input of a slave.

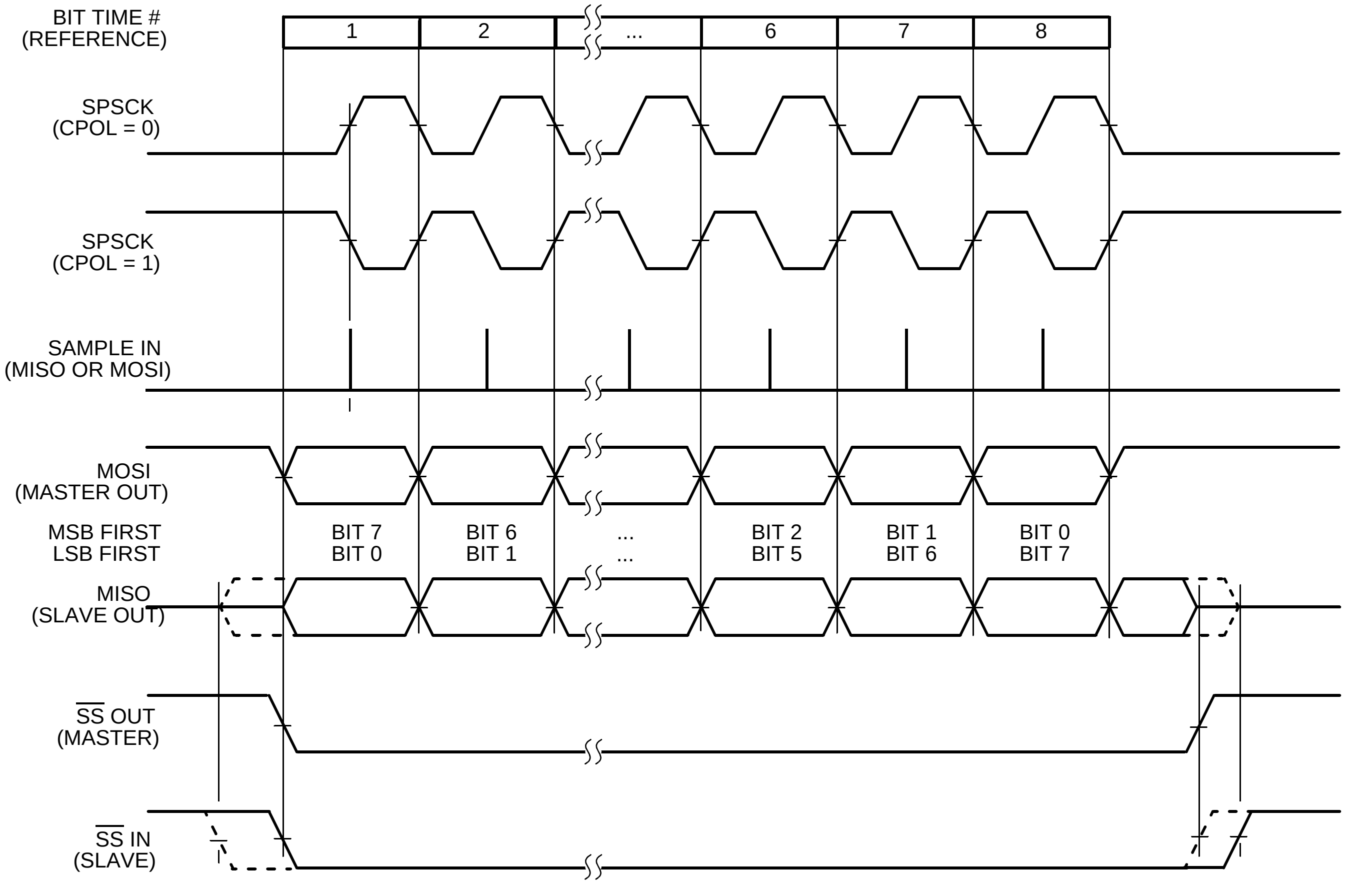


**Figure 15-5. SPI Clock Formats (CPHA = 0)**

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when SS goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{\text{SS}}$ input must go to its inactive high level between transfers.

### 15.5.4 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).

The SPI uses Interrupt Vector No. 17.

### 15.5.5 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 $\overline{\text{SS}}$ pin (provided the $\overline{\text{SS}}$ pin is configured as the mode fault input signal). The $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

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

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

## 15.6 Modes of Operation

The SPI is affected by the device mode of operation and the module clock can be disabled:

- For lower power, the SPI module clock can be disabled via the SPI control bit of the SCGC2 Register
- LPRun - the reference oscillator is divided by 64 and all clocks are scaled by this factor. The SPI can run but the SPI clock and timing changes during this mode accordingly.
- Wait modes - there are two wait modes

- Wait - when entered, the CPU clock is disabled and the bus clock still runs at normal speed. In addition, the SPISWAI control bit in the SCI control SPIC2 register must be enabled to allow the SCI clock to run during Wait. If the SPI is a slave, it can still respond to a transfer and enable an interrupt request. No master transfers would occur as the CPU is disabled
- LPWait - when entered, the CPU clock is disabled and the bus clock still runs but at a reduced speed. In addition, the SPISWAI control bit in the SCI control SPIC2 register must be enabled to allow the SCI clock to run during Wait. If the SPI is a slave, it can still respond to a transfer and enable an interrupt request. No master transfers would occur as the CPU is disabled.

- The SPI is disabled in Stop3 mode, regardless of the settings before executing the STOP instruction.
  - During Stop3 mode, clocks to the SPI module are halted. No registers are affected. If Stop3 is exited with a reset, the SPI will be put into its reset state. If Stop3 is exited with an interrupt, the SPI continues from the state it was in when Stop3 was entered.
- Background Mode Operation -When the microcontroller is in active background mode, the SPI temporarily suspends all counting until the microcontroller returns to normal user mode.

## 15.7 Register Definition

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data. The SPI registers are all located in the Direct-Page Register Map.

### 15.7.1 SPI Control Register 1 (SPI1C1)

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

SPI1C1 0x0070

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 15-6. SPI Control Register 1 (SPI1C1)**

**Table 15-1. SPI1C1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIE | **SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events.<br>0 Interrupts from SPRF and MODF inhibited (use polling)<br>1 When SPRF or MODF is 1, request a hardware interrupt |
| 6<br>SPE | **SPI System Enable** — Disabling the SPI halts any transfer that is in progress and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty.<br>0 SPI system inactive<br>1 SPI system enabled |

**Table 15-1. SPI1C1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>SPTIE | **SPI Transmit Interrupt Enable** — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF).<br>0 Interrupts from SPTEF inhibited (use polling)<br>1 When SPTEF is 1, hardware interrupt requested |
| 4<br>MSTR | **Master/Slave Mode Select**<br>0 SPI module configured as a slave SPI device<br>1 SPI module configured as a master SPI device |
| 3<br>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. See Section 15.5.3, “SPI Clock Formats” for more details.<br>0 Active-high SPI clock (idles low)<br>1 Active-low SPI clock (idles high) |
| 2<br>CPHA | **Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. See Section 15.5.3, “SPI Clock Formats” for more details.<br>0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer<br>1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer |
| 1<br>SSOE | **Slave Select Output Enable** — This bit is used in combination with the mode fault enable (MODFEN) bit in SPI1C2 and the master/slave (MSTR) control bit to determine the function of the $\overline{\text{SS}}$ pin as shown in Table 15-2. |
| 0<br>LSBFE | **LSB First (Shifter Direction)**<br>0 SPI serial data transfers start with most significant bit<br>1 SPI serial data transfers start with least significant bit |

**Table 15-2. $\overline{\text{SS}}$ 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 | $\overline{\text{SS}}$ input for mode fault | Slave select input |
| 1 | 1 | Automatic $\overline{\text{SS}}$ output | Slave select input |

## 15.7.2 SPI Control Register 2 (SPI1C2)

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

| SPI1C2 | | | | | 0x0071 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| W | ▨ | ▨ | ▨ | | | ▨ | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 15-7. SPI Control Register 2 (SPI1C2)**

**Table 15-3. SPI1C2 Register Field Descriptions**

| Field | Description |
|---|---|
| 4<br>MODFEN | **Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{\text{SS}}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{\text{SS}}$ pin is used (see Table 15-2 for more details).<br>0 Mode fault function disabled, master $\overline{\text{SS}}$ pin reverts to general-purpose I/O not controlled by SPI<br>1 Mode fault function enabled, master $\overline{\text{SS}}$ pin acts as the mode fault input or the slave select output |
| 3<br>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.<br>0 Output driver disabled so SPI data I/O pin acts as an input<br>1 SPI I/O pin enabled as an output |
| 1<br>SPISWAI | **SPI Stop in Wait Mode**<br>0 SPI clocks continue to operate in wait mode<br>1 SPI clocks stop when the MCU enters wait mode |
| 0<br>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.<br>0 SPI uses separate pins for data input and data output<br>1 SPI configured for single-wire bidirectional operation |

## 15.7.3 SPI Baud Rate Register (SPI1BR)

This register is used to set the prescaler and bit rate divisor for an SPI master. See Section 15.5.1, “Bit Rate Generation”.

**SPI1BR** **0x0072**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| W | ▨ | | | | ▨ | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Table 15-4. SPI1BR Register Field Descriptions**

| Field | Description |
|---|---|
| 6:4<br>SPPR[2:0] | **SPI Baud Rate Pre scale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 15-5. 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 15-3). |
| 2:0<br>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 Table 15-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 15-3). The output of this divider is the SPI bit rate clock for master mode. |

Table 15-5. SPI Baud Rate Prescaler Divisor

| SPPR[2:0] | Prescaler Divisor |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 3 |
| 011 | 4 |
| 100 | 5 |
| 101 | 6 |
| 110 | 7 |
| 111 | 8 |

Table 15-6. SPI Baud Rate Divisor

| SPR[2:0] | Rate Divisor |
|---|---|
| 000 | 2 |
| 001 | 4 |
| 010 | 8 |
| 011 | 16 |
| 100 | 32 |
| 101 | 64 |
| 110 | 128 |
| 111 | 256 |

## 15.7.4 SPI Status Register (SPI1S)

This read-only register has provides the SPI status bits.

| | SPI1S | | | | 0x0073 | | | |
|---|---|---|---|---|---|---|---|---|
| | 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 15-8. SPI Status Register (SPI1S)

Table 15-7. SPI1S Register Field Descriptions

| Field | Description |
|---|---|
| 7<br>SPRF | **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.<br>0 No data available in the receive data buffer<br>1 Data available in the receive data buffer |
| 5<br>SPTEF | **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 shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter.<br>0 SPI transmit buffer not empty<br>1 SPI transmit buffer empty |
| 4<br>MODF | **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).<br>0 No mode fault error<br>1 Mode fault error detected |

## 15.7.5 SPI Data Register (SPI1D)

The read/write SPI1D register is used to write data to the transmitter and read data from the receiver.

- 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.

| SPI1D | | | | | 0x0075 | | | |
|---|---|---|---|---|---|---|---|---|
| | 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 15-9. SPI Data Register (SPI1D)**

**Table 15-8. SPI1D Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>R[7:0] | **Read Data -** This is the read byte from the SPI receiver buffer. |
| 7:0<br>W[7:0] | **Write Data -** This is the write byte to the SPI transmit buffer. |

# Chapter 16
# Timer/PWM (TPM Module)

## 16.1 Introduction

The MC13234/MC13237 has four independent timer/PWM modules, each with one channel. Each TPM module is based on a 16-bit counter and provides input capture, output compare, and pulse width modulation. Each TPM module has one associated I/O pin for input capture or counter/PWM output. Each TPM has an independent set of control/status registers.

## 16.2 Features

Each TPM has the following features:

- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM)
- Clock sources independently selectable per TPM
- Selectable clock sources
  - Bus clock
  - Optional 32.768 kHz oscillator
  - 32 MHz reference oscillator divided-by 1024 (31,250 Hz)
- Clock prescaler taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Module enable
- One interrupt request per channel and a terminal count interrupt request for each TPM module
- Channel features:
  - The channel may be input capture, output compare, or buffered 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

### 16.2.1 Block Diagram

Figure 16-1 is a block diagram of a single channel TPM module.

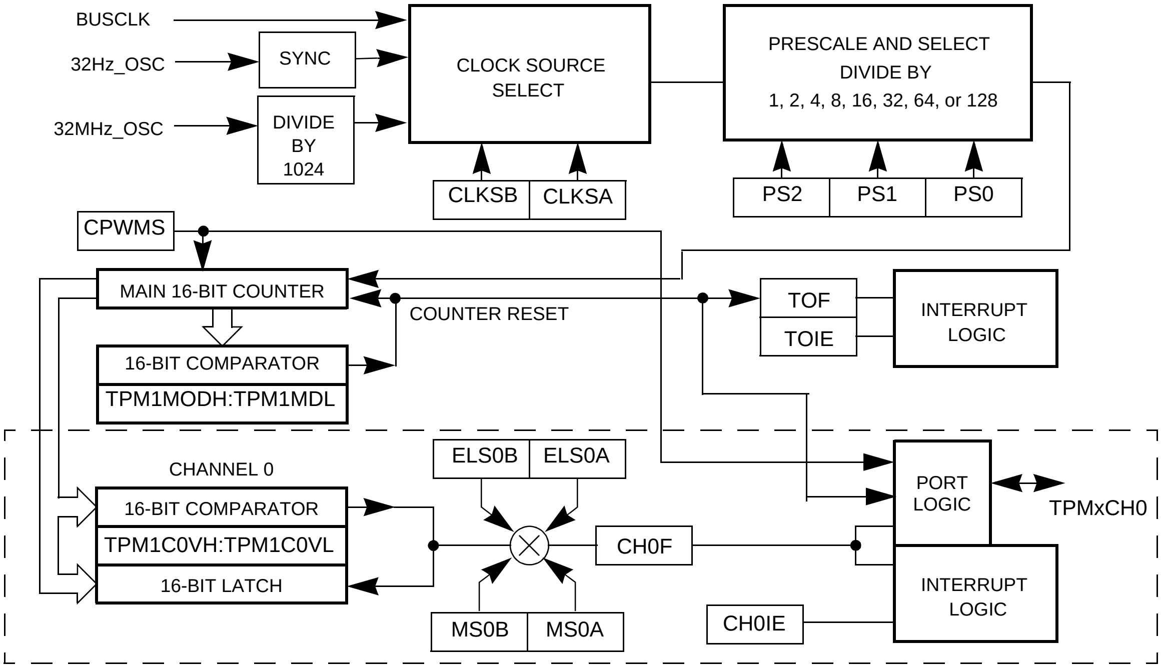


**Figure 16-1. TPM Module Block Diagram**

### 16.2.2 External Signals (TPM3:TPM0)

The are 4 timer I/O signals:

- TPM0 - associated with Channel 0, TPM1
- TPM1 - associated with Channel 0, TPM2
- TPM2 - associated with Channel 0, TPM3
- TPM3 - associated with Channel 0, TPM4

The function of any pin depends on the configuration of its channel:

- If no pin function is needed, the pin reverts to being controlled by the general-purpose port I/O controls
- When used as an input, the TPMx I/O can be used for input capture or timer gating. It cannot be used as a clock source to the TPM counter.
- The TPMx I/O may also be used as an output for various PWM and counter modes.

## 16.3 Functional Description

As shown in Figure 16-1, all TPM functions are associated with a main 16-bit counter that allows flexible selection of its clock source and pre scale divisor. A 16-bit modulo register also is associated with the main 16-bit counter in the TPM for comparator functions.

Each TPM has a single timing channel, with an associated external I/O pin, that supports center-aligned pulse-width-modulation (PWM), input event capture, output compare, and edge-aligned PWM mode.

Each TPM is completely independent and has its own external I/O pin and control registers

## 16.3.1 TPM Counter Clock Source

As seen in Figure 16-1, the main 16-bit counter has a programmable clock source. The primary source is selected by a multiplexer:

- The MC13234/MC13237 bus clock (peripheral clock) - the CPU and bus clock (equal to 1/2 CPU clock) are programmable
- An optional onboard 32.768 kHz oscillator - an external 32.768 kHz crystal must be used and the oscillator enabled
- The 32 MHz reference oscillator divided-by-1024 (31,250 Hz)

The multiplexer is controlled by the CLKSB:CLKSA Bits of Register TPMxSC, and this field can also be used to disable the TPMx module by selecting no clock. Table 16-1 shows the coding for the TPM clock source selection.

**Table 16-1. TPM Clock Source Selection (CLKSB:CLKSA)**

| CLKSB:CLKSA | TPM Clock Source to Prescaler Input |
|---|---|
| 0:0 | No clock selected (TPM disabled) |
| 0:1 | Bus rate clock (BUSCLK) |
| 1:0 | Optional 32.768 kHz oscillator |
| 1:1 | Reference oscillator divided-by-1024 (31,250 Hz) |

The selected clock source, in turn, drives a prescaler that feeds the counter directly. The prescaler is programmable for divide ratios from 1 to 128 and is controlled by the PS[2:0] field of Register TPMxSC. Table 16-2 shows the coding for the prescaler divide ratio selection.

**Table 16-2. TPM Clock Prescale Divisor Selection (PS[2:0])**

| PS[2:0] | TPM Clock Prescaler Divide Ratio |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 4 |
| 011 | 8 |
| 100 | 16 |
| 101 | 32 |
| 110 | 64 |
| 111 | 128 |

## 16.3.2 Counter

All timer functions are based on the main 16-bit counter (TPM1CNTH:TPM1CNTL). This section discusses up-counting vs. up-/down-counting, end-of-count overflow, and manual counter reset.

After any device reset, CLKSB:CLKSA = 0:0 so no clock source is selected and the TPM is inactive. Typically, CLKSB:CLKSA are set to 0:1 so the bus clock drives the timer counter.

The main 16-bit counter has two counting modes:

- When center-aligned PWM is selected (CPWMS = 1), the counter operates in up-/down-counting mode. The counter counts upward from 0x0000 through its terminal count and then counts downward to 0x0000 where it returns to up-counting. The terminal count is 0xFFFF or a modulus value in TPM1MODH:TPM1MODL registers, depending on programming.
- Otherwise (CPWMS = 0), the counter operates as a simple up-counter. As an up-counter, the main 16-bit counter counts from 0x0000 through its terminal count and then rolls over to 0x0000. The terminal count is 0xFFFF or a modulus value in TPM1MODH:TPM1MODL registers, depending on programming.

The timer overflow status flag (TOF) is associated with the main 16-bit counter and is an indication that the timer counter has overflowed. An interrupt request enable bit (TOIE) allows an IRQ to automatically be generated whenever the TOF flag is 1. The conditions that cause TOF to become set depend on the counting mode (up or up/down):

- Up-counting mode -
  - If a modulus limit is not set - the 16-bit 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.
  - If a modulus limit is set - the 16-bit counter counts from 0x0000 through to the value set in the modulus register and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from the value set in the modulus register to 0x0000.
- Up-/down-counting mode - the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

Because the CPU is an 8-bit architecture, a coherency mechanism is built into the timer counter for read operations. Whenever either byte of the counter is read (TPMxCNTH or TPMxCNTL), both bytes are captured into a buffer so when the other byte is read, the value will represent the other byte of the count at the time the first byte was read. The counter continues to count normally, but no new value can be read from either byte until both bytes of the old count have been read.

The main timer counter can be reset manually at any time by writing any value to either byte of the timer count TPM1CNTH or TPM1CNTL. Resetting the counter in this manner also resets the coherency mechanism in case only one byte of the counter was read before resetting the count.

## 16.3.3 Channel Mode Selection

The MC13234/MC13237 has only one channel per TPM module, i.e., Channel 1. Provided CPWMS = 0 (center-aligned PWM operation is not specified), the MS0B and MS0AMS0A control bits in the channel status and control registers determine the basic mode of operation for Channel 0. Mode choices include input capture, output compare, and buffered edge-aligned PWM.

A single channel status flag is used for any of the channel modes.

### 16.3.3.1 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 (TPMxC0VH:TPMxC0VL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either byte of the 16-bit capture register is read, both bytes are latched into a buffer to support coherent 16-bit accesses regardless of order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxC0SC).

An input capture event sets a flag bit (CH0F) that can optionally generate an interrupt request.

### 16.3.3.2 Output Compare Mode

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

In output compare mode, values are transferred to the corresponding timer channel value registers only after both 8-bit bytes of a 16-bit register have been written. This coherency sequence can be manually reset by writing to the channel status/control register (TPMxC0SC).

An output compare event sets a flag bit (CH0F) that can optionally generate a CPU interrupt request.

### 16.3.3.3 Edge-Aligned PWM Mode

Edge-aligned PWM output mode uses the normal up-counting mode of the timer counter (CPWMS = 0). The period of this PWM signal is determined by the setting in the modulus register (TPM1MODH:TPM1MODL). The duty cycle is determined by the setting in the timer channel value register (TPMxC0VH:TPMxC0VL). The polarity of this PWM signal is determined by the setting in the ELS0A control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As Figure 16-2 shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. The time between the modulus overflow and the output compare is the pulse width. If ELS0A = 0, the counter overflow forces the PWM signal high and the output compare forces the PWM signal low. If ELS0A = 1, the counter overflow forces the PWM signal low and the output compare forces the PWM signal high.

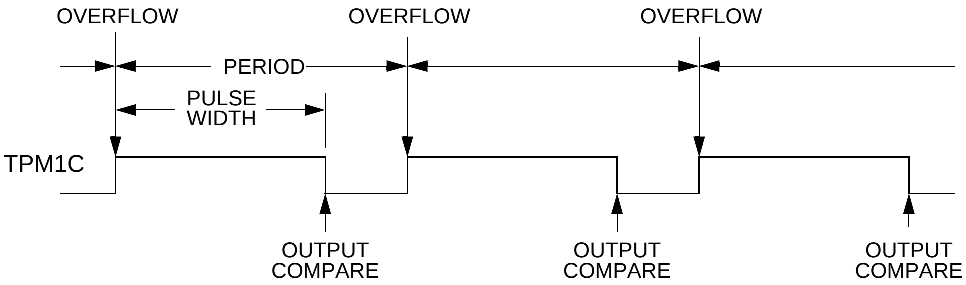


**Figure 16-2. PWM Period and Pulse Width (ELS0A = 0)**

When the channel value register is set to 0x0000, the duty cycle is 0 percent. By setting the timer channel value register (TPMxC0VH:TPMxC0VL) to a value greater than the modulus setting, a 100% duty cycle can be achieved. This implies that the modulus setting must be less than 0xFFFF to get 100% duty cycle.

Because the CPU is an 8-bit architecture, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to either register, TPMxC0VH or TPMxC0VL, write to buffer registers. In edge-PWM mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the value in the TPMxCNTH:TPMxCNTL counter is 0x0000. (The new duty cycle does not take effect until the next full period.)

## 16.3.4 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 TPMxC0VH:TPMxC0VL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELS0A will determine the polarity of the CPWM output.

$$\textbf{pulse width = 2 x (TPMxC0VH:TPMxC0VL)} \qquad \textit{Eqn. 16-1}$$

$$\begin{gathered}\textbf{period = 2 x (TPMxMODH:TPMxMODL);}\\ \text{for TPMxMODH:TPMxMODL = 0x0001–0x7FFF}\end{gathered} \qquad \textit{Eqn. 16-2}$$

If the channel value register TPMxC0VH:TPMxC0VL is zero or negative (Bit 15 set), the duty cycle will be 0%. If TPMxC0VH:TPMxC0VL is a positive value (Bit 15 clear) and is greater than the (nonzero) 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 generation of 100% duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

TPMxC0VH:TPMxC0VL = 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 to change directions from up-counting to down-counting.

Figure 16-3 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If ELS0A = 0, the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

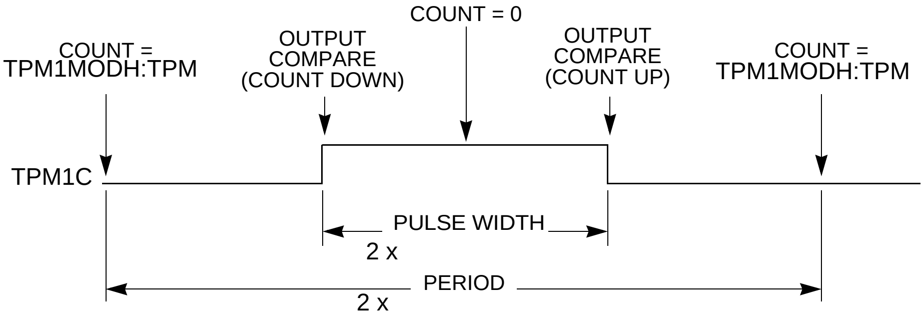


**Figure 16-3. CPWM Period and Pulse Width (ELS0A = 0)**

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Because the CPU is an 8-bit architecture, 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, TPMxMODH, TPMxMODL, TPMxC0VH, and TPMxC0VL, actually write to buffer registers. Values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the timer counter overflows (reverses direction from up-counting to down-counting at the end of the terminal count in the modulus register). This TPMxCNT overflow requirement only applies to PWM channels, not output compares.

Optionally, when TPMxCNTH:TPMxCNTL = TPMxMODH:TPMxMODL, the TPM can generate a TOF interrupt at the end of this count. The user can choose to reload any number of the PWM buffers, and they will all update simultaneously at the start of a new period.

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL 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 TPMxC0VH:TPMxC0VL.

## 16.4 TPM Interrupts

Each TPM has two interrupt request sources:

- The main counter overflow status (TOF)
- The Channel 0 status flag (CH0F) - the meaning of channel status depends on the mode of operation for the channel.
  - If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized.
  - If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

For each interrupt request source in the TPM, a flag bit is set on recognition of the interrupt condition such as timer overflow, channel input capture, or output compare events. This flag may be read (polled) by software to verify that the action has occurred, or an associated enable bit (TOIE or CH0IE) can be set to enable hardware interrupt request generation. While the interrupt enable bit is set, a static interrupt request will be generated whenever the associated interrupt flag equals 1. It is the responsibility of user software to perform a sequence of steps to clear the interrupt flag before returning from the interrupt service routine.

### 16.4.1 Clearing Timer Interrupt Flags

TPM interrupt flags are cleared by a 2-step process that includes a read of the flag bit while it is set (1) followed by a write of 0 to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

### 16.4.2 Timer Overflow Interrupt Description

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the 16-bit timer counter counts from 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 counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

### 16.4.3 Channel Event Interrupt Description

The meaning of channel interrupts depends on the current mode of the channel (input capture, output compare, edge-aligned PWM, or center-aligned PWM).

When a channel is configured as an input capture channel, the ELS0B:ELS0A control bits select rising edges, falling edges, any edge, or no edge (off) as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the 2-step sequence described in Section 16.4.1, “Clearing Timer Interrupt Flags”.

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the 2-step sequence described in Section 16.4.1, “Clearing Timer Interrupt Flags”.

### 16.4.4 PWM End-of-Duty-Cycle Events

For the channel that is configured for PWM operation, there are two possibilities:

- When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period.
- When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle, which are the times when the timer counter matches the channel value register.

The flag is cleared by the 2-step sequence described in Section 16.4.1, “Clearing Timer Interrupt Flags”.

## 16.5 Modes of operation

The TPM is affected by the device mode of operation and the module clock can be disabled:

- For lower power, the TPM module clock can be disabled via the KBIx control bit of the SCGC1 Register
- LPRun - the reference oscillator is divided by 64 and all clocks are scaled by this factor. It is recommended to disable the TPM during this mode as all frequencies and timing changes accordingly.
- Wait modes - there are two wait modes
  - Wait - when entered, the CPU clock is disabled and the bus clock still runs at normal speed . The TPM, if enabled, will continue to operate normally. However, there will be no new codes or changes of operation while in wait mode, because the CPU is not operating.
  - LP Wait - when entered, the CPU clock is disabled and the bus clock still runs but at a reduced speed. The TPM, if enabled, will continue to operate, however, all timing would be incorrect. Operation is not recommended.
- The TPM is disabled in Stop3 mode, regardless of the settings before executing the STOP instruction.
  - During Stop3 mode, clocks to the TPM module are halted. No registers are affected. If Stop3 is exited with a reset, the TPM will be put into its reset state. If Stop3 is exited with an interrupt, the TPM will resume operation. It is recommended that the TPM be halted before entering Stop3.
- Background Mode Operation -When the microcontroller is in active background mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user mode. When background mode is active, the timer counter and the coherency mechanism are frozen such that the buffer latches remain in the state they were in when the background mode became active even if one or both bytes of the counter are read while background mode is active..

## 16.6 Register Definition

The TPM includes:

- An 8-bit status and control register (TPMxSC)
- A 16-bit counter (TPMxCNTH:TPMxCNTL)
- A 16-bit modulo register (TPMxMODH:TPMxMODL)

Timer Channel 0 has:

- An 8-bit status and control register (TPMxC0SC)
- A 16-bit channel value register (TPMxC0VH:TPMxC0VL)

The MC13234/MC13237 has four TPM modules designated as TPM1, TPM2, TPM3, and TPM4. Table 16-3 lists the TPM module base addresses; only TPM4 is in the High-Page Register Map.The following register descriptions are for a single TPM module.

Table 16-3. TPM Register Base Addresses

| TPM Module | Base Address |
|---|---|
| TPM1 | 0x0008 |
| TPM2 | 0x0010 |
| TPM3 | 0x0018 |
| TPM4 | 0x1860 |

The following register descriptions are for a single TPM module.

## 16.6.1 Timer x Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.

TPMxSC Base + 0x00

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

▨ = Unimplemented or Reserved

Figure 16-4. Timer x Status and Control Register (TPMxSC)

Table 16-4. TPMxSC Register Field Descriptions

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — This flag is set when the TPM counter changes to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value. Clear TOF by reading the TPM status and control register when TOF is set and then writing a 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. Reset clears TOF. Writing a 1 to TOF has no effect.<br>0 TPM counter has not reached modulo value or overflow<br>1 TPM counter has overflowed |
| 6<br>TOIE | **Timer Overflow Interrupt Enable** — This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals 1. Reset clears TOIE.<br>0 TOF interrupts inhibited (use software polling)<br>1 TOF interrupts enabled |

Table 16-4. TPMxSC Register Field Descriptions (continued)

| Field | Description |
|---|---|
| 5<br>CPWMS | **Center-Aligned PWM Select** — This read/write bit selects CPWM operating mode. Reset clears this bit so the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up-/down-counting mode for CPWM functions. Reset clears CPWMS.<br>0 All TPM1 channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MS0B:MS0A control bits in each channel's status and control register<br>1 All TPM1 channels operate in center-aligned PWM mode |
| 4:3<br>CLKS[B:A] | **Clock Source Select** — This 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. See Section 16.3.1, "TPM Counter Clock Source" and Table 16-1 |
| 2:0<br>PS[2:0] | **Pre scale Divisor Select** — This 3-bit field selects one of eight divisors for the TPM counter prescaler. This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system. See Section 16.3.1, "TPM Counter Clock Source" and Table 16-2 |

## 16.6.2 Timer x Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This allows coherent 16-bit reads in either order. The coherency mechanism is automatically restarted by an MCU reset, a write of any value to TPMxCNTH or TPMxCNTL, or any write to the timer status/control register (TPM1SC).

Reset clears the TPM counter registers.

| | TPMxCNTH | | | | Base + 0x01 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| W | Any write to TPMxCNTH clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 16-5. Timer x Counter Register TPMxCNTH

| | TPMxCNTL | | | | Base + 0x02 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| W | Any write to TPMxCNTL clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 16-6. Timer x Counter Register TPMxCNTL

Table 16-5. TPMxCNTH and TPMxCNTL Register Field Descriptions

| Field | Description |
|---|---|
| CNTH[7:0]<br>CNT[15:8] | **Timer Count Value [15:8] -** Contains high byte of 16-bit counter value |
| CNTL[7:0]<br>CNT[7:0] | **Timer Count Value [7:0] -** Contains low byte of 16-bit counter value |

## 16.6.3 Timer x Counter Modulo Registers (TPMxMODH:TPMxMODL)

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 (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits TOF 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).

It is good practice to wait for an overflow interrupt so both bytes of the modulo register can be written well before a new overflow. An alternative approach is to reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

**TPMxMODH** **Base + 0x03**

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

Figure 16-7. Timer x Counter Modulo Register TPMxMODH

**TPMxMODL** **Base + 0x04**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 16-8. Timer x Counter Modulo Register TPMxMODL

Table 16-6. TPMxMODH and TPMxMODL Register Field Descriptions

| Field | Description |
|---|---|
| MODH[7:0]<br>MOD[15:8] | **Timer Modulo Value [15:8] -** Contains high byte of 16-bit timer modulo value |
| MODL[7:0]<br>MOD[7:0] | **Timer Modulo Value [7:0] -** Contains low byte of 16-bit timer modulo value |

## 16.6.4 Timer x Channel 0 Status and Control Register (TPM1C0SC)

TPMxC0SC contains the Channel 0 interrupt status flag and control bits that are used to configure the interrupt enable, channel configuration, and pin function.

**TPMxC0SC** **Base + 0x05**

|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| W |  |  |  |  |  |  | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

Table 16-7. TPMxC0SC Register Field Descriptions

| Field | Description |
|---|---|
| 7<br>CH0F | **Channel 0 Flag** — When Channel 0 is configured for input capture, this flag bit is set when an active edge occurs on the TPMx pin. When Channel 0 is an output compare or edge-aligned PWM channel, CH0F is set when the value in the TPM counter registers matches the value in the TPM Channel 0 value registers. This flag is seldom used with center-aligned PWMs because it is set every time the counter matches the channel value register, which correspond to both edges of the active duty cycle period.<br>A corresponding interrupt is requested when CH0F is set and interrupts are enabled (CH0IE = 1). Clear CH0F by reading TPM1CnSC while CH0F is set and then writing a 0 to CH0F. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CH0F would remain set after the clear sequence was completed for the earlier CH0F. This is done so a CH0F interrupt request cannot be lost by clearing a previous CH0F. Reset clears CH0F. Writing a 1 to CH0F has no effect.<br>0 No input capture or output compare event occurred on channel n<br>1 Input capture or output compare event occurred on channel n |
| 6<br>CH0IE | **Channel 0 Interrupt Enable** — This read/write bit enables interrupts from Channel 0. Reset clears CH0IE.<br>0 Channel 0 interrupt requests disabled (use software polling)<br>1 Channel 0 interrupt requests enabled |
| 5<br>MS0B | **Mode Select B for TPM Channel 0** — When CPWMS = 0, MS0B = 1 configures TPM Channel 0 for edge-aligned PWM mode. For a summary of channel mode and setup controls, see Table 16-8. |

Table 16-7. TPMxC0SC Register Field Descriptions

| Field | Description |
|---|---|
| 4<br>MS0A | **Mode Select A for TPM Channel 0** — When CPWMS = 0 and MS0B = 0, MS0A configures TPM Channel 0 for input capture mode or output compare mode. See Table 16-8 for a summary of channel mode and setup controls. |
| 3:2<br>ELS0[B:A] | **Edge/Level Select Bits** — Depending on the operating mode for the timer channel as set by CPWMS:MS0B:MS0A and shown in Table 16-8, 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.<br>Setting ELS0B:ELS0A to 0:0 configures the related timer pin as a general-purpose I/O pin unrelated to any timer channel functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general-purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin. |

Table 16-8. Mode, Edge, and Level Selection

| CPWMS | MS0B:MS0A | ELS0B:ELS0A | Mode | Configuration |
|---|---|---|---|---|
| X | XX | 00 | Pin not used for TPM channel; use as an external gate for the TPM or revert to general-purpose I/O | |
| 0 | 00 | 01 | Input capture | Capture on rising edge only |
| | | 10 | | Capture on falling edge only |
| | | 11 | | Capture on rising or falling edge |
| | 01 | 00 | Output compare | Software compare only |
| | | 01 | | Toggle output on compare |
| | | 10 | | Clear output on compare |
| | | 11 | | Set output on compare |
| | 1X | 10 | Edge-aligned PWM | High-true pulses (clear output on compare) |
| | | X1 | | Low-true pulses (set output on compare) |
| 1 | XX | 10 | Center-aligned PWM | High-true pulses (clear output on compare-up) |
| | | X1 | | Low-true pulses (set output on compare-up) |

If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts or using the status flags to avoid any unexpected behavior.

## 16.6.5 Timer x Channel Value Registers (TPMxC0VH:TPMxC0VL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel value registers are cleared

In input capture mode, reading either byte (TPMxC0VH or TPMxC0VL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPM1CnSC register is written.

In output compare or PWM modes, writing to either byte (TPMxC0VH or TPMxC0VL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the timer channel value registers. The high byte should be written first, followed by the low byte. This latching mechanism may be manually reset by writing to the TPM1CnSC register.

This latching mechanism allows coherent 16-bit writes in either order, which is friendly to various compiler implementations.

| | TPMxC0VH | | | | Base + 0x06 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-9. Timer x Channel Value Register TPMxC0VH**

| | TPMxC0VL | | | | Base + 0x07 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Timer x Channel Value Register TPMxC0VL**

**Table 16-9. TPMxC0VH and TPMxC0VL Register Field Descriptions**

| Field | Description |
|---|---|
| CoVH[7:0]<br>C0V[15:8] | **Channel 0 Value [15:8] -** Contains high byte of 16-bit Channel 0 capture value |
| C0VL[7:0]<br>C0V[7:0] | **Channel 0 Value [7:0] -** Contains low byte of 16-bit Channel 0 capture value |

# Chapter 17
# Carrier Modulator Timer (CMT) Module

## 17.1 Introduction

Carrier Modulator Timer (CMT) module is a IR LED driver. The CMT consists of a carrier generator, modulator, and transmitter that drives the infrared out (IRO) pin. The module can transmit data to IRO output pin either in baseband or in FSK mode. The IRO pin has 20mA current drive capability.

## 17.2 Features

The features of the CMT module include:

- Four modes of operation
  - Time with independent control of high and low times
  - Baseband
  - Frequency shift key (FSK)
  - Direct software control of IRO pin
- Extended space operation in time, baseband, and FSK modes
- Selectable input clock divide: 1, 2, 4, or 8
- Interrupt on end of cycle
- Ability to disable IRO pin and use as timer interrupt
- CMT use as a timer resource.

## 17.3 Block Diagram

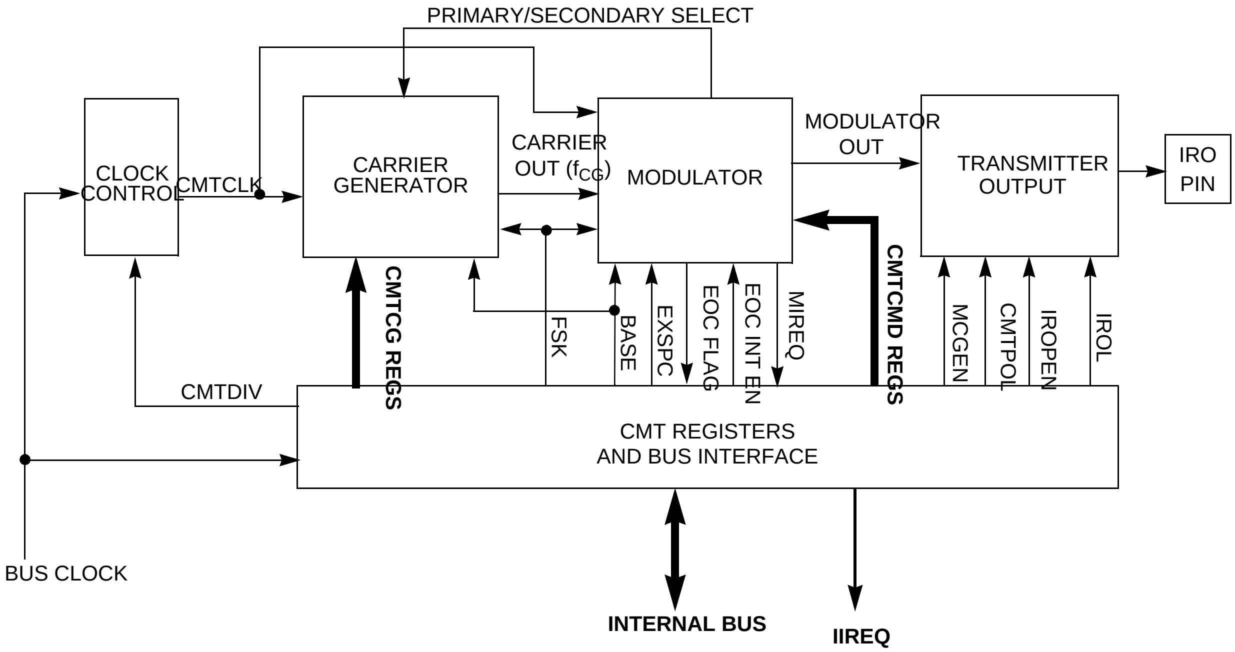


**Figure 17-1. Carrier Modulator Timer Module Block Diagram**

## 17.4 External Signal (IRO)

The CMT has only one external pin called IRO. The pin is driven by the transmitter output when the MCGEN bit in the CMTMSC register and the IROPEN bit in the CMTOC register are set. If the MCGEN bit is clear and the IROPEN bit is set, the pin is driven by the IROL bit in the CMTOC register. This enables user software to directly control the state of the IRO pin by writing to the IROL bit. If the IROPEN bit is clear, the pin is disabled and is not driven by the CMT module. This is so the CMT can be configured as a modulo timer for generating periodic interrupts without causing pin activity.

## 17.5 Functional Description

The CMT module consists of a carrier generator, a modulator, a transmitter output, and control registers. The block diagram is shown in Figure 17-1. The module CMTCLK is derived from the bus clock.

### 17.5.1 Clock Control

The CMT clock (CMTCLK) is generated from a prescaler that is controlled by the CMTDIV[2:0] field which is located across two registers, i.e., CMTOC register (see Section 17.7.2, “CMT Output Control Register (CMTOC)”) and CMTMSC register (see Section 17.7.3, “CMT Modulator Status and Control Register (CMTMSC)”). The prescaler is clocked by the bus clock which is typically 16MHz. Table 17-1 lists the divide ratio vs. the CMTDIV[2:0] field setting.

The CMT clock can also be disabled by the CMT bit of control register SCGC1, see Section 5.8.12, “System Clock Gating Control 1 Register (SCGC1)”.

Table 17-1. CMT Timing Parameters vs. CMTDIV[2:0] @ 16 MHz Bus Clock

| CMTDIV[2:0] | Divide Ratio | Clock Resolution w/16 MHz Bus Clock (µs) | Carrier Generator Resolution (µs) | Min. Carrier Generator Period (µs) | Min. Modulator Period (µs) |
|---|---|---|---|---|---|
| 000 | ÷ 1 | 0.0625 | 0.0625 | 0.125 | 0.5 |
| 001 | ÷ 2 | 0.125 | 0.125 | 0.250 | 1.0 |
| 010 | ÷ 4 | 0.250 | 0.250 | 0.500 | 2.0 |
| 011 | ÷ 8 | 0.500 | 0.500 | 1.000 | 4.0 |
| 100 | ÷ 16 | 1.000 | 1.000 | 2.000 | 8.0 |
| 101 | ÷ 1 | 0.0625 | 0.0625 | 0.125 | 0.5 |
| 110 | ÷ 1 | 0.0625 | 0.0625 | 0.125 | 0.5 |
| 111 | ÷ 1 | 0.0625 | 0.0625 | 0.125 | 0.5 |

## 17.5.2 Operating Modes Overview

The module has three main modes of operation: time, baseband, and frequency shift keying (FSK). These are shown in Figure 17-2.

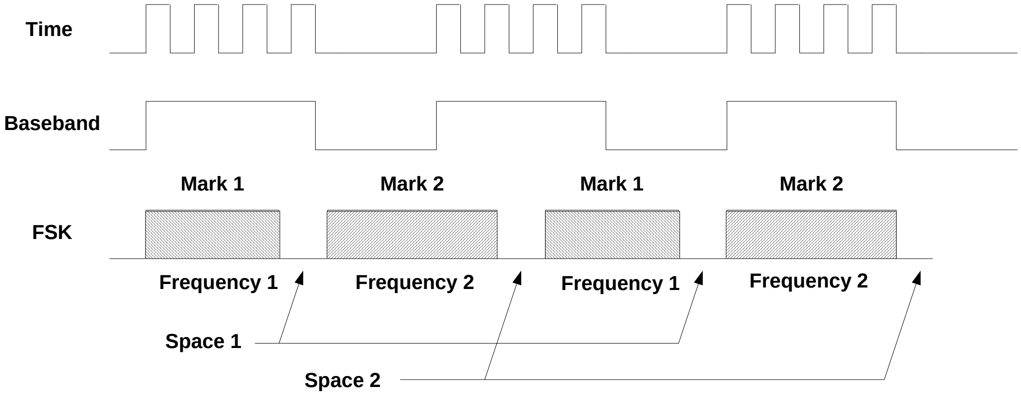


Figure 17-2. CMT Operating Modes

### 17.5.2.1 Time Mode

When operating in time mode, the output pulse stream is a carrier signal where the carrier high and low times are independently defined to determine both period and duty cycle. This frequency is developed in the carrier generator and it is gated on and off by the modulator.

The best carrier generator resolution is 62.5 ns when operating with an 16 MHz bus frequency (CMTDIV[2:0] =0; divide-by-1). The carrier generator can generate signals with periods between 125 ns (8 MHz) and 31.875 µs (31.37 kHz) in steps of 62.5 ns. See Table 17-1 for the minimum carrier generator resolution and period. If a lower frequency is desired, the pre scale divide ratio may be increased with the cost low a longer resolution time.

The possible duty cycle options will depend upon the number of counts required to complete the carrier period. For example, a 1.6 MHz signal has a period of 625 ns and will therefore require 10 of the 62.5 nS counts to generate. These counts may be split between high and low times, so various duty cycles are available.

For lower frequency signals with larger periods, higher resolution (as a percentage of the total period) duty cycles are possible.

The on/off times are also programmable in the modulator.

### 17.5.2.2 Baseband Mode

Baseband mode is a special version of time mode. The generator output is held continuously high and allows the modulator output to directly drive the CMT output signal. The output signal is then determined by the desired up time (mark) and down time (space) of the modulator.

### 17.5.2.3 FSK Mode

A third mode allows the carrier generator to alternate between two sets of high and low times (two different frequencies). When operating in FSK mode, the generator toggles between the two frequency sets when instructed by the modulator and requires no programming intervention.

The modulator can set different mark and space timing for the two FSK frequencies. The modulator provides a simple method to control protocol timing. It can count bus clocks (to provide real-time control) or it can count carrier clocks (for self-clocked protocols). See Section 17.5.4, “Modulator”, for more details.

### 17.5.2.4 Mode Control and Options

In addition to the three common modes described above:

- These modes can be disabled, and the CMT output state can be controlled by a control register bit. This allows a software managed output state.
- There is an option for extended “space” time between FSK pulses.
- The IRO output can be disabled and the timers used as a simple timing resource.

The modes are controlled primarily by the following control bits:

1. Modulator and Carrier Generator Enable (MCGEN) - this bit enables the carrier generator, modulator, and clocks for normal operation. When disabled, the IRO output state can be set by writing to the IROL control bit.
2. Baseband Enable (BASE) - when enabled the generator output is forced high and the output timing is determined by the modulator
3. FSK Mode Select (FSK) - when set selects FSK mode. When clear, time/baseband mode is enabled and modified by the BASE enable bit.
4. Extended Space Enable (EXSPC) - this is an optional feature that allows for extended time (“space” time) between FSK pulses (“mark” time).

A summary of the possible modes is shown in Table 17-2.

Table 17-2. CMT Modes of Operation

| Mode | MCGEN Bit[1] | BASE Bit[2] | FSK Bit[(2)] | EXSPC Bit | Comment |
|---|---|---|---|---|---|
| Time | 1 | 0 | 0 | 0 | $f_{CG}$ controlled by primary high and low registers.<br>$f_{CG}$ transmitted to IRO pin when modulator gate is open. |
| Baseband | 1 | 1 | x | 0 | $f_{CG}$ is always high. IRO pin high when modulator gate is open. |
| FSK | 1 | 0 | 1 | 0 | $f_{CG}$ control alternates between primary high/low registers and secondary high/low registers.<br>$f_{CG}$ transmitted to IRO pin when modulator gate is open. |
| Extended Space | 1 | x | x | 1 | Setting the EXSPC bit causes subsequent modulator cycles to be spaces (modulator out not asserted) for the duration of the modulator period (mark and space times). |
| IRO Latch | 0 | x | x | x | IROL bit controls state of IRO pin. |

[1] To prevent spurious operation, initialize all data and control registers before beginning a transmission (MCGEN=1).

[2] These bits are not double buffered and should not be changed during a transmission (while MCGEN=1).

### 17.5.3 Carrier Generator

The carrier generator block diagram is shown in Figure 17-3. The carrier signal is generated by counting a register-selected number of CMTCLK input clocks (62.5 ns minimum period for an 16 MHz bus clock) for both the carrier high time and the carrier low time. The period is determined by the total number of clocks counted. The duty cycle is determined by the ratio of high time clocks to total clocks counted. The high and low time values are user programmable and are held in two registers.

An alternate set of high/low count values is held in second set of registers to allow the generation of dual frequency FSK (frequency shift keying) protocols without CPU intervention.

NOTE

Only non-zero data values are allowed. The carrier generator will not work if any of the count values are equal to zero.

The MCGEN bit in the CMTMSC register must be set and the BASE bit must be cleared to enable carrier generator clocks. When the BASE bit is set, the carrier output to the modulator is held high continuously. The block diagram is shown in Figure 17-3.

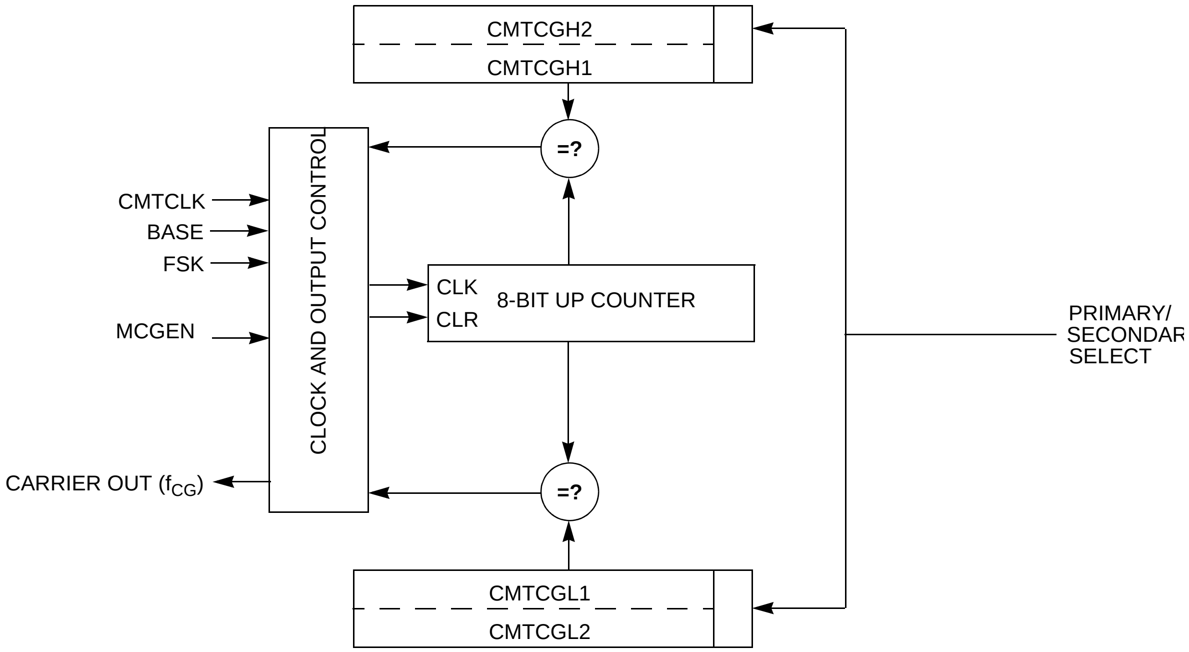


**Figure 17-3. Carrier Generator Block Diagram**

The high/low time counter is an 8-bit up counter. After each increment, the contents of the counter are compared with the appropriate high or low count value register. When the compare value is reached, the counter is reset to a value of 0x01, and the compare is redirected to the other count value register.

Assuming that the high time count compare register is currently active, a valid compare will cause the carrier output to be driven low. The counter will continue to increment (starting at reset value of 0x01). When the value stored in the selected low count value register is reached, the counter will again be reset and the carrier output will be driven high.

The cycle repeats, automatically generating a periodic signal that is directed to the modulator. The lowest frequency (maximum period) and highest frequency (minimum period) that can be generated are defined as:

$$f_{max} = f_{CMTCLK} / (2 \times 1) \text{ Hz} \qquad \textit{Eqn. 17-1}$$

$$f_{min} = f_{CMTCLK} / (2 \times (2^8 - 1)) \text{ Hz} \qquad \textit{Eqn. 17-2}$$

In the general case, the carrier generator output frequency is:

$$f_{CG} = f_{CMTCLK} / (\text{Highcount} + \text{Lowcount}) \text{ Hz} \qquad \textit{Eqn. 17-3}$$

Where: 0 < *Highcount* < 256 and
0 < *Lowcount* < 256

The duty cycle of the carrier signal is controlled by varying the ratio of high time to low-plus-high time. As the input clock period is fixed, the duty cycle resolution will be proportional to the number of counts required to generate the desired carrier period.

$$\text{Duty Cycle} = \frac{\text{Highcount}}{\text{Highcount} + \text{Lowcount}} \qquad \textit{Eqn. 17-4}$$

### 17.5.4 Modulator

The modulator block diagram is shown in Figure 17-4. The modulator has three main modes of operation:

- Gate the carrier onto the modulator output (time mode)
- Control the logic level of the modulator output (baseband mode)
- Count carrier periods and instruct the carrier generator to alternate between two carrier frequencies whenever a modulation period (mark + space counts) expires (FSK mode)

The modulator includes a 17-bit down counter with underflow detection. The counter is loaded from the 16-bit modulation mark period buffer registers, CMTCMD1 and CMTCMD2. The most significant bit is loaded with a logic zero and serves as a sign bit. When the counter holds a positive value, the modulator gate is open and the carrier signal is driven to the transmitter block.

When the counter underflows, the modulator gate is closed and a 16-bit comparator is enabled that compares the logical complement of the value of the down-counter with the contents of the modulation space period register (which has been loaded from the registers CMTCMD3 and CMTCMD4).

When a match is obtained the cycle repeats by opening the modulator gate, reloading the counter with the contents of CMTCMD1 and CMTCMD2, and reloading the modulation space period register with the contents of CMTCMD3 and CMTCMD4.

If the contents of the modulation space period register are all zeroes, the match will be immediate and no space period will be generated (for instance, for FSK protocols that require successive bursts of different frequencies).

The MCGEN bit in the CMTMSC register must be set to enable the modulator timer.

**NOTE**

> The CMTCLK input is then prescaled by a divide ratio of 8 before clocking the modulator counters. The minimum clock period for the modulator then is 62.5 ns times 8 or 500 ns.

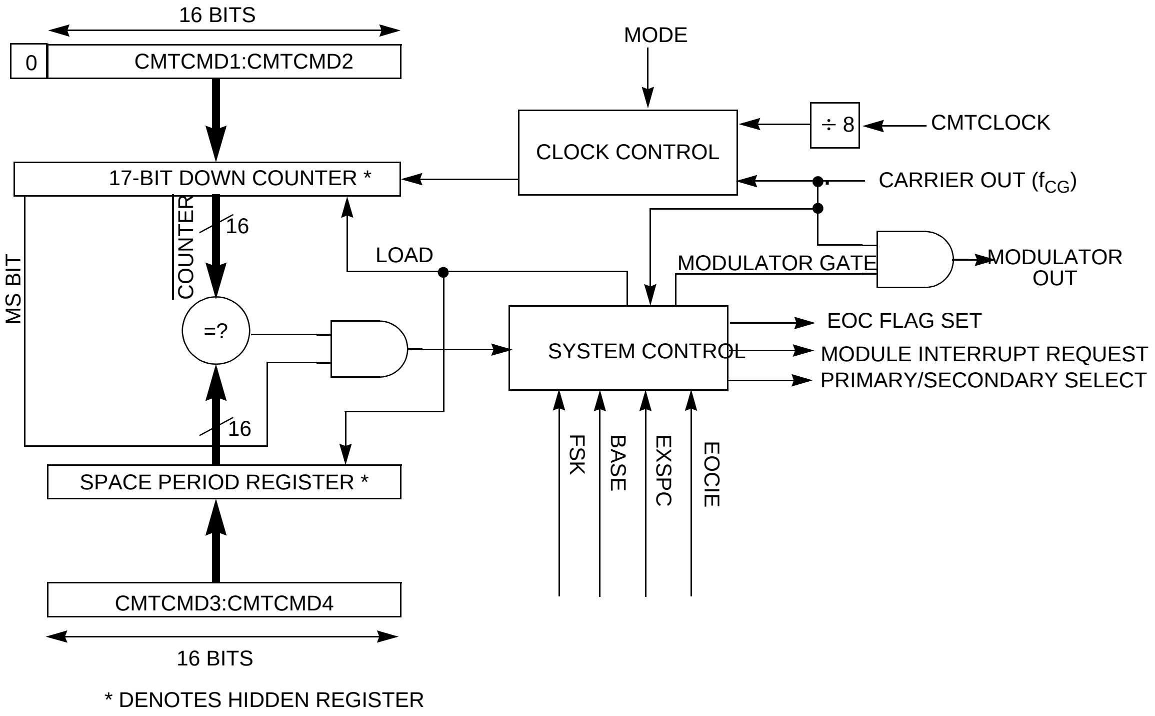


**Figure 17-4. Modulator Block Diagram**

### 17.5.4.1 Modulator Time Mode

When the modulator operates in time mode (MCGEN bit is set, BASE bit is clear, and FSK bit is clear), the modulation mark period consists of an integer number of CMTCLK/8 clock periods. The modulation space period consists of zero or an integer number of CMTCLK/8 clock periods. With an 16 MHz bus and CMTDIV[2:0] = 00, the modulator resolution is 0.5 µs and has a maximum mark and space period of about 32.767 ms each. See Figure 17-5 for an example of the time mode and baseband mode outputs.

The mark and space time equations for time and baseband mode are:

$$t_{mark} = (CMTCMD1{:}CMTCMD2 + 1) / (f_{CMTCLK} / 8) \qquad \textit{Eqn. 17-5}$$

$$t_{space} = CMTCMD3{:}CMTCMD4 / (f_{CMTCLK} / 8) \qquad \textit{Eqn. 17-6}$$

where CMTCMD1:CMTCMD2 and CMTCMD3:CMTCMD4 are the decimal values of the concatenated registers.

## NOTE

If the modulator is disabled while the $t_{mark}$ time is less than the programmed carrier high time ($\mathbf{t_{mark}}$ < CMTCGH1/$f_{CMTCLK}$), the modulator can enter into an illegal state and end the current cycle before the programmed value. Make sure to program $\mathbf{t_{mark}}$ greater than the carrier high time to avoid this illegal state.

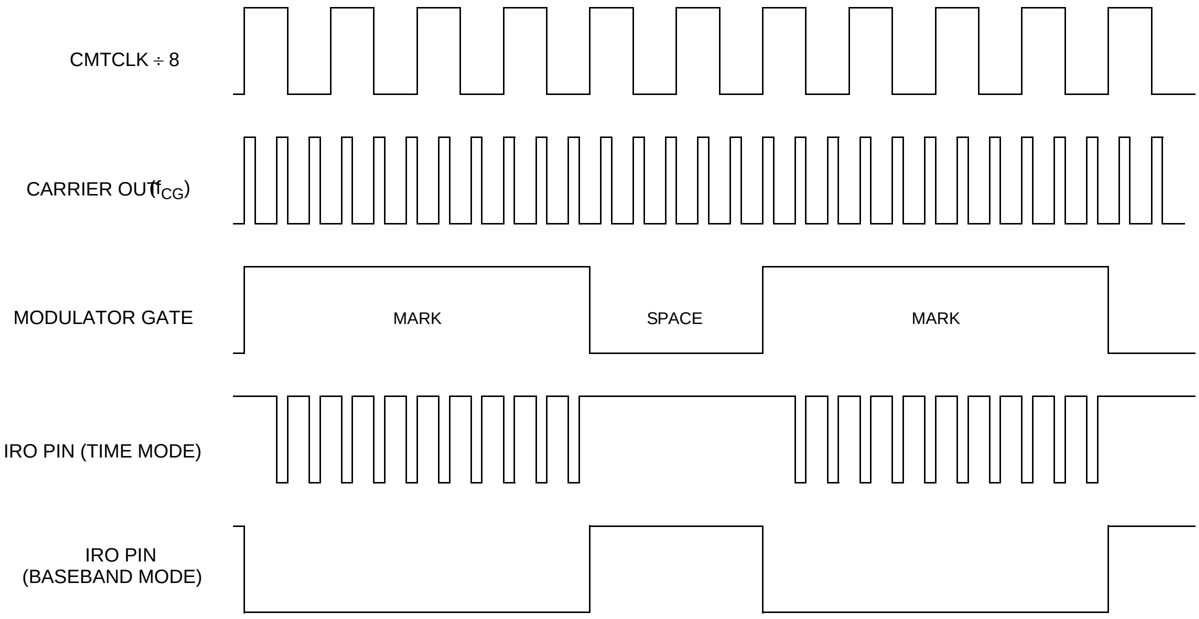


**Figure 17-5. Example CMT Output in Time and Baseband Modes**

### 17.5.4.2 Modulator Baseband Mode

Baseband mode (MCGEN bit is set and BASE bit is set) is a derivative of time mode, where the mark and space period is based on (CMTCLK / 8) counts. The mark and space calculations are the same as in time mode. In this mode the modulator output will be at a logic 1 for the duration of the mark period and at a logic 0 for the duration of a space period. See Figure 17-5 for an example of the output for both baseband and time modes. In the example, the carrier out frequency ($f_{CG}$) is generated with a high count of 0x01 and a low count of 0x02, which results in a divide of 3 of CMTCLK with a 33 percent duty cycle. The modulator down-counter was loaded with the value 0x0003 and the space period register with 0x0002.

**NOTE**

> The waveforms in Figure 17-4 and Figure 17-5 are for the purpose of conceptual illustration and are not meant to represent precise timing relationships between the signals shown.

### 17.5.4.3 Modulator FSK Mode

When the modulator operates in FSK mode (MCGEN bit is set, FSK bit is set, and BASE bit is clear), the modulation mark and space periods consist of an integer number of carrier clocks (space period can be 0). When the mark period expires, the space period is transparently started (as in time mode). The carrier generator toggles between primary and secondary data register values whenever the modulator space period expires.

The space period provides an interpulse gap (no carrier). If CMTCMD3:CMTCMD4 = 0x0000, then the modulator and carrier generator will switch between carrier frequencies without a gap or any carrier glitches (zero space).

Using timing data for carrier burst and interpulse gap length calculated by the CPU, FSK mode can automatically generate a phase-coherent, dual-frequency FSK signal with programmable burst and interburst gaps.

The mark and space time equations for FSK mode are:

$$t_{mark} = (CMTCMD1{:}CMTCMD2 + 1) / f_{CG} \qquad \textit{Eqn. 17-7}$$

$$t_{space} = CMTCMD3{:}CMTCMD4 / f_{CG} \qquad \textit{Eqn. 17-8}$$

Where $f_{CG}$ is the frequency output from the carrier generator. The example in Figure 17-6 shows what the IRO pin output looks like in FSK mode with the following values: CMTCMD1:CMTCMD2 = 0x0003, CMTCMD3:CMTCMD4 = 0x0002, primary carrier high count = 0x01, primary carrier low count = 0x02, secondary carrier high count = 0x03, and secondary carrier low count = 0x01.

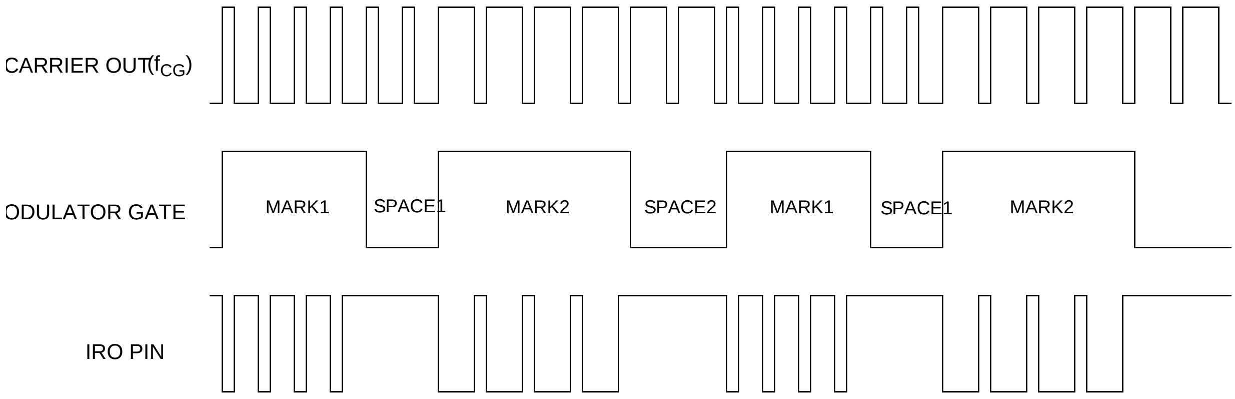


**Figure 17-6. Example CMT Output in FSK Mode**

## 17.5.5 Extended Space Operation

In time, baseband, or FSK mode, the space period can be made longer than the maximum possible value of the space period register. Setting the EXSPC bit in the CMTMSC register will force the modulator to treat the next modulation period (beginning with the next load of the counter and space period register) as a space period equal in length to the mark and space counts combined. Subsequent modulation periods will consist entirely of these extended space periods with no mark periods. Clearing EXSPC will return the modulator to standard operation at the beginning of the next modulation period.

#### 17.5.5.1 EXSPC Operation in Time Mode

To calculate the length of an extended space in time or baseband modes, add the mark and space times and multiply by the number of modulation periods that EXSPC is set.

$$t_{exspace} = t_{space} + (t_{mark} + t_{space}) \text{ x (number of modulation periods)} \qquad \textit{Eqn. 17-9}$$

For an example of extended space operation, see Figure 17-7.

**NOTE**

The EXSPC feature can be used to emulate a zero mark event.

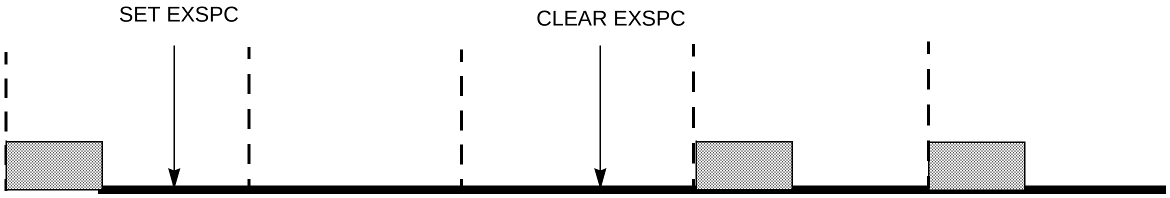


**Figure 17-7. Extended Space Operation**

#### 17.5.5.2 EXSPC Operation in FSK Mode

In FSK mode, the modulator continues to count carrier out clocks, alternating between the primary and secondary registers at the end of each modulation period.

To calculate the length of an extended space in FSK mode, the user must know whether the EXSPC bit was set on a primary or secondary modulation period, as well as the total number of both primary and secondary modulation periods completed while the EXSPC bit is high. A status bit for the current modulation is not accessible to the CPU. If necessary, software should maintain tracking of the current modulation cycle (primary or secondary). The extended space period ends at the completion of the space period time of the modulation period during which the EXSPC bit is cleared.

If the EXSPC bit was set during a primary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_p + (t_{mark} + t_{space})_s + (t_{mark} + t_{space})_p + \ldots \qquad \textit{Eqn. 17-10}$$

Where the subscripts p and s see mark and space times for the primary and secondary modulation cycles.

If the EXSPC bit was set during a secondary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_s + (t_{mark} + t_{space})_p + (t_{mark} + t_{space})_s + \ldots \qquad \textit{Eqn. 17-11}$$

### 17.5.6 Transmitter

The transmitter output block controls the state of the infrared out pin (IRO). The modulator output is gated to the IRO pin when the modulator/carrier generator is enabled. When the modulator/carrier generator is disabled, the IRO pin is controlled by the state of the IRO latch.

A polarity bit in the CMTOC register enables the IRO pin to be high true or low true.

### 17.5.7 CMT Interrupts

The CMT has only one source of an interrupt request which is the end of cycle flag (EOCF). This status flag is set when:

- The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission
- At the end of each modulation cycle (when the counter is reloaded from CMTCMD1:CMTCMD2) while the MCGEN bit is set

In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, the EOCF bit will not be set when the MCGEN is set, but will become set at the end of the current modulation cycle.

When the MCGEN becomes disabled, the CMT module does not set the EOC flag at the end of the last modulation cycle.

The EOCF bit is cleared by reading the CMT modulator status and control register (CMTMSC) followed by an access of CMTCMD2 or CMTCMD4.

If the EOC interrupt enable (EOCIE) bit is high when the EOCF bit is set, the CMT module will generate an interrupt request. The EOCF bit must be cleared within the interrupt service routine to prevent another interrupt from being generated after exiting the interrupt service routine.

The EOC interrupt is coincident with loading the down-counter with the contents of CMTCMD1:CMTCMD2 and loading the space period register with the contents of CMTCMD3:CMTCMD4. The EOC interrupt provides a means for the user to reload new mark/space values into the modulator data registers. Modulator data register updates will take effect at the end of the current modulation cycle. Note that the down-counter and space period register are updated at the end of every modulation cycle, regardless of interrupt handling and the state of the EOCF flag.

**NOTE**

If the CMT output is disabled, the CMT can be programmed for operation and still generate a corresponding status. This mode may be used as a cyclical timer to supplement other device timing resources.

## 17.6 Modes of Operation

The CMT is affected by the device mode of operation and the module clock can be disabled:

- For lower power, the CMT module clock can be disabled via the SPI control bit of the SCGC1 Register
- LPRun - the reference oscillator is divided by 64 and all clocks are scaled by this factor. It is recommended to disable the CMT during this mode as all frequencies and timing changes accordingly.
- Wait modes - there are two wait modes
  - Wait - when entered, the CPU clock is disabled and the bus clock still runs at normal speed . The CMT, if enabled, will continue to operate normally. However, there will be no new codes or changes of pattern mode while in wait mode, because the CPU is not operating.

- LP Wait - when entered, the CPU clock is disabled and the bus clock still runs but at a reduced speed. The CMT, if enabled, will continue to operate, however, all timing would be incorrect. Operation is not recommended.
- The CMT is disabled in Stop3 mode, regardless of the settings before executing the STOP instruction.
  - During Stop3 mode, clocks to the CMT module are halted. No registers are affected. If Stop3 is exited with a reset, the CMT will be put into its reset state. If Stop3 is exited with an interrupt, the CMT will resume operation. Software should ensure Stop3 mode is not entered while the modulator is in operation to prevent the IRO pin from being asserted while in stop mode. This may require a time-out period from the time that the MCGEN bit is cleared to allow the last modulator cycle to complete.
- Background Mode Operation -When the microcontroller is in active background mode, the CMT temporarily suspends all counting until the microcontroller returns to normal user mode.

## 17.7 Register Definition

The following registers control and monitor CMT operation:

- CMT carrier generator data registers (CMTCGH1, CMTCGL1, CMTCGH2, CMTCGL2)
- CMT output control register (CMTOC)
- CMT modulator status and control register (CMTMSC)
- CMT modulator period data registers (CMTCMD1, CMTCMD2, CMTCMD3, CMTCMD4)

The CMT registers are all located in the Direct-Page Register Map.

### 17.7.1 Carrier Generator Data Registers ( CMTCGH1, CMTCGL1, CMTCGH2, and CMTCGL2)

The carrier generator data registers contain the primary and secondary high and low values for generating the carrier output.

| | CMTCGH1 | | | | 0x0060 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PH7 | PH6 | PH5 | PH4 | PH3 | PH2 | PH1 | PH0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

**Figure 17-8. Carrier Generator Data Register CMTCGH1**

Table 17-3. CMTCGH1 Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>PH[7:0] | **Primary Carrier High Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 17.5.4.1, “Modulator Time Mode”), this register pair is always selected. When operating in FSK mode (see Section 17.5.4.3, “Modulator FSK Mode”), this register pair and the secondary register pair are alternatively selected under control of the modulator. The primary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled to avoid spurious results. |

| | CMTCGL1 | | | | 0x0061 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | PL7 | PL6 | PL5 | PL4 | PL3 | PL2 | PL1 | PL0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

Figure 17-9. Carrier Generator Data Register CMTCGL1

Table 17-4. CMTCGL1 Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>PL[7:0] | **Primary Carrier Low Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 17.5.4.1, “Modulator Time Mode”), this register pair is always selected. When operating in FSK mode (see Section 17.5.4.3, “Modulator FSK Mode”), this register pair and the secondary register pair are alternatively selected under control of the modulator. The primary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled to avoid spurious results. |

| | CMTCGH2 | | | | 0x0062 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | SH7 | SH6 | SH5 | SH4 | SH3 | SH2 | SH1 | SH0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

Figure 17-10. Carrier Generator Data Register CMTCGH2

### Table 17-5. CMTCGH2 Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>SH[7:0] | **Secondary Carrier High Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 17.5.4.1, “Modulator Time Mode”), this register pair is never selected. When operating in FSK mode (see Section 17.5.4.3, “Modulator FSK Mode”), this register pair and the primary register pair are alternatively selected under control of the modulator. The secondary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled when operating in FSK mode. |

| | **CMTCGL2** | | | | **0x0063** | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | SL7 | SL6 | SL5 | SL4 | SL3 | SL2 | SL1 | SL0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

### Table 17-6. CMTCGL2 Field Descriptions

| Field | Description |
|---|---|
| 7:0<br>SL[7:0] | **Secondary Carrier Low Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 17.5.4.1, “Modulator Time Mode”), this register pair is never selected. When operating in FSK mode (see Section 17.5.4.3, “Modulator FSK Mode”), this register pair and the primary register pair are alternatively selected under control of the modulator. The secondary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled when operating in FSK mode. |

## 17.7.2 CMT Output Control Register (CMTOC)

This register is used to control the IRO output of the CMT module.

| CMTOC | | | | | 0x0064 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | IROL | CMTPOL | IROPEN | CMTDIV2 | 0 | 0 | 0 | 0 |
| W | | | | | ▨ | ▨ | ▨ | ▨ |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

IROPEN

**Figure 17-11. CMT Output Control Register (CMTOC)**

**Table 17-7. CMTOC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IROL | **IRO Latch Control** — Reading IROL reads the state of the IRO latch. Writing IROL changes the state of the IRO pin when the MCGEN bit is clear in the CMTMSC register and the IROPEN bit is set. |
| 6<br>CMTPOL | **CMT Output Polarity** — The CMTPOL bit controls the polarity of the IRO pin output of the CMT.<br>0 IRO pin is active low<br>1 IRO pin is active high |
| 5<br>IROPEN | **IRO Pin Enable** — The IROPEN bit is used to enable and disable the IRO pin. When the pin is enabled, it is an output that drives out either the CMT transmitter output or the state of the IROL bit depending on whether the MCGEN bit in the CMTMSC register is set. Also, the state of the output is either inverted or not depending on the state of the CMTPOL bit. When the pin is disabled, it is in a high impedance state so it doesn't draw any current. The pin is disabled during reset.<br>0 IRO pin disabled<br>1 IRO pin enabled as output |
| 4<br>CMTDIV2 | **CMT Clock Divide Prescaler 2**— This field is the MSB of the 3-bit CMTDIV[2:0] field that sets the clock prescaler ratio for the CMTCLK. CMTDIV[2:0] are Bit6 and Bit 5 of the CMTMSC Register. See Table 17-1for field settings and pre scale divide ratios. |

### 17.7.3 CMT Modulator Status and Control Register (CMTMSC)

The CMT modulator status and control register (CMTMSC) contains the modulator and carrier generator enable (MCGEN), end of cycle interrupt enable (EOCIE), FSK mode select (FSK), baseband enable (BASE), extended space (EXSPC), prescaler (CMTDIV1:CMTDIV0) bits, and the end of cycle (EOCF) status bit.

| | CMTMSC | | | | 0x0065 | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | EOCF | CMTDIV1 | CMTDIV0 | EXSPC | BASE | FSK | EOCIE | MCGEN |
| W | ▨ | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 17-12. CMT Modulator Status and Control Register (CMTMSC)**

**Table 17-8. CMTMSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EOCF | **End of Cycle Status Flag** — The EOCF bit is set when:<br>• The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission.<br>• At the end of each modulation cycle while the MCGEN bit is set. This is recognized when a match occurs between the contents of the space period register and the down-counter. At this time, the counter is initialized with the (possibly new) contents of the mark period buffer, CMTCMD1 and CMTCMD2. The space period register is loaded with the (possibly new) contents of the space period buffer, CMTCMD3 and CMTCMD4.<br>This flag is cleared by a read of the CMTMSC register followed by an access of CMTCMD2 or CMTCMD4.<br>In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, EOCF will not be set when MCGEN is set, but will be set at the end of the current modulation cycle.<br>0 No end of modulation cycle occurrence since flag last cleared<br>1 End of modulator cycle has occurred |
| 6:5<br>CMTDIV[1:0] | **CMT Clock Divide Prescaler [1:0]**— This field is the LS two bits of the 3-bit CMTDIV[2:0] field that sets the clock prescaler ratio for the CMTCLK. CMTDIV3 is Bit 4 of the CMTOC Register. See Table 17-1for field settings and pre scale divide ratios. |
| 4<br>EXSPC | **Extended Space Enable** — The EXSPC bit enables extended space operation.<br>0 Extended space disabled<br>1 Extended space enabled |
| 3<br>BASE | **Baseband Enable** — When set, the BASE bit disables the carrier generator and forces the carrier output high for generation of baseband protocols. When BASE is clear, the carrier generator is enabled and the carrier output toggles at the frequency determined by values stored in the carrier data registers. See Section 17.5.4.2, "Modulator Baseband Mode". This bit is cleared by reset. This bit is not double buffered and should not be written to during a transmission.<br>0 Baseband mode disabled<br>1 Baseband mode enabled |
| 2<br>FSK | **FSK Mode Select** — The FSK bit enables FSK operation.<br>0 CMT operates in time or baseband mode<br>1 CMT operates in FSK mode |

Table 17-8. CMTMSC Field Descriptions (continued)

| Field | Description |
|---|---|
| 1<br>EOCIE | **End of Cycle Interrupt Enable** — A CPU interrupt will be requested when EOCF is set if EOCIE is high.<br>0 CPU interrupt disabled<br>1 CPU interrupt enabled |
| 0<br>MCGEN | **Modulator and Carrier Generator Enable** — Setting MCGEN will initialize the carrier generator and modulator and enable all clocks. After it is enabled, the carrier generator and modulator will function continuously. When MCGEN is cleared, the current modulator cycle will be allowed to expire before all carrier and modulator clocks are disabled (to save power) and the modulator output is forced low. To prevent spurious operation, the user should initialize all data and control registers before enabling the system.<br>0 Modulator and carrier generator disabled<br>1 Modulator and carrier generator enabled |

### 17.7.4 CMT Modulator Data Registers (CMTCMD1, CMTCMD2, CMTCMD3, and CMTCMD4)

The modulator data registers control the mark and space periods of the modulator for all modes. The contents of these registers are transferred to the modulator down counter and space period register upon the completion of a modulation period.

Table 17-9. CMTCMDx Registers

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| **CMTCMD1** | R | MB15 | MB14 | MB13 | MB12 | MB11 | MB10 | MB9 | MB8 |
| **0x0066** | W | | | | | | | | |
| **CMTCMD2** | R | MB7 | MB6 | MB5 | MB4 | MB3 | MB2 | MB1 | MB0 |
| **0x0067** | W | | | | | | | | |
| **CMTCMD3** | R | SB15 | SB14 | SB13 | SB12 | SB11 | SB10 | SB9 | SB8 |
| **0x0068** | W | | | | | | | | |
| **CMTCMD4** | R | SB7 | SB6 | SB5 | SB4 | SB3 | SB2 | SB1 | SB0 |
| **0x0068** | W | | | | | | | | |

Table 17-10. CMT Modulator Data Registers Field Descriptions

| Field | Description |
|---|---|
| CMTCMD1[7:0]<br>MB[15:8] | **Mark Data [15:8]** - Contains high byte of 16-bit modulator mark period buffer |
| CMTCMD2[7:0]<br>MB[7:0] | **Mark Data [7:0]** - Contains low byte of 16-bit modulator mark period buffer |
| CMTCMD3[7:0]<br>SB[15:8] | **Space Data [15:8]** - Contains high byte of 16-bit modulator space period buffer |
| CMTCMD4[7:0]<br>SB[7:0] | **Space Data [7:0]** - Contains high byte of 16-bit modulator space period buffer |

# Chapter 18
# Development Support

## 18.1 Introduction

The MC13234/MC13237 provides a comprehensive debug/development capability for the HCS08 MCU. The debug system is implemented using two blocks:

- The Background Debug Controller (BDC) - has a single-wire debug interface (signal BKGD) to the 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.
- The on-chip Debug Module (DBG) - implements an on-chip ICE (in-circuit emulation) system and allows non-intrusive debug of application software by providing an on-chip trace buffer with flexible triggering capability

The single-wire background debug interface supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities that 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.

Because the MC13234/MC13237 does 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.

### 18.1.1 Features

Features of the background debug controller (BDC) include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access

- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the debug module (DBG) include:

- Two trigger comparators:
  - Two address + read/write (R/W)
  - 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
  - Event-only data
- Two types of breakpoints:
  - Tag breakpoints for instruction opcodes
  - Force breakpoints for any address access
- Nine trigger modes:
  - A-only
  - A OR B
  - A then B
  - A AND B data (full mode)
  - A AND NOT B data (full mode)
  - Event-only B (store data)
  - A then event-only B (store data)
  - Inside range (A ≤ address ≤ B)
  - Outside range (address < A or address > B)

### 18.1.2 Forcing Active Background Mode

Active background mode can be forced as follows:

- After a power-on reset by holding the BKGD pin low as the device exits the reset condition.
- 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

### 18.1.3 Debug Clock

The Bus Clock (default 16 MHz) is the source clock to the BDG and the debug port. The bus clock to the DBG can be gated on and off using the DBG bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the DBG bit can be cleared to disable the clock to this module when not in use. See Section 5.4.4, “Managing Clock Resources”, for details.

## 18.2 Background Debug Controller (BDC)

The BDC module operation and registers are described in the following description

### 18.2.1 BDC Overview

The MC13234/MC13237 BDC module provides a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. It does not interfere with normal application resources and does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

### 18.2.2 Block Diagram

Figure 18-1 is a block diagram of the BDC.

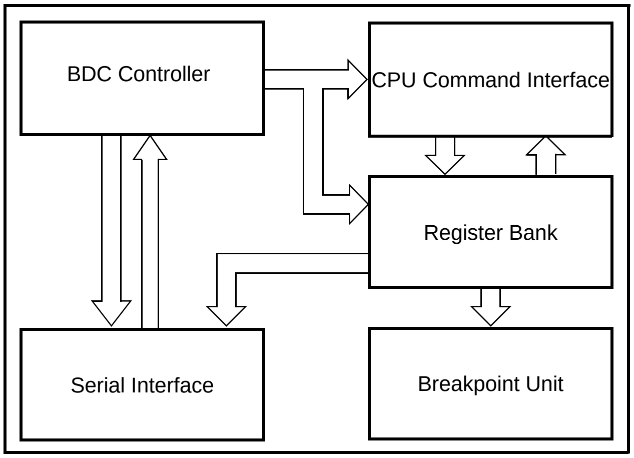


Figure 18-1. Block Diagram

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.

## 18.2.3 Standard Debug (BDM) Port

The single pin serial debug communications port is connected to the BDC and uses external signal BKGD. Typically, a simple debug interface pod is connected to the BDM port (see Figure 18-2) and is used to translate commands from a host computer to the single-wire, serial interface background debug port.

- The pod typically connects to the MC13234/MC13237 through a standardized port with ground, the BKGD pin, RESET, and VDD connections.
- If used, the $\overline{\text{RESET}}$ is an open-drain drive to the MC13234/MC13237 reset input that allows the host to force a target system hardware 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.
- VDD is the target system supply voltage and is not required. However, VDD can be used to supply pod power from the target system to avoid the need for a separate power supply. 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.

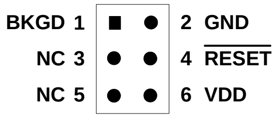


Figure 18-2. Standard BDM Port Connector

## 18.2.4 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.

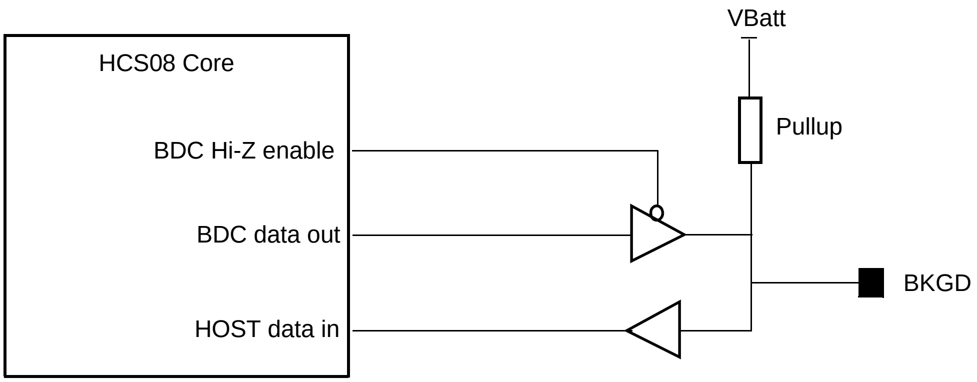


Figure 18-3. BKGD Pin Logic

BDC serial communications use a custom serial protocol that 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).

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 has 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. The source of this speedup pulse is the host for transmit cases and the target for receive cases.

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. It is not necessary to reset the target MCU to communicate with it through the background debug interface.

## 18.2.5 Serial Interface Communication Protocol

The communication channel protocol and timing are described in the following sections.

### 18.2.5.1 General Command protocol

As previously stated, the communication channel uses a single-wire interface that has a command protocol implemented on it. Figure 18-4 shows a generalized top-level command protocol.

- All commands are initiated by the host
- The host to the target command includes -
  - Always an 8-bit command opcode
  - Any additional command information (if required) - 1) address alone , or 2) address and data

- After any required wait, the target will send an ACK pulse, followed by any required read data or status
- All address data is a 16-bit length
- Write data can be an 8-bit or 16-bit length based on the command
- Read data can be an 8-bit or 16-bit length based on the command

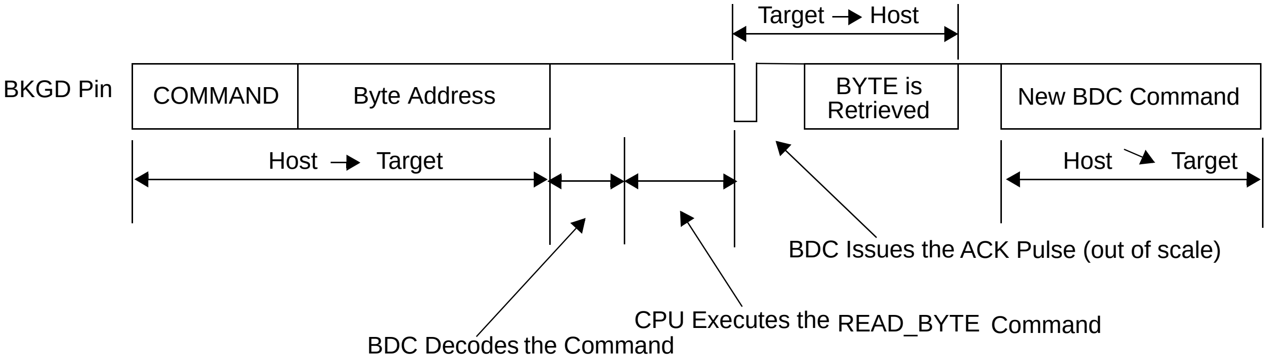


**Figure 18-4. BDM Command Level Protocol**

### 18.2.5.2 Serial Bit Timing

The serial command protocol is composed of individual “serial bits” with timing as described in this section. For each serial bit of data:

**NOTE**

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU

- The external controller generates 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.
- 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 time-out occurs is aborted without affecting the memory or operating mode of the target MCU system.

The protocol requires the debug pod to know the target BDC communication clock speed. For the MC13234/MC13237 the serial interface runs from the Bus Clock; the bus clock is 1/2 the reference oscillator frequency or 16MHz.

**NOTE**

- The MC13234/MC13237 reference oscillator must be running for the BDM port to be functional.
- When the BDM port is not in used, the bus clock to the DBG can be gated on and off using the DBG bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the DBG bit can be cleared to disable the clock to this module when not in use. See Section 5.4.4, “Managing Clock Resources” for details.

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 target 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.

### 18.2.5.2.1 Host-to-Target Bit Timing

Figure 18-5 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target. 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 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.

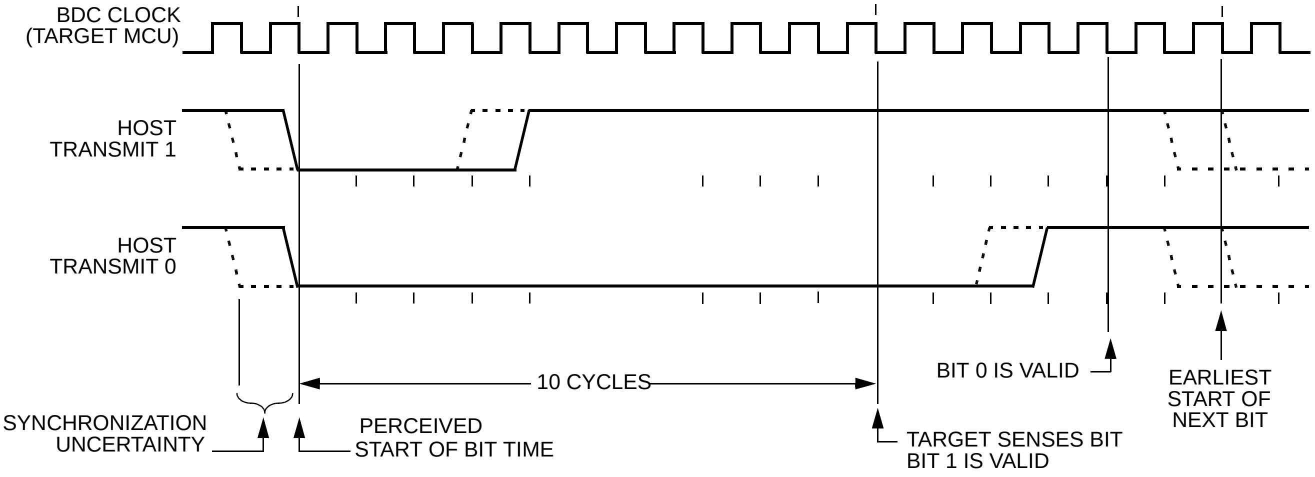


**Figure 18-5. BDC Host-to-Target Serial Bit Timing**

### 18.2.5.2.2 Target-to-Host Bit Timing

Figure 18-6 shows the host retrieving a logic 1 from the target. The host must hold the BKGD pin low, typically for 4 clock cycles, which is enough for the target to recognize the neg-edge in the BKPD pin (it should be at least two target clock cycles). After perceiving the neg-edge the target waits for seven target clock cycles and then issues a speed-up pulse for one cycle. Note that, the host must release the BKGD pin before the target drives the high speedup pulse, seven clock cycles after the perceived start of the bit time, otherwise a conflict will occur. The host should sample the BKGD pin about 10 clock cycles after it started the bit time to evaluate if it is a logic 1 or logic 0 bit retrieve.

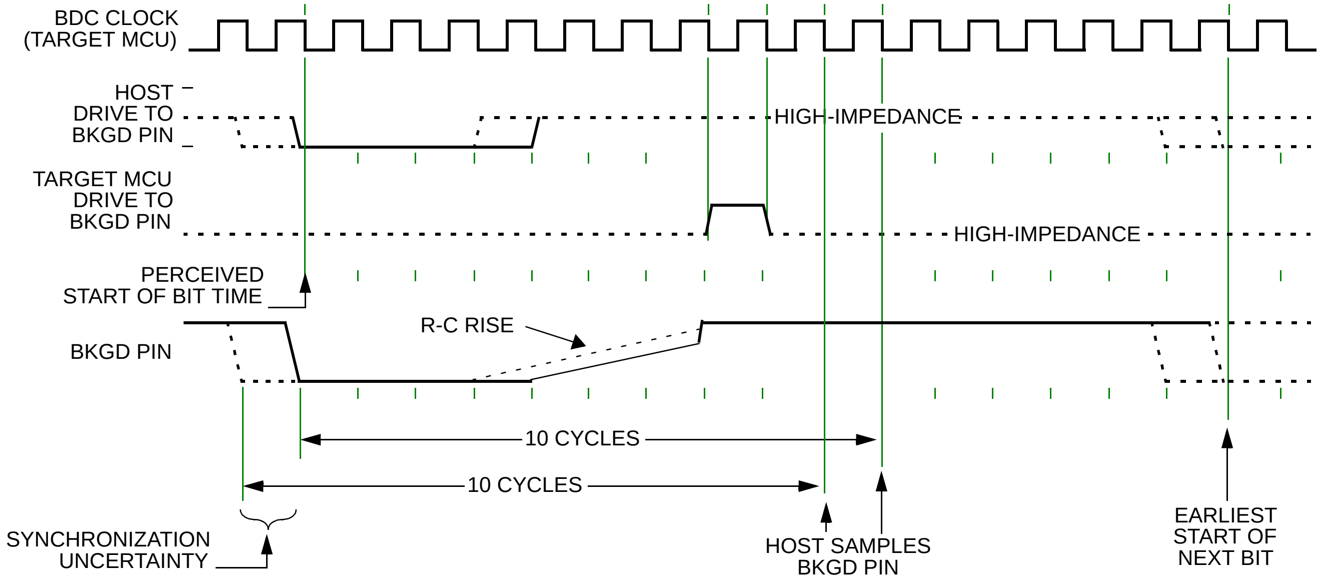


**Figure 18-6. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 18-7 shows the host retrieving a logic 0 from the target MCU. The host initiates the bit time, but the target MCU finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 12 clock cycles then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 clock cycles after starting the bit time.

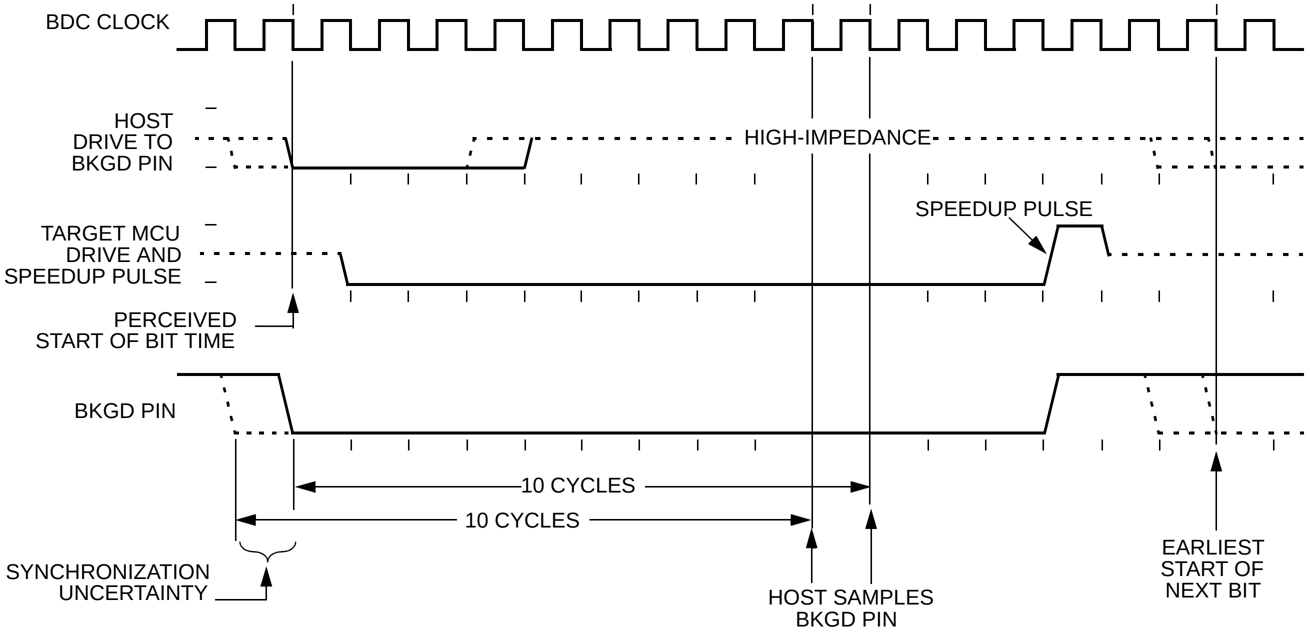


**Figure 18-7. BDC Target-to-Host Serial Bit Timing (Logic 0)**

### 18.2.5.3 Handshake Protocol using ACK

Because BDC commands are executed at the target bus rate and can be asynchronous to the serial interface clock, a handshake or acknowledge signal (ACK) is provided to the host to determine when an issued command has been executed by the target. Refer to Figure 18-4.

Figure 18-8 show the ACK pulse timing. The ACK is implemented as a 16 serial clock cycles low pulse followed by a brief speedup pulse on the BKGD pin, generated by the target when a command had been successfully executed.

- After the ACK pulse has finished, the host can start the bit retrieve, if the last issued command was a read command, or start a new command, if the last command was a write command or a control command (BACKGROUND, GO, GO_UNTIL or TRACE1).
- The ACK pulse is not issued any earlier then 32 serial clock cycles after the BDC command was issued.
- The end of the BDC command is assumed to be the 16th tick of the last bit. This minimum delay assures enough time for the host to perceive the ACK pulse.

**NOTE**

There is no upper limit for the delay between the command and the related ACK pulse.

- If a BDC command is not acknowledged by an ACK pulse, the host must first abort the pending command before issuing a new BDC command. - When the CPU enters Wait or Stop mode while the host issues a command that requires CPU execution (e.g.: WRITE_BYTE), the target discards the incoming command. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case.

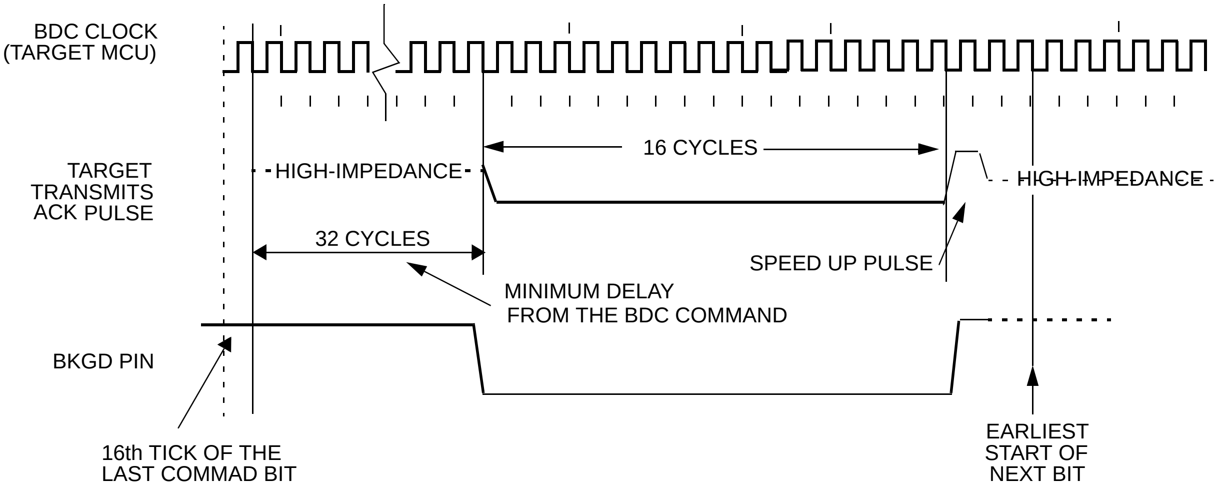


**Figure 18-8. Target Acknowledge Pulse (ACK)**

### 18.2.5.4 Hardware Handshake Abort Procedure

If a BDC command has not been acknowledged, an abort procedure must be executed before issuing a new command. The abort procedure is based on the SYNC command, which is really a timed Low pulse. To execute the SYNC command, the host controller should generate a low pulse in the BKGD pin by driving it low for at least 128 serial clock cycles and then driving it high for one serial clock cycle for the speedup pulse. By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, and assumes that the pending command and therefore the related ACK pulse, are being aborted. After the SYNC protocol completes, the host is free to issue new BDC commands. Figure 18-9 illustrates a SYNC command aborting a READ_BYTE. Note that after the command is aborted, a new command could be issued by the host computer.

It is important to notice that any issued BDC command that requires CPU execution will be executed at the next instruction boundary, provided the CPU does not enter Wait or Stop modes. If the host aborts a command by sending the SYNC pulse, it should then read the BDCSCR after the SYNC response is issued by the target, checking for DVF = 0, before attempting to send any new command that requires CPU execution. This prevents the new command from being discarded at the BDC-CPU interface, due to the pending command being executed by the CPU. Any new command should be issued only after DVF = 0.

There are two reasons that could cause a command to take too long to be executed, measured in terms of the serial communication rate. Either the BDC clock frequency is much faster if compared to the CPU clock frequency, or the CPU is accessing a slow memory, which would cause suspend cycles to occur. The hardware handshake protocol is appropriate for both situations, but the host could also decide to use the software handshake protocol instead. In this case, if the DVF bit is at logic 1, there is a BDC command pending at the BDC-CPU interface. The host controller should monitor the DVF bit and wait until it is at logic 0 to be able to issue a new command that requires CPU execution. Note that the WSF BDCSCR bit should be at logic 0 in this case. However, if WSF bit was at logic 1, the host should assume the last command failed due to a WAIT or STOP instruction being executed by the CPU. In this case, the host controller should enable background mode and then issue a BACKGROUND command to put the CPU into background mode. After that, new commands could be issued, including those that require CPU execution.

Figure 18-9 shows a SYNC command aborting a READ_BYTE. Note that after the command is aborted, a new command could be issued by the host computer.

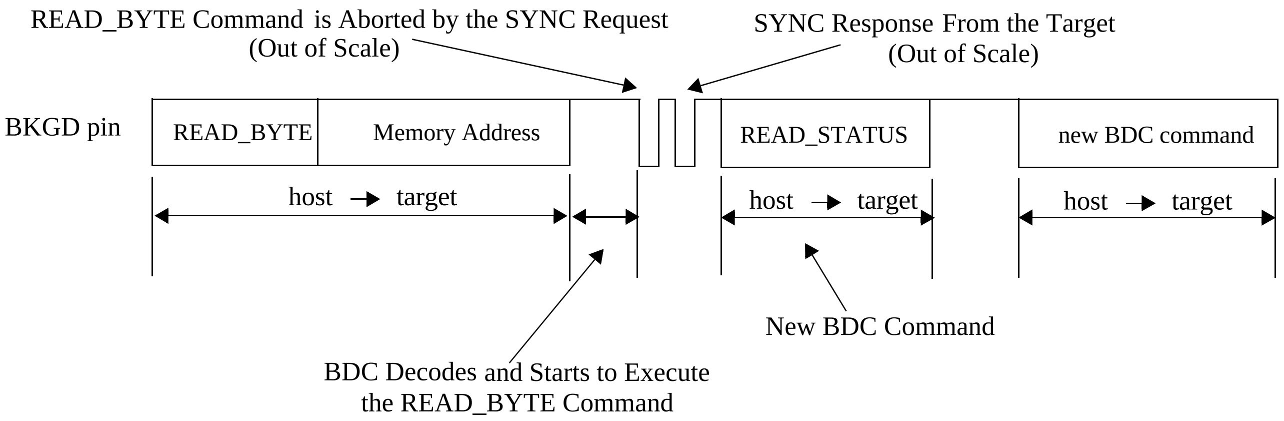


**Figure 18-9. ACK Abort Procedure at the Command Level**

Figure 18-10 shows a conflict between the ACK pulse and the SYNC request pulse. This conflict could occur if a POD device is connected to the target BKGD pin and the target is already in debug mode. Consider that the target CPU is executing a pending BDC command at the exact moment the POD is being connected to the BKGD pin. In this case an ACK pulse is issued along with the SYNC command. In this case there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. Because this is not a probable situation, the protocol does not prevent this conflict from happening.

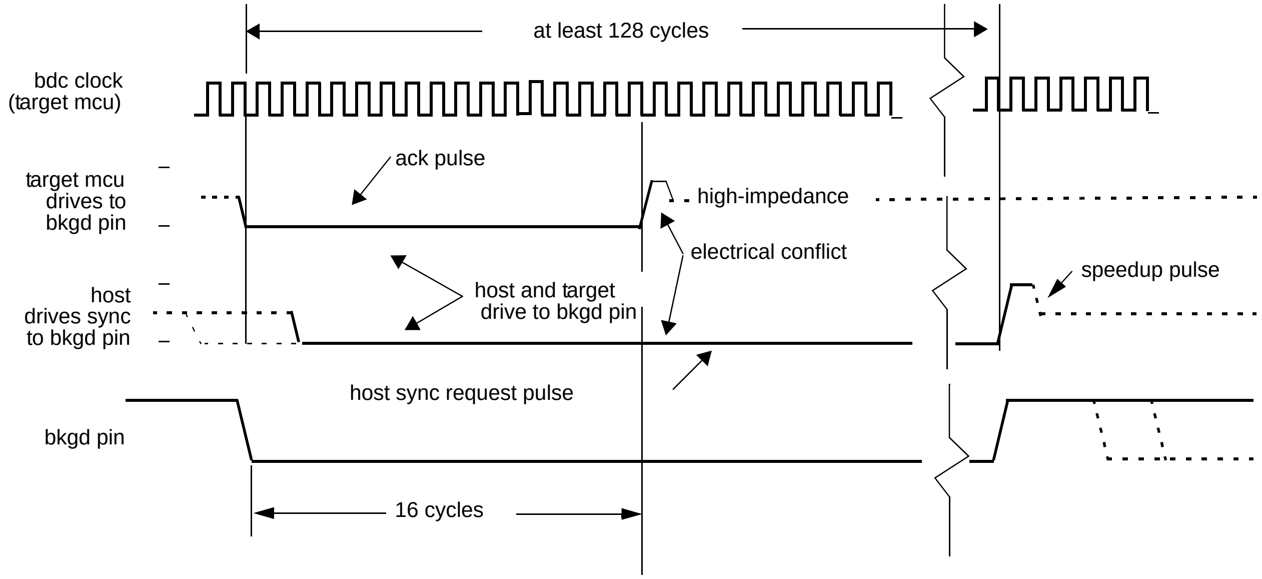


**Figure 18-10. ACK Pulse and SYNC Request Conflict**

The hardware handshake protocol is enabled by the ACK_ENABLE and disabled by the ACK_DISABLE BDC commands, providing backwards compatibility with old POD devices, which may not support the hardware handshake protocol. It also allows for new POD devices supporting the hardware handshake protocol to freely communicate with the target device without the need for waiting for the ACK pulse, if desired. The commands are described as follows:

- ACK_ENABLE - enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE - disables the ACK pulse protocol. In this case the host should verify the state of the DVF bit in the BDC Status and Control register to evaluate if there are pending commands and to check if the CPU changed to or from background mode.

The default state of the protocol, after reset, is hardware handshake protocol disabled.

The commands that do not require CPU execution, or that have the status register included in the retrieved bit stream, do not perform the hardware handshake protocol. Therefore, the target will not respond with an ACK pulse for those commands even if the hardware protocol was enabled. The commands are: READ_STATUS, WRITE_CONTROL, WRITE_BWS, READ_BWS, READ_NWS, WRITE_NWS, WRITE_BKPT, READ_BKPT, READ_LAST and ACK_DISABLE.

### NOTE

> The TAGGO command does not have the ACK pulse as a response. Except by the ACK pulse, this command is equivalent to the GO command. It is being implemented for compatibility with previous BDC versions. The HCS08 core does not provide support for external tag using the BKGD pin.

Only commands that require CPU execution perform the hardware handshake protocol. These commands are: WRITE_BYTE, READ_BYTE, WRITE_NEXT, READ_NEXT, WRITE_A, READ_A, WRITE_CCR, READ_CCR, WRITE_SP, READ_SP, WRITE_HX, READ_HX, WRITE_PC, READ_PC. An exception is the ACK_ENABLE, which does not require CPU execution but responds with the ACK pulse. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case the ACK_ENABLE command is ignored by the target, because it is not recognized as a valid command.

The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

The GO_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used if the host wants to trace if a breakpoint match had occurred which caused the CPU to enter background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a BDC Breakpoint match, or an external force/tag, or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TAGGO command is equivalent to the GO command, but will not have an ACK pulse as a response. This command is being kept for backwards compatibility reasons. It is expected that the GO command to be used instead.

## 18.2.5.5 Serial Communication Time-out

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target will keep waiting for a rising edge on BKGD to answer the SYNC request pulse. If the rising edge is not detected, the target will keep waiting forever, without any time-out limit.

Consider now the case where the host returns BKGD to logic one before 128 cycles. This is interpreted as a valid bit transmission, and not as a SYNC request. The target will keep waiting for another falling edge marking the start of a new bit. If, however, a new falling edge is not detected by the target within 512 clock cycles since the last falling edge, a time-out occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset.

If a read command is issued but the data is not retrieved within 512 serial clock cycles a soft-reset will occur causing the command to be disregarded. The data is not available for retrieving after the time-out had occurred. This is the expected behavior if the handshake protocol is not enabled. However, consider the behavior where the BDC is running in a frequency much greater than the CPU frequency. In this case, the command could time-out before the data is ready to be retrieved. To allow the data to be retrieved even with a large clock frequency mismatch, between BDC and CPU, when the hardware handshake protocol is enabled the time-out between a read command and the data retrieve is disabled. Therefore, the host could wait for more then 512 serial clock cycles and still be able to retrieve the data from an issued read command. However, after the handshake pulse is issued, the time-out feature is re-activated, meaning that the target will time-out after 512 clock cycles. Therefore, the host needs to retrieve the data within a 512 serial clock cycles time frame after the ACK pulse had been issued. After that period the read command is discarded and the data is no longer available for retrieve. Any neg-edge in the BKGD pin after the time-out period is considered to be a new command or a SYNC request.

## NOTE

Whenever a partially issued command, or partially retrieved data had occurred, the time-out in the serial communication is active, meaning that, if a time frame higher than 512 serial clock cycles is observed between two consecutive negedges and the command being issued or data being retrieved is not completed, a soft-reset will occur causing the partially received command or data retrieve to be disregarded. The next negedge in the BKGD pin, after the soft-reset had occurred, is considered by the target as the start of a new BDC command, or the start of a SYNC request pulse.

## 18.3 Register Definition

### 18.3.1 BDC Registers

The BDC provides two registers that can be accessed and modified with dedicated BDC commands: the BDC Status and Control Register (BDCSCR) and the BDC Breakpoint Register (BDCBKPT). These registers are not located in the memory space of the MCU (i.e., do not have addresses) and cannot be accessed by user programs. They can only be accessed by the host computer through the serial single-wire debug interface (BKGD pin).

#### 18.3.1.1 BDC Status and Control Register

**BDCSCR**

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| R | ENBDM | BDMACT | BKPTEN | FTS | CLKSW | WS | WSF | DVF |
| W | | ▨ | | | | ▨ | ▨ | ▨ |
| Normal Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset in BDM | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

▨ = Unimplemented

**Figure 18-11. BDC Status and Control Register (BDCSCR)**

**Table 18-1. BDCSCR Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ENBDM | Enable Background Debug Mode. This is a read/write control bit. The CPU may enter background debug mode only when this bit is set to logic one. If the CPU is placed in background debug mode, this control bit cannot be written until background debug mode is exited.<br>1 = CPU is allowed to enter background debug mode.<br>0 = CPU is not allowed to enter background debug mode. |
| 6<br>BDMACT | Background Debug Mode Active. This is a read-only status bit. This status bit is controlled by the CPU, and indicates if background debug mode is active. When this bit is set to logic one the BDMEN bit cannot not be written by WRITE_CONTROL commands. Upon reset, while the CPU is executing the reset sequence, this bit is a logic zero. If, however, during reset the BKGD pin was held low commanding “BDM out of reset”, soon after the reset sequence finishes, the CPU will enter background mode (provided no higher priority exception is present) and the BDMACT bit will be set to logic one.<br>1 = CPU is in background debug mode.<br>0 = CPU is not in background debug mode. |
| 5<br>BKPTEN | Breakpoint Enable bit. This is a read/write control bit. If this bit is set to logic one, the BDC breakpoint functionality is enabled, provided that BDMEN is also set. If this bit is set to logic zero, the BDC breakpoint functionality is disabled and the FTS control bit and the BDCBKPT match register are ignored.<br>1 = BDC breakpoint functionality enabled.<br>0 = BDC breakpoint functionality disabled. |

Table 18-1. BDCSCR Field Descriptions (continued)

| Field | Description |
|---|---|
| 4<br>FTS | Force/Tag Select bit. This is a read/write control bit. If this bit is set to logic one, a breakpoint is requested whenever a valid address in the CPU address bus matches the value in the BDCBKPT register. If this bit is set to logic zero, a match between the address of a fetched opcode and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the top of the instruction pipe unit, the CPU enters background debug mode rather than executing the tagged opcode.<br>1 = Breakpoint match forces background debug mode at the next instruction boundary (provided no higher priority events are present.)<br>0 = Tag opcode at breakpoint address and enter background debug mode if the CPU attempts to execute that instruction (provided no higher priority event is present.) |
| 3<br>CLKSW | Clock Switch bit. This is a read/write control bit. This bit controls the bypass mode of the BDC Local Oscillator block. If it is set to logic zero the BDC Local Oscillator is engaged setting the rate for the serial communication to be the BDC Local Oscillator rate. If it is set to logic one, the BDC Local Oscillator is disengaged, setting the ext_lclk output clock signal to be connected to the ext_clk24 MCU clock signal. In this case the BDC serial rate is the CPU frequency.<br>1 = Communication frequency is ext_clk24.<br>0 = Communication frequency is ext_lclk (BDC Local Oscillator frequency). |
| 2<br>WS | Wait/Stop bit. This is a read-only status bit. This status bit is controlled by the CPU and indicates that the CPU is in Wait/Stop mode, or the CPU exited Wait/Stop mode into background debug mode via a BACKGROUND command.<br>1 = CPU is in Wait/Stop mode, or the CPU exited Wait/Stop mode into background debug mode via a BACKGROUND command.<br>0 = CPU is not in Wait/Stop mode. |
| 1<br>WSF | Wait/Stop Failure. This is a read-only status bit. This bit indicates that a foreground memory command failed because the CPU was in Wait or Stop mode or was about to enter one of these modes. The recovery strategy is to put the CPU in BDM active mode and re-issue the command in background mode.<br>1 = Foreground memory command failed because the CPU entered Wait of Stop mode.<br>0 = Foreground memory command did not conflict with Wait or Stop mode. |
| 0<br>DVF | Data Valid Failure. This is a read-only status bit. This bit indicates that a CPU command failed or is pending. The recovery strategy is to issue READ_LAST (for read accesses) or READ_STATUS (for write accesses) commands until the status information indicates the CPU command had executed successfully.<br>1 = CPU command failed.<br>0 = CPU command executed successfully. |

## NOTE

The DVF bit is asserted also by the CPU registers access commands. This behavior allows to execute the software handshake protocol when accessing CPU registers. The READ_LAST instruction should be used to access the last accessed CPU registers data, available after the DVF bit is cleared. Note that the READ_LAST instruction returns 16 bit data only. For 8-bit register read commands only the first byte transmitted should be considered as valid data, the second byte has no meaning in this case.

### 18.3.1.2 BDC Breakpoint Register

BDCBKPT

| | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-12. BDC Breakpoint 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. Breakpoints are normally set while the target MCU is in background debug mode before running the user application program. However, because READ_BKPT and WRITE_BKPT are foreground commands, they could be executed even while the user program is running.

## 18.3.2 DBG Registers

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

Note: For all registers below, consider: U = Unchanged, bit maintain its value after reset.

### 18.3.2.1 Debug Comparator A High Register (DBGCAH)

DBGCAH — Base + 0x0000

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| POR or non-end-run | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

**Figure 18-13. Debug Comparator A High Register (DBGCAH)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-2. DBGCAH Field Descriptions**

| Field | Description |
|---|---|
| Bits 15–8 | **Comparator A High Compare Bits** — The Comparator A High compare bits control whether Comparator A will compare the address bus bits [15:8] to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

#### 18.3.2.2 Debug Comparator A Low Register (DBGCAL)

DBGCAL | Base + 0x0001

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| POR or non-end-run | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

**Figure 18-14. Debug Comparator A Low Register**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-3. DBGCAL Field Descriptions**

| Field | Description |
|---|---|
| Bits 7–0 | **Comparator A Low Compare Bits** — The Comparator A Low compare bits control whether Comparator A will compare the address bus bits [7:0] to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

### 18.3.2.3 Debug Comparator B High Register (DBGCBH)

**DBGCBH** **Base + 0x0002**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

**Figure 18-15. Debug Comparator B High Register (DBGCBH)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-4. DBGCBH Field Descriptions**

| Field | Description |
|---|---|
| Bits 15–8 | **Comparator B High Compare Bits** — The Comparator B High compare bits control whether Comparator B will compare the address bus bits [15:8] to a logic 1 or logic 0. Not used in Full mode.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

### 18.3.2.4 Debug Comparator B Low Register (DBGCBL)

**DBGCBL** **Base + 0x0003**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

**Figure 18-16. Debug Comparator B Low Register (DBGCBL)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-5. DBGCBL Field Descriptions**

| Field | Description |
|---|---|
| Bits 7–0 | **Comparator B Low Compare Bits** — The Comparator B Low compare bits control whether Comparator B will compare the address bus or data bus bits [7:0] to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0, compares to data if in Full mode<br>1 Compare corresponding address bit to a logic 1, compares to data if in Full mode |

### 18.3.2.5 Debug Comparator C High Register (DBGCCH)

**DBGCCCH** **Base + 0x0004**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

**Figure 18-17. Debug Comparator C High Register (DBGCCH)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-6. DBGCCH Field Descriptions**

| Field | Description |
|---|---|
| Bits 15–8 | **Comparator C High Compare Bits** — The Comparator C High compare bits control whether Comparator C will compare the address bus bits [15:8] to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

### 18.3.2.6 Debug Comparator C Low Register (DBGCCL)

**DBGCCCL** **Base + 0x0005**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

**Figure 18-18. Debug Comparator C Low Register (DBGCCL)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-7. DBGCCL Field Descriptions**

| Field | Description |
|---|---|
| Bits 7–0 | **Comparator C Low Compare Bits** — The Comparator C Low compare bits control whether Comparator C will compare the address bus bits [7:0] to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

### 18.3.2.7 Debug FIFO High Register (DBGFH)

**DBGFH** **Base + 0x0006**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

▨ = Unimplemented or Reserved

**Figure 18-19. Debug FIFO High Register (DBGFH)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-8. DBGFH Field Descriptions**

| Field | Description |
|---|---|
| Bits 15–8 | **FIFO High Data Bits** — The FIFO High data bits provide access to bits [15:8] of data in the FIFO. This register is not used in event only modes and will read a $00 for valid FIFO words. |

### 18.3.2.8 Debug FIFO Low Register (DBGFL)

**DBGFL** **Base + 0x0007**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | U | U | U | U | U |

▨ = Unimplemented or Reserved

**Figure 18-20. Debug FIFO Low Register (DBGFL)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-9. DBGFL Field Descriptions**

| Field | Description |
|---|---|
| Bits 7–0 | **FIFO Low Data Bits** — The FIFO Low data bits contain the least significant byte of data in the FIFO. When reading FIFO words, read DBGFX and DBGFH before reading DBGFL because reading DBGFL causes the FIFO pointers to advance to the next FIFO location. In event-only modes, there is no useful information in DBGFX and DBGFH so it is not necessary to read them before reading DBGFL. |

## 18.3.2.9 Debug Comparator A Extension Register (DBGCAX)

**DBGCAX** **Base + 0x0008**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RWAEN | RWA | PAGSEL | 0 | 0 | 0 | 0 | Bit 16 |
| W | | | | ▨ | ▨ | ▨ | ▨ | |
| R | RWAEN | RWA | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | 0 | 0 | 0 | 0 | U |

▨ = Unimplemented or Reserved

**Figure 18-21. Debug Comparator A Extension Register (DBGCAX)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-10. DBGCAX Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RWAEN | **Read/Write Comparator A Enable Bit** — The RWAEN bit controls whether read or write comparison is enabled for Comparator A.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 6<br>RWA | **Read/Write Comparator A Value Bit** — The RWA bit controls whether read or write is used in compare for Comparator A. The RWA bit is not used if RWAEN = 0.<br>0 Write cycle will be matched<br>1 Read cycle will be matched |

Table 18-10. DBGCAX Field Descriptions (continued)

| Field | Description |
|---|---|
| 5<br>PAGSEL | **Comparator A Page Select Bit** — This PAGSEL bit controls whether Comparator A will be qualified with the internal signal (mmu_ppage_sel) that indicates an extended access through the PPAGE mechanism. When mmu_ppage_sel = 1, the 17-bit core address is a paged program access, and the 17-bit core address is made up of PPAGE[2:0]:addr[13:0]. When mmu_ppage_sel = 0, the 17-bit core address is either a 16-bit CPU address with a leading 0 in bit 16, or a 17-bit linear address pointer value.<br>0 Match qualified by mmu_ppage_sel = 0 so address bits [16:0] correspond to a 17-bit CPU address with a leading zero at bit 16, or a 17-bit linear address pointer address{,}<br>1 Match qualified by mmu_ppage_sel = 1 so address bits [16:0] compare to flash memory address made up of PPAGE[2:0]:addr[13:0]{,} |
| 0<br>Bit 16 | **Comparator A Extended Address Bit 16 Compare Bit** — The Comparator A bit 16 compare bit controls whether Comparator A will compare the core address bus bit 16 to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

## 18.3.2.10 Debug Comparator B Extension Register (DBGCBX)

DBGCBX

Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RWBEN | RWB | PAGSEL | 0 | 0 | 0 | 0 | Bit 16 |
| W | | | | ▨ | ▨ | ▨ | ▨ | |
| R | RWBEN | RWB | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | 0 | 0 | 0 | 0 | U |

▨ = Unimplemented or Reserved

Figure 18-22. Debug Comparator B Extension Register (DBGCBX)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

### Table 18-11. DBGCBX Field Descriptions

| Field | Description |
|---|---|
| 7<br>RWBEN | **Read/Write Comparator B Enable Bit** — The RWBEN bit controls whether read or write comparison is enabled for Comparator B. In full modes, RWAEN and RWA are used to control comparison of R/W and RWBEN is ignored.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 6<br>RWB | **Read/Write Comparator B Value Bit** — The RWB bit controls whether read or write is used in compare for Comparator B. The RWB bit is not used if RWBEN = 0. In full modes, RWAEN and RWA are used to control comparison of R/W and RWB is ignored.<br>0 Write cycle will be matched<br>1 Read cycle will be matched |
| 5<br>PAGSEL | **Comparator B Page Select Bit** — This PAGSEL bit controls whether Comparator B will be qualified with the internal signal (mmu_papge_sel) that indicates an extended access through the PPAGE mechanism. When mmu_ppage_sel = 1, the 17-bit core address is a paged program access, and the 17-bit core address is made up of PPAGE[2:0]:addr[13:0]. When mmu_papge_sel = 0, the 17-bit core address is either a 16-bit CPU address with a leading 0 in bit 16, or a 17-bit linear address pointer value. This bit is not used in full modes where comparator B is used to match the data value.<br>0 Match qualified by mmu_ppage_sel = 0 so address bits [16:0] correspond to a 17-bit CPU address with a leading zero at bit 16, or a 17-bit linear address pointer address{,}<br>1 Match qualified by mmu_ppage_sel = 1 so address bits [16:0] compare to flash memory address made up of PPAGE[2:0]:addr[13:0]{,} |
| 0<br>Bit 16 | **Comparator B Extended Address Bit 16 Compare Bit** — The Comparator B bit 16 compare bit controls whether Comparator B will compare the core address bus bit 16 to a logic 1 or logic 0. This bit is not used in full modes where comparator B is used to match the data value.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

### 18.3.2.11 Debug Comparator C Extension Register (DBGCCX)

**DBGCCX** **Base + 0x000A**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RWCEN | RWC | PAGSEL | 0 | 0 | 0 | 0 | Bit 16 |
| W | | | | ▨ | ▨ | ▨ | ▨ | |
| R | RWCEN | RWC | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | U | 0 | 0 | 0 | 0 | U |

▨ = Unimplemented or Reserved

**Figure 18-23. Debug Comparator C Extension Register (DBGCCX)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-12. DBGCCX Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RWCEN | **Read/Write Comparator C Enable Bit** — The RWCEN bit controls whether read or write comparison is enabled for Comparator C{dbg_C_brk_rw,dbg_C_freecyc}.<br>0 Read/Write is not used in comparison{dbg_C_brk_rw,dbg_C_freecyc}<br>1 Read/Write is used in comparison{dbg_C_brk_rw,dbg_C_freecyc} |
| 6<br>RWC | **Read/Write Comparator C Value Bit** — The RWC bit controls whether read or write is used in compare for Comparator C{dbg_C_brk_rw,dbg_C_freecyc}. The RWC bit is not used if RWCEN = 0{dbg_C_brk_rw,dbg_C_freecyc}.<br>0 Write cycle will be matched{dbg_C_brk_rw,dbg_C_freecyc}<br>1 Read cycle will be matched{dbg_C_brk_rw,dbg_C_freecyc} |
| 5<br>PAGSEL | **Comparator C Page Select Bit** — This PAGSEL bit controls whether Comparator C will be qualified with the internal signal (mmu_papge_sel) that indicates an extended access through the PPAGE mechanism. When mmu_ppage_sel = 1, the 17-bit core address is a paged program access, and the 17-bit core address is made up of PPAGE[2:0]:addr[13:0]. When mmu_papge_sel = 0, the 17-bit core address is either a 16-bit CPU address with a leading 0 in bit 16, or a 17-bit linear address pointer value.<br>0 Match qualified by mmu_ppage_sel = 0 so address bits [16:0] correspond to a 17-bit CPU address with a leading zero at bit 16, or a 17-bit linear address pointer address{,}<br>1 Match qualified by mmu_ppage_sel = 1 so address bits [16:0] compare to flash memory address made up of PPAGE[2:0]:addr[13:0]{,} |
| 0<br>Bit 16 | **Comparator C Extended Address Bit 16 Compare Bit** — The Comparator C bit 16 compare bit controls whether Comparator C will compare the core address bus bit 16 to a logic 1 or logic 0.<br>0 Compare corresponding address bit to a logic 0<br>1 Compare corresponding address bit to a logic 1 |

### 18.3.2.12 Debug FIFO Extended Information Register (DBGFX)

DBGFX

Base + 0x000B

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | PPACC | 0 | 0 | 0 | 0 | 0 | 0 | Bit 16 |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR<br>or non-<br>end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset<br>end-run[1] | U | 0 | 0 | 0 | 0 | 0 | 0 | U |

▨ = Unimplemented or Reserved

**Figure 18-24. Debug FIFO Extended Information Register (DBGFX)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

**Table 18-13. DBGFX Field Descriptions**

| Field | Description |
|---|---|
| 7<br>PPACC | **PPAGE Access Indicator Bit** — This bit indicates whether the captured information in the current FIFO word is associated with an extended access through the PPAGE mechanism or not{,,}. This is indicated by the internal signal mmu_ppage_sel which is 1 when the access is through the PPAGE mechanism.<br>0 The information in the corresponding FIFO word is event-only data or an unpaged 17-bit CPU address with bit-16 = 0<br>1 The information in the corresponding FIFO word is a 17-bit flash address with PPAGE[2:0] in the three most significant bits and CPU address[13:0] in the 14 least significant bits |
| 0<br>Bit 16 | **Extended Address Bit 16** — This bit is the most significant bit of the 17-bit core address |

## 18.3.2.13 Debug Control Register (DBGC)

**DBGC** **Base + 0x000C**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DBGEN | ARM | TAG | BRKEN | 0 | 0 | 0 | LOOP1 |
| W | | | | | ▨ | ▨ | ▨ | |
| POR or non-end-run | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | 0 | U | 0 | 0 | 0 | 0 | U |

▨ = Unimplemented or Reserved

**Figure 18-25. Debug Control Register (DBGC)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the ARM and BRKEN bits are cleared but the remaining control bits in this register do not change after reset.

**Table 18-14. DBGC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DBGEN | **DBG Module Enable Bit** — The DBGEN bit enables the DBG module. The DBGEN bit is forced to zero and cannot be set if the MCU is secure{dbg_secure}.<br>0 DBG not enabled{,dbg_reg}<br>1 DBG enabled{,dbg_reg} |
| 6<br>ARM | **Arm Bit** — The ARM bit controls whether the debugger is comparing and storing data in FIFO. See Section 18.4.2.4.2, "Arming the DBG Module" for more information.<br>0 Debugger not armed<br>1 Debugger armed |
| 5<br>TAG | **Tag or Force Bit** — The TAG bit controls whether a debugger or comparator C breakpoint will be requested as a tag or force breakpoint to the CPU. The TAG bit is not used if BRKEN = 0{,dbg_C_brk_tag_force,}.<br>0 Force request selected<br>1 Tag request selected |
| 4<br>BRKEN | **Break Enable Bit** — The BRKEN bit controls whether the debugger will request a breakpoint to the CPU at the end of a trace run, and whether comparator C will request a breakpoint to the CPU<br>0 CPU break request not enabled<br>1 CPU break request enabled |
| 0<br>LOOP1 | **Select LOOP1 Capture Mode** — This bit selects either normal capture mode or LOOP1 capture mode. LOOP1 is not used in event-only modes.<br>0 Normal operation - capture COF events into the capture buffer FIFO<br>1 LOOP1 capture mode enabled. When the conditions are met to store a COF value into the FIFO, compare the current COF address with the address in comparator C. If these addresses match, override the FIFO capture and do not increment the FIFO count. If the address does not match comparator C, capture the COF address, including the PPACC indicator, into the FIFO and into comparator C. |

### 18.3.2.14 Debug Trigger Register (DBGT)

DBGT Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TRGSEL | BEGIN | 0 | 0 | TRG | | | |
| W[2] | | | ▨ | ▨ | | | | |
| POR or non-end-run | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | U | U | 0 | 0 | U | U | U | U |

▨ = Unimplemented or Reserved

**Figure 18-26. Debug Trigger Register (DBGT)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the control bits in this register do not change after reset.

[2] The DBG trigger register (DBGT) can not be changed unless ARM=0{dbg_reg}.

**Table 18-15. DBGT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TRGSEL | **Trigger Selection Bit** — The TRGSEL bit controls the triggering condition for the comparators. See Section 18.4.2.4, "Trigger Break Control (TBC)" for more information.<br>0 Trigger on any compare address access<br>1 Trigger if opcode at compare address is executed |
| 6<br>BEGIN | **Begin/End Trigger Bit** — The BEGIN bit controls whether the trigger begins or ends storing of data in FIFO.<br>0 Trigger at end of stored data<br>1 Trigger before storing data |
| 3–0<br>TRG | **Trigger Mode Bits** — The TRG bits select the trigger mode of the DBG module as shown in Table 18-16. |

**Table 18-16. Trigger Mode Encoding**

| TRG Value | Meaning |
|---|---|
| 0000 | A Only |
| 0001 | A Or B |
| 0010 | A Then B |
| 0011 | Event Only B |
| 0100 | A Then Event Only B |
| 0101 | A And B (Full Mode) |
| 0110 | A And Not B (Full mode) |
| 0111 | Inside Range |
| 1000 | Outside Range |

**Table 18-16. Trigger Mode Encoding (continued)**

| TRG Value | Meaning |
|---|---|
| 1001<br>↓<br>1111 | No Trigger |

**NOTE**

The DBG trigger register (DBGT) can not be changed unless ARM=0{dbg_reg}.

### 18.3.2.15 Debug Status Register (DBGS)

DBGS Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AF | BF | CF | 0 | 0 | 0 | 0 | ARMF |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Reset end-run[1] | U | U | U | 0 | 0 | 0 | 0 | 0 |

▨ = Unimplemented or Reserved

**Figure 18-27. Debug Status Register (DBGS)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, ARMF gets cleared by reset but AF, BF, and CF do not change after reset.

**Table 18-17. DBGS Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AF | **Trigger A Match Bit** — The AF bit indicates if Trigger A match condition was met since arming.<br>0 Comparator A did not match<br>1 Comparator A match |
| 6<br>BF | **Trigger B Match Bit** — The BF bit indicates if Trigger B match condition was met since arming.<br>0 Comparator B did not match<br>1 Comparator B match |
| 5<br>CF | **Trigger C Match Bit** — The CF bit indicates if Trigger C match condition was met since arming.<br>0 Comparator C did not match<br>1 Comparator C match |
| 0<br>ARMF | **Arm Flag Bit** — The ARMF bit indicates whether the debugger is waiting for trigger or waiting for the FIFO to fill. While DBGEN = 1, this status bit is a read-only image of the ARM bit in DBGC. See Section 18.4.2.4.2, “Arming the DBG Module” for more information.<br>0 Debugger not armed<br>1 Debugger armed |

## 18.3.2.16 Debug Count Status Register (DBGCNT)

**DBGCNT** **Base + 0x000F**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CNT | | | |
| W | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| POR or non-end-run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset end-run[1] | 0 | 0 | 0 | 0 | U | U | U | U |

▨ = Unimplemented or Reserved

**Figure 18-28. Debug Count Status Register (DBGCNT)**

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the CNT[3:0] bits do not change after reset.

**Table 18-18. DBGS Field Descriptions**

| Field | Description |
|---|---|
| 3–0<br>CNT | **FIFO Valid Count Bits** — The CNT bits indicate the amount of valid data stored in the FIFO. Table 18-19 shows the correlation between the CNT bits and the amount of valid data in FIFO. The CNT will stop after a count to eight even if more data is being stored in the FIFO. The CNT bits are cleared when the DBG module is armed, and the count increments each time a new word is captured into the FIFO. The host development system is responsible for checking the value in CNT[3:0] and reading the correct number of words from the FIFO because the count does not decrement as data is read out of the FIFO at the end of a trace run. |

**Table 18-19. CNT Bits**

| CNT Value | Meaning |
|---|---|
| 0000 | No data valid |
| 0001 | 1 word valid |
| 0010 | 2 words valid |
| 0011 | 3 words valid |
| 0100 | 4 words valid |
| 0101 | 5 words valid |
| 0110 | 6 words valid |
| 0111 | 7 words valid |
| 1000 | 8 words valid |

# 18.4 Functional Description

## 18.4.1 BDC Functional Description

### 18.4.1.1 Background Debug Mode

The Background Debug Controller provides five methods to enter background debug mode provided the mode is enabled via the ENBDM bit. Each of these methods is described below.

#### 18.4.1.1.1 BACKGROUND command

The usual way to get into Background Debug Mode, after the mode is enabled, is by issuing a BACKGROUND command from the host. This causes the BDC to issue a request to the CPU to enter background debug mode. This request is handled by the CPU at instruction boundaries according to the event priority.

The request to enter background debug mode interrupts the normal program flow and forces the CPU to execute a BGND instruction. Upon entering background debug mode, the CPU sets the BDMACT bit in the BDC status and control register. While in background debug mode, the ENBDM bit in the BDC status and control register cannot be cleared through WRITE_CONTROL commands. Both foreground and background commands can be executed while the CPU is in background debug mode. However, only foreground commands can be executed in normal mode (i.e., user code running). Normal program flow can be resumed with a GO, TRACE1, TAGGO or GO_UNTIL command. Execution starts with the instruction that would have been executed had the BGND instruction not been forced, provided the program counter is not altered while in background debug mode and no higher priority events are pending.

If the hardware handshake protocol is enabled, an ACK pulse is issued by the target as soon as the target CPU enters background mode due to a BACKGROUND BDC instruction. The handshake abort procedure should be used if the host decides to send a new BDC command before the ACK pulse is issued by the target.

#### 18.4.1.1.2 Hardware Breakpoint

The second method of entering background mode consists of using the hardware breakpoint functionality of the BDC. To use the hardware breakpoint, the user must enable it by writing a logic one to the BKPTEN bit in the BDCSCR register. This can be accomplished by issuing a WRITE_CONTROL command from the host. Note that the ENBDM bit in the BDCSCR register must be a logic one to use the breakpoint functionality. If ENBDM is a logic zero, the hardware breakpoint will not issue requests to the CPU regardless of the value of BKPTEN.

The user can set a breakpoint address in the BDCBKPT register by issuing a WRITE_BKPT command. The contents of the BDCBKPT register can be read by issuing the complementary READ_BKPT command. The breakpoint circuitry of the BDC requests the CPU to enter background debug mode whenever the contents of the BDCBKPT register match a valid address in the address bus. This request can be generated in two different ways depending on the configuration of the breakpoint functionality.

The user should write a logic one to the FTS bit when he wishes to enter background debug mode as a result of any valid access (read or write) to the breakpoint address. Under this configuration, if a valid address in the address bus matches the contents of the BDCBKPT register, the BDC requests the CPU to enter background mode. As described before, this forces the CPU to execute a BGND instruction on an instruction boundary if no higher priority events are pending.

Alternatively, the user should write a logic zero to the FTS bit when he wishes to enter background debug mode instead of executing a specific instruction in the program sequence. Under this configuration, if the address of a program read matches the contents of the BDCBKPT register, the BDC indicates to the CPU that it should enter background debug mode when the current instruction in the read data bus reaches the top of the instruction pipe. This is referred to as instruction tagging. If the data in the read data bus is tagged but is not an instruction (due to user error) the request is ignored.

Note that while background debug mode is active, the breakpoint functionality is disabled. This implies that while background debug mode is active the breakpoint functionality will not issue requests to the CPU.

As in the first case, normal program flow can be resumed with a GO, TRACE1, TAGGO or GO_UNTIL command. Execution starts with the instruction that would have been executed had the BGND instruction not been forced, provided the PC is not altered while in background debug mode and no higher priority events are pending.

If the hardware handshake protocol is enabled, an ACK pulse is issued by the target as soon as the target CPU enters background mode, provide the GO_UNTIL command had been used to put the CPU in normal mode. The SYNC command should be used if the host decides to send a new BDC command before the ACK pulse is issued by the target, in other words, before the breakpoint address is reached

**NOTE**

A match detected in the Breakpoint unit does not clear the DVF bit in the BDC Status and Control register. This bit only indicates a command had failed or is pending and is not modified by the Breakpoint unit. To evaluate if a breakpoint match had occurred, without using the hardware handshake protocol and the GO_UNTIL command, the host should check for the BDMACT BDCSCR bit to verify if it is at logic one, indicating that the CPU had entered Background Active mode.

#### 18.4.1.1.3 BGND Instruction

The third method consists of executing a BGND instruction. This can be accomplished by placing a BGND instruction in the normal program sequence, usually by issuing BDC foreground commands. The BGND instruction places the CPU in background debug mode. As in the first case, normal program flow can be resumed with a GO, TRACE1, TAGGO or GO_UNTIL command. However, in this case, execution starts with the instruction following BGND in the program sequence, provided the PC is not altered while in background debug mode and no higher priority events are pending.

Note that if the ENBDM bit is not set to logic one, the BGND instruction is treated as an illegal opcode. Also note that if the BGND instruction comes from secure memory, then the instruction is treated as an illegal opcode. This is to prevent attempts to break security by using background debug mode as follows:

1. Enter background debug Mode by issuing a BACKGROUND command.
2. Set the PC to an address in secure memory.

3. Issue a GO command.

If the data pointed by the PC is the opcode of a BGND instruction, then it will be executed as a secure opcode, defeating the security mechanism. To prevent this from happening, a BGND instruction from secure memory is treated as an illegal opcode.

If the hardware handshake protocol is enabled, an ACK pulse is issued by the target as soon as the target CPU enters background mode after executing the BGND instruction, provide the GO_UNTIL command had been used to put the CPU in normal mode. The hardware handshake abort procedure should be used if the host decides to send a new BDC command before the ACK pulse is issued by the target.

#### 18.4.1.1.4 External Force and Tag

The fourth and fifth methods of entering background debug mode is through the MCU's external force/tag capabilities. This method behaves essentially in the same way as the internal force/tagging capabilities. The main difference is that a force/tag may be issued when background debug mode is not enabled. In such a case, the CPU executes the SWI instruction instead of the BGND instruction and sets core_illegal_op_t4 status signal. In addition, the CPU sets core_force_exec_t4 or core_tag_exec_t4 to indicate whether a force or tag led to this condition. At the chip level these signals can be used to reset the core or used to detect that an external force or tag was responsible for the SWI flow.

If the hardware handshake protocol is enabled, an ACK pulse is issued by the target as soon as the target CPU enters background mode, provide the GO_UNTIL command had been used to put the CPU in normal mode. The hardware handshake abort procedure should be used, if the host decides to send a new BDC command before the ACK pulse is issued.

### 18.4.1.2 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communication protocol. BDC commands are divided in two classes: foreground commands and background commands. Foreground commands can be executed while the CPU is running application code. Background commands can be executed only if the core is in background debug mode. Figure 18-21 and Figure 18-22 describe the foreground and background commands. Table 18-20 explains the symbols used for the command structure. Subsequent paragraphs will describe each command in greater detail.

**Table 18-20. BDC Commands Legend**

| Symbol | Description |
|---|---|
| / | Separates parts of a BDM command |
| OP | 8-bit opcode (host-to-target) |
| d | Delay 16 clock cycles (based on the reference clock for serial communication) |
| AAAA | 16-bit address (host-to-target) |
| SS | 8 bits of read data from the BDC status and control register (target-to-host) |
| CC | 8 bits of write data to the BDC status and control register (host-to-target) |
| RB | 8 bits of read data (target-to-host) |

Table 18-20. BDC Commands Legend

| Symbol | Description |
|---|---|
| / | Separates parts of a BDM command |
| WB | 8 bits of write data (host-to-target) |
| RBKP | 16 bits of read data from the BDC breakpoint register (target-to-host) |
| WBKP | 16 bits of write data to the BDC breakpoint register (host-to-target) |
| RHWD | 16 bits of read data (target-to-host) |
| WHWD | 16 bits of write data (host-to-target) |

Table 18-21. BDC Foreground Commands

| Command | Opcode (Hex) | Command Structure[1] | Description |
|---|---|---|---|
| SYNC | None | None[2] | Request timed reference pulse |
| ACK_ENABLE | D5 | OP/d | Enable Handshake. Issues an ACK pulse after the command is executed. |
| ACK_DISABLE | D6 | OP/d | Disable Handshake. This command does not issues an ACK pulse. |
| BACKGROUND | 90 | OP/d | Enter Background Debug Mode if enabled. Issues an ACK pulse after the CPU had entered into BDM, if the hardware handshake protocol was enabled. If ENBDM = 0 the command is ignored and the ACK pulse is not issued. |
| READ_STATUS | E4 | OP/SS | Read the BDC status and control register (BDCSCR) |
| WRITE_CONTROL | C4 | OP/CC | Write to the BDCSCR register |
| READ_LAST | E8 | OP/SS/RB or OP/RHWD | Retrieves the BDCSCR and data from the last READ_BYTE. Retrieves ACC or CCR if last command was READ_A or READ_CCR and an extra byte that is not valid data. If the last command was READ_HX, READ_SP or READ_PC only the CPU register data is retrieved. |
| READ_BYTE | E0 | OP/AAAA/d/RB | Read from memory. Issues an ACK pulse when the data is ready to be retrieved, if the hardware handshake protocol was enabled. |
| WRITE_BYTE | C0 | OP/AAAA/WB/d | Write to memory. Issues an ACK when the data is written to the addressed memory location, if the hardware handshake protocol was enabled. |
| READ_BWS | E1 | OP/AAAA/d/SS/RB | Read from memory with BDC status |
| WRITE_BWS | C1 | OP/AAAA/WB/d/SS | Write to memory with BDC status |
| READ_BKPT | E2 | OP/RBKP | Read breakpoint address |
| WRITE_BKPT | C2 | OP/WBKP | Write breakpoint address |

[1] Commands begin with an 8-bit hexadecimal code in the host-to-target direction (MSB first).
[2] The SYNC command is a special operation which does not have a command code.

Table 18-22. BDC Background Commands

| Command | Opcode (Hex) | Command Structure | Description |
|---|---|---|---|
| GO | 08 | OP/d | Resume normal program flow. If in background mode issues an ACK pulse after the CPU exits BDM, if the hardware handshake protocol is enabled. |
| GO_UNTIL | 0C | OP/d | Resume normal program flow. If in background mode, issues an ACK pulse after the CPU enters back to BDM, if the hardware handshake protocol is enabled. |
| TRACE1 | 10 | OP/d | Execute one instruction in the normal program sequence then return to Background Debug Mode. If in background mode, issues an ACK pulse when the CPU enters back into BDM, after the user instruction is executed, if the hardware handshake protocol is enabled. |
| TAGGO | 18 | OP/d | Same as GO, except that this command will not issue an ACK pulse as a response. |
| READ_CCR | 69 | OP/d/RB | Read condition code register (CCR). If in background mode, issues an ACK pulse when the CCR data is ready to be retrieved if the hardware handshake protocol was enabled. |
| WRITE_CCR | 49 | OP/WB/d | Write condition code register (CCR). If in background mode, issues an ACK pulse after the CCR register is updated, provided the hardware handshake protocol is enabled. |
| READ_A | 68 | OP/d/RB | Read Accumulator (A). If in background mode, issues an ACK pulse when the ACC data is ready to be retrieved, provided the hardware handshake protocol is enabled. |
| WRITE_A | 48 | OP/WB/d | Write Accumulator (A). If in background mode, issues an ACK pulse when the ACC register is updated, provided the hardware handshake protocol is enabled. |
| READ_HX | 6C | OP/d/RHWD | Read H:X index register. If in background mode, issues an ACK pulse when the HX data is ready to be retrieved, provided the hardware handshake protocol is enabled. |
| WRITE_HX | 4C | OP/WHWD/d | Write H:X index register. If in background mode, issues an ACK pulse when the HX register is updated, provided the hardware handshake protocol is enabled. |
| READ_SP | 6F | OP/d/RHWD | Read stack pointer (SP). If in background mode, issues an ACK pulse when the SP data is ready to be retrieved, provided the hardware handshake protocol is enabled. |
| WRITE_SP | 4F | OP/WHWD/d | Write stack pointer (SP). If in background mode, issues an ACK pulse when the SP register is updated, provided the hardware handshake protocol is enabled. |
| READ_PC | 6B | OP/d/RHWD | Read program counter (PC). If in background mode, issues an ACK pulse when the PC data is ready to be retrieved, provided the hardware handshake protocol is enabled. |
| WRITE_PC | 4B | OP/WHWD/d | Write program counter (PC). If in background mode, issues an ACK pulse when the PC register is updated, provided the hardware handshake protocol is enabled. |

Table 18-22. BDC Background Commands (continued)

| Command | Opcode (Hex) | Command Structure | Description |
|---|---|---|---|
| READ_NEXT | 70 | OP/d/RB | Increment H:X by 1 and read from memory at H:X. If in background mode, issues an ACK pulse when the Memory data is ready to be retrieved, provided the hardware handshake protocol is enabled. |
| WRITE_NEXT | 50 | OP/WB/d | Increment H:X by 1 and write to memory at H:X. If in background mode, issues an ACK pulse when the Memory location is updated, provided the hardware handshake protocol is enabled. |
| READ_NWS | 71 | OP/d/SS/RB | Increment H:X by 1 and read from memory at H:X with BDC status |
| WRITE_NWS | 51 | OP/WB/d/SS | Increment H:X by 1 and write to memory at H:X with BDC status |

The serial communication could be set in two different rates: the ext_clk24, which is double the CPU bus frequency and ext_lclk, which is the independent Local Oscillator frequency source. These modes are defined by the CLKSW bit in the BDC Status and Control register. If CLKSW = 0 the Local Oscillator is engaged and the serial communication rate is defined by the Local Oscillator frequency. In this mode the serial communication rate is asynchronous related to the MCU bus clock. If CLKSW = 1, the Local Oscillator is disengaged, causing the ext_lclk to be derived from the ext_clk24 MCU clock signal. In this case the serial communication clock is synchronous related to the MCU bus clock.

### 18.4.1.2.1 Serial Communication Time-out

The host initiates a host-to-target serial transmission by generating a falling edge on the BKGD pin. If BKGD is kept low for more than 128 target clock cycles, the target understands that a SYNC command was issued. In this case, the target will keep waiting for a rising edge on BKGD to answer the SYNC request pulse. If the rising edge is not detected, the target will keep waiting forever, without any time-out limit.

Consider now the case where the host returns BKGD to logic one before 128 cycles. This is interpreted as a valid bit transmission, and not as a SYNC request. The target will keep waiting for another falling edge marking the start of a new bit. If, however, a new falling edge is not detected by the target within 512 clock cycles since the last falling edge, a time-out occurs and the current command is discarded without affecting memory or the operating mode of the MCU. This is referred to as a soft-reset.

If a read command is issued but the data is not retrieved within 512 serial clock cycles a soft-reset will occur causing the command to be disregarded. The data is not available for retrieving after the time-out had occurred. This is the expected behavior if the handshake protocol is not enabled. However, consider the behavior where the BDC is running in a frequency much greater than the CPU frequency. In this case, the command could time-out before the data is ready to be retrieved. To allow the data to be retrieved even with a large clock frequency mismatch, between BDC and CPU, when the hardware handshake protocol is enabled the time-out between a read command and the data retrieve is disabled. Therefore, the host could wait for more then 512 serial clock cycles and still be able to retrieve the data from an issued read command. However, after the handshake pulse is issued, the time-out feature is re-activated, meaning that the target will time-out after 512 clock cycles. Therefore, the host needs to retrieve the data within a 512 serial clock cycles time frame after the ACK pulse had been issued. After that period the read command is discarded and the data is no longer available for retrieve. Any negedge in the BKGD pin after the time-out period is considered to be a new command or a SYNC request.

Note that, whenever a partially issued command, or partially retrieved data had occurred, the time-out in the serial communication is active, meaning that, if a time frame higher than 512 serial clock cycles is observed between two consecutive negedges and the command being issued or data being retrieved is not completed, a soft-reset will occur causing the partially received command or data retrieve to be disregarded. The next negedge in the BKGD pin, after the soft-reset had occurred, is considered by the target as the start of a new BDC command, or the start of a SYNC request pulse.

## 18.4.2 DBG Functional Description

This section provides a complete functional description of the on-chip ICE system. The DBG module is enabled by setting the DBGEN bit in the DBGC register. Enabling the module allows the arming, triggering and storing of data in the FIFO. The DBG module is made up of three main blocks, the Comparators, Trigger Break Control logic and the FIFO.

### 18.4.2.1 Comparator

The DBG module contains three comparators, A, B, and C. Comparator A compares the core address bus with the address stored in the DBGCAX, DBGCAH, and DBGCAL registers. Comparator B compares the core address bus with the address stored in the DBGCBX, DBGCBH, and DBGCBL registers except in full mode, where it compares the data buses to the data stored in the DBGCBL register. Comparator C compares the core address bus with the address stored in the DBGCCX, DBGCCH, and DBGCCL registers. Matches on Comparators A, B, and C are signaled to the Trigger Break Control (TBC) block.

#### 18.4.2.1.1 RWA and RWAEN in Full Modes

In full modes ("A And B" and "A And Not B") RWAEN and RWA are used to select read or write comparisons for both comparators A and B. To select write comparisons and the write data bus in Full Modes set RWAEN=1 and RWA=0, otherwise read comparisons and the read data bus will be selected. The RWBEN and RWB bits are not used and will be ignored in Full Modes.

#### 18.4.2.1.2 Comparator C in LOOP1 Capture Mode

Normally comparator C is used as a third hardware breakpoint and is not involved in the trigger logic for the on-chip ICE system. In this mode, it compares the core address bus with the address stored in the DBGCCX, DBGCCH, and DBGCCL registers. However, in LOOP1 capture mode, comparator C is managed by logic in the DBG module to track the address of the most recent change-of-flow event that was captured into the FIFO buffer. In LOOP1 capture mode, comparator C is not available for use as a normal hardware breakpoint.

When the ARM and DBGEN bits are set to one in LOOP1 capture mode, comparator C value registers are cleared to prevent the previous contents of these registers from interfering with the LOOP1 capture mode operation. When a COF event is detected, the address of the event is compared to the contents of the DBGCCX, DBGCCH, and DBGCCL registers to determine whether it is the same as the previous COF entry in the capture FIFO. If the values match, the capture is inhibited to prevent the FIFO from filling up with duplicate entries. If the values do not match, the COF event is captured into the FIFO and the DBGCCX, DBGCCH, and DBGCCL registers are updated to reflect the address of the captured COF event. When comparator C is updated, the PAGSEL bit (bit-7 of DBGCCX) is updated with the PPACC value that is captured into the FIFO. This bit indicates whether the COF address was a paged 17-bit program address using the PPAGE mechanism (PPACC=1) or a 17-bit CPU address that resulted from an unpaged CPU access.

### 18.4.2.2 Breakpoints

A breakpoint request to the CPU at the end of a trace run can be created if the BRKEN bit in the DBGC register is set. The value of the BEGIN bit in DBGT register determines when the breakpoint request to the CPU will occur. If the BEGIN bit is set, begin-trigger is selected and the breakpoint request will not occur until the FIFO is filled with 8 words. If the BEGIN bit is cleared, end-trigger is selected and the breakpoint request will occur immediately at the trigger cycle.

When traditional hardware breakpoints from comparators A or B are desired, set BEGIN=0 to select an end-trace run and set the trigger mode to either 0x0 (A-only) or 0x1 (A OR B) mode.

There are two types of breakpoint requests supported by the DBG module, tag-type and force-type. Tagged breakpoints are associated with opcode addresses and allow breaking just before a specific instruction executes. Force breakpoints are not associated with opcode addresses and allow breaking at the next instruction boundary. The TAG bit in the DBGC register determines whether CPU breakpoint requests will be a tag-type or force-type breakpoints. When TAG=0, a force-type breakpoint is requested and it will take effect at the next instruction boundary after the request. When TAG=1, a tag-type breakpoint is registered into the instruction queue and the CPU will break if/when this tag reaches the head of the instruction queue and the tagged instruction is about to be executed.

#### 18.4.2.2.1 Hardware Breakpoints

Comparators A, B, and C can be used as three traditional hardware breakpoints whether the on-chip ICE real-time capture function is required or not. To use any breakpoint or trace run capture functions set DBGEN=1. BRKEN and TAG affect all three comparators. When BRKEN=0, no CPU breakpoints are enabled. When BRKEN=1, CPU breakpoints are enabled and the TAG bit determines whether the breakpoints will be tag-type or force-type breakpoints. To use comparators A and B as hardware breakpoints, set DBGT=0x81 for tag-type breakpoints and 0x01 for force-type breakpoints. This sets up an end-type trace with trigger mode “A OR B”.

Comparator C is not involved in the trigger logic for the on-chip ICE system.

### 18.4.2.3 Trigger Selection

The TRGSEL bit in the DBGT register is used to determine the triggering condition of the on-chip ICE system. TRGSEL applies to both trigger A and B except in the event only trigger modes. By setting the TRGSEL bit, the comparators will qualify a match with the output of opcode tracking logic. The opcode tracking logic is internal to each comparator and determines whether the CPU executed the opcode at the compare address. With the TRGSEL bit cleared a comparator match is all that is necessary for a trigger condition to be met.

NOTE

If the TRGSEL is set, the address stored in the comparator match address registers must be an opcode address for the trigger to occur.

### 18.4.2.4 Trigger Break Control (TBC)

The TBC is the main controller for the DBG module. Its function is to decide whether data should be stored in the FIFO based on the trigger mode and the match signals from the comparator. The TBC also determines whether a request to break the CPU should occur.

The TAG bit in DBGC controls whether CPU breakpoints are treated as tag-type or force-type breakpoints. The TRGSEL bit in DBGT controls whether a comparator A or B match is further qualified by opcode tracking logic. Each comparator has a separate circuit to track opcodes because the comparators could correspond to separate instructions that could be propagating through the instruction queue at the same time.

In end-type trace runs (BEGIN=0), when the comparator registers match, including the optional R/W match, this signal goes to the CPU break logic where BRKEN determines whether a CPU break is requested and the TAG control bit determines whether the CPU break will be a tag-type or force-type breakpoint. When TRGSEL is set, the R/W qualified comparator match signal also passes through the opcode tracking logic. If/when it propagates through this logic, it will cause a trigger to the ICE logic to begin or end capturing information into the FIFO. In the case of an end-type (BEGIN=0) trace run, the qualified comparator signal stops the FIFO from capturing any more information.

If a CPU breakpoint is also enabled, TAG and TRGSEL to agree so that the CPU break occurs at the same place in the application program as the FIFO stopped capturing information. If TRGSEL was 0 and TAG was 1 in an end-type trace run, the FIFO would stop capturing as soon as the comparator address matched, but the CPU would continue running until a TAG signal could propagate through the CPUs instruction queue which could take a long time in the case where changes of flow caused the instruction queue to be flushed. If TRGSEL was one and TAG was zero in an end-type trace run, the CPU would break before the comparator match signal could propagate through the opcode tracking logic to end the trace run.

In begin-type trace runs (BEGIN=1), the start of FIFO capturing is triggered by the qualified comparator signals, and the CPU breakpoint (if enabled by BRKEN=1) is triggered when the FIFO becomes full. Because this FIFO full condition does not correspond to the execution of a tagged instruction, it would not make sense to use TAG=1 for a begin-type trace run.

#### 18.4.2.4.1 Begin- and End-Trigger

The definition of begin- and end-trigger as used in the DBG module are as follows:

- Begin-trigger: Storage in FIFO occurs after the trigger and continues until 8 locations are filled.
- End-trigger: Storage in FIFO occurs until the trigger with the least recent data falling out of the FIFO if more than 8 words are collected.

#### 18.4.2.4.2 Arming the DBG Module

Arming occurs by enabling the DBG module by setting the DBGEN bit and by setting the ARM bit in the DBGC register. The ARM bit in the DBGC register and the ARMF bit in the DBGS register are cleared when the trigger condition is met in end-trigger mode or when the FIFO is filled in begin-trigger mode. In the case of an end-trace where DBGEN=1 and BEGIN=0, ARM and ARMF are cleared by any reset to end the trace run that was in progress. The ARMF bit is also cleared if ARM is written to zero or when the DBGEN bit is low. The TBC logic determines whether a trigger condition has been met based on the trigger mode and the trigger selection.

#### 18.4.2.4.3 Trigger Modes

The on-chip ICE system supports nine trigger modes. The trigger modes are encoded as shown in Table 18-16. The trigger mode is used as a qualifier for either starting or ending the storing of data in the FIFO. When the match condition is met, the appropriate flag AF or BF is set in DBGS register. Arming the DBG module clears the AF, BF, and CF flags in the DBGS register. In all trigger modes except for the event only modes change of flow addresses are stored in the FIFO. In the event only modes only the value on the data bus at the trigger event B comparator match address will be stored.

**A Only**

In the A Only trigger mode, if the match condition for A is met, the AF flag in the DBGS register is set.

**A Or B**

In the A Or B trigger mode, if the match condition for A or B is met, the corresponding flag(s) in the DBGS register are set.

**A Then B**

In the A Then B trigger mode, the match condition for A must be met before the match condition for B is compared. When the match condition for A or B is met, the corresponding flag in the DBGS register is set.

**Event Only B**

In the Event Only B trigger mode, if the match condition for B is met, the BF flag in the DBGS register is set. The Event Only B trigger mode is considered a begin-trigger type and the BEGIN bit in the DBGT register is ignored.

**A Then Event Only B**

In the A Then Event Only B trigger mode, the match condition for A must be met before the match condition for B is compared. When the match condition for A or B is met, the corresponding flag in the DBGS register is set. The A Then Event Only B trigger mode is considered a begin-trigger type and the BEGIN bit in the DBGT register is ignored.

**A And B (Full Mode)**

In the A And B trigger mode, Comparator A compares to the address bus and Comparator B compares to the data bus. In the A and B trigger mode, if the match condition for A and B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only B happens, no flags are set.

For Breakpoint tagging operation with an end-trigger type trace, only matches from Comparator A will be used to determine whether the Breakpoint conditions are met and Comparator B matches will be ignored.

### A And Not B (Full Mode)

In the A And Not B trigger mode, comparator A compares to the address bus and comparator B compares to the data bus. In the A And Not B trigger mode, if the match condition for A and Not B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only Not B occur no flags are set.

For Breakpoint tagging operation with an end-trigger type trace, only matches from Comparator A will be used to determine whether the Breakpoint conditions are met and Comparator B matches will be ignored.

### Inside Range, A $\leq$ address $\leq$ B

In the Inside Range trigger mode, if the match condition for A and B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only B occur no flags are set.

### Outside Range, address < A or address > B

In the Outside Range trigger mode, if the match condition for A or B is met, the corresponding flag in the DBGS register is set.

The four control bits BEGIN and TRGSEL in DBGT, and BRKEN and TAG in DBGC, determine the basic type of debug run as shown in Table 18-23. Some of the 16 possible combinations are not used (refer to the notes at the end of the table).

**Table 18-23. Basic Types of Debug Runs**

| BEGIN | TRGSEL | BRKEN | TAG | Type of Debug Run |
|---|---|---|---|---|
| 0 | 0 | 0 | x[1] | Fill FIFO until trigger address (No CPU breakpoint - keep running) |
| 0 | 0 | 1 | 0 | Fill FIFO until trigger address, then force CPU breakpoint |
| 0 | 0 | 1 | 1 | Do not use[2] |
| 0 | 1 | 0 | x[1] | Fill FIFO until trigger opcode about to execute (No CPU breakpoint - keep running) |
| 0 | 1 | 1 | 0 | Do not use[3] |
| 0 | 1 | 1 | 1 | Fill FIFO until trigger opcode about to execute (trigger causes CPU breakpoint) |
| 1 | 0 | 0 | x[1] | Start FIFO at trigger address (No CPU breakpoint - keep running) |
| 1 | 0 | 1 | 0 | Start FIFO at trigger address, force CPU breakpoint when FIFO full |
| 1 | 0 | 1 | 1 | Do not use[4] |
| 1 | 1 | 0 | x[1] | Start FIFO at trigger opcode (No CPU breakpoint - keep running) |
| 1 | 1 | 1 | 0 | Start FIFO at trigger opcode, force CPU breakpoint when FIFO full |
| 1 | 1 | 1 | 1 | Do not use[4] |

[1] When BRKEN = 0, TAG is do not care (x in the table).

[2] In end trace configurations (BEGIN = 0) where a CPU breakpoint is enabled (BRKEN = 1), TRGSEL should agree with TAG. In this case, where TRGSEL = 0 to select no opcode tracking qualification and TAG = 1 to specify a tag-type CPU breakpoint, the CPU breakpoint would not take effect until sometime after the FIFO stopped storing values. Depending on program loops or interrupts, the delay could be very long.

[3] In end trace configurations (BEGIN = 0) where a CPU breakpoint is enabled (BRKEN = 1), TRGSEL should agree with TAG. In this case, where TRGSEL = 1 to select opcode tracking qualification and TAG = 0 to specify a force-type CPU breakpoint, the CPU breakpoint would erroneously take effect before the FIFO stopped storing values and the debug run would not complete normally.

4 In begin trace configurations (BEGIN = 1) where a CPU breakpoint is enabled (BRKEN = 1), TAG should not be set to 1. In begin trace debug runs, the CPU breakpoint corresponds to the FIFO full condition which does not correspond to a taggable instruction fetch.

### 18.4.2.5 FIFO

The FIFO is an eight word deep FIFO. In all trigger modes except for event only, the data stored in the FIFO will be change of flow addresses. In the event only trigger modes only the data bus value corresponding to the event is stored. In event only trigger modes, the high byte of the valid data from the FIFO will always read a 0x00 and the extended information byte in DBGFX will always read 0x00.

#### 18.4.2.5.1 Storing Data in FIFO

In all trigger modes except for the event only modes, the address stored in the FIFO will be determined by the change of flow indicators from the core. The signal core_cof[1] indicates the current core address is the destination address of an indirect JSR or JMP instruction, or a RTS, RTC, or RTI instruction or interrupt vector and the destination address should be stored. The signal core_cof[0] indicates that a conditional branch was taken and that the source address of the conditional branch should be stored.

#### 18.4.2.5.2 Storing with Begin-Trigger

Storing with Begin-Trigger can be used in all trigger modes. After the DBG module is enabled and armed in the begin-trigger mode, data is not stored in the FIFO until the trigger condition is met. After the trigger condition is met the DBG module will remain armed until 8 words are stored in the FIFO. If the core_cof[1] signal becomes asserted, the current address is stored in the FIFO. If the core_cof[0] signal becomes asserted, the address registered during the previous last cycle is decremented by two and stored in the FIFO.

#### 18.4.2.5.3 Storing with End-Trigger

Storing with End-Trigger cannot be used in event-only trigger modes. After the DBG module is enabled and armed in the end-trigger mode, data is stored in the FIFO until the trigger condition is met. If the core_cof[1] signal becomes asserted, the current address is stored in the FIFO. If the core_cof[0] signal becomes asserted, the address registered during the previous last cycle is decremented by two and stored in the FIFO. When the trigger condition is met, the ARM and ARMF will be cleared and no more data will be stored. In non-event only end-trigger modes, if the trigger is at a change of flow address the trigger event will be stored in the FIFO.

#### 18.4.2.5.4 Reading Data from FIFO

The data stored in the FIFO can be read using BDM commands provided the DBG module is enabled and not armed (DBGEN=1 and ARM=0). The FIFO data is read out first-in-first-out. By reading the CNT bits in the DBGCNT register at the end of a trace run, the number of valid words can be determined. The FIFO data is read by optionally reading the DBGFX and DBGFH registers followed by the DBGFL register. Each time the DBGFL register is read the FIFO is shifted to allow reading of the next word however the count does not decrement. In event-only trigger modes where the FIFO will contain only the data bus values stored, to read the FIFO only DBGFL needs to be accessed.

The FIFO is normally only read while ARM and ARMF=0, however reading the FIFO while the DBG module is armed will return the data value in the oldest location of the FIFO and the TBC will not allow

# Appendix A
# IEEE 802.15.4 PHY Messaging Overview

## A.1 Introduction

To better understand the full capabilities of the MC13234/MC13237 Sequence Manager, it is necessary to have a basic understanding of the IEEE 802.15.4 PHY messaging protocol. This appendix is meant as a simple overview and is not intended to provide an in-depth understanding of the subject. For more information, the reader is directed to the IEEE 802.15.4 Standard, *“Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”, IEEE Std 802.15.4™-2006*

## A.2 Basic PHY Functions

At the 802.15.4 PHY level, all activities for any wireless node can be built from three basic functions:

- Clear Channel Assessment (CCA) / ED scan - this function involves enabling the radio receiver and scanning the selected 802.15.4 channel frequency for any energy present. The algorithm for detecting the energy can vary and the detected energy can be compared to a pre-determined level, but this activity is passive to the network.
- Packet Transmit - this function causes the radio transmitter to power-up and sent a packet formatted as described in Section 6.4.2, “MAC Frame Structures”.
- Packet Receive - this function causes the radio receiver to power-up and receive a packet that has been transmitted by another wireless node.

All the higher level protocol detailed here are built from these components.

## A.3 Non-beacon-Enabled versus Beacon-Enabled Communication

Without getting into the details of network architectures, communication at the over-the-air PHY level can be divided into non-beacon-enabled vs. beacon-enabled modes.

### A.3.1 Non-beacon-Enabled Communication

Non-beacon-enabled devices use a simple model for communication between devices (typically a coordinator and another network device):

- Only three of the MAC frame types are used - includes data frame, acknowledgement (ACK) frame, and MAC command frame. Beacon frames are not used.
- To send information, a data frame is sent from a source to a destination, and an ACK may be optionally used. The ACK frame uses a special short format, and if required, must be sent within a specified time limit. The ACK is typically used but always not required.

- There are no device-to-device timing or synchronization mechanisms.
- A standard packet exchange uses a single CCA cycle (assuming no interference) followed by the packet transmission. An ACK follows if appropriate.
- All access uses unslotted CSMA-CA access - that means that if a device during a CCA cycle finds the channel busy, its back off period is random and independent of all other devices.
- A simple, point-to-point unencumbered communication model

## A.3.2 Beacon-Enabled Communication

For lower latency and more deterministic communication especially with a larger number of nodes, a more sophisticated model is beacon-enabled communication. The protocol is built around the concept of synchronized timing between devices to determine when communication between the coordinator and its network devices is allowed.

Beacon-enabled communication is based on the use of a “Superframe” structure. Figure A-1 shows the most simple superframe structure. The coordinator sends out beacon frames out to the network devices on a periodic basis. In this simple model, the entire time between beacons is available for communication. The time available for communication is called the Contention Access Period (CAP) and is always divided into 16 equal time slots.

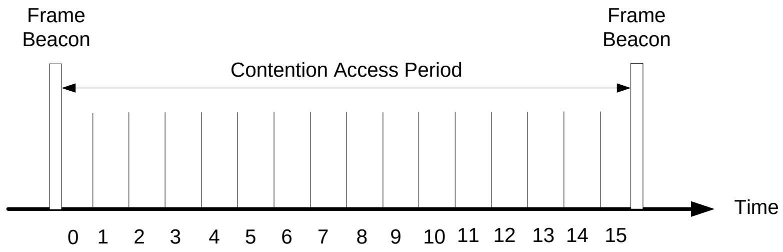


**Figure A-1. Simple Superframe Structure**

Devices wishing to access the coordinator use a slotted CSMA-CA channel access mechanism:

- All channel access is relative to the start of the beacon transmission.
- The back off slots of all devices within the PAN are aligned to the PAN coordinator and all back off slot lengths are standard at 20 symbols in length.
- Slotted access means that two CCA cycles are used for transmit - two CCA cycles (separated by a back off slot time) must both find an idle channel before the transmit can commence.
- Each time a device wishes to transmit - it locates the boundary of the next back off slot and then waits for a random number of back off slots. If the channel is busy, following this random back off, the device waits for another random number of back off slots before trying to access the channel again. If the channel is idle, the device begins transmitting on the next available back off slot boundary.
- All transactions must be completed by the time of the next network beacon.

The superframe structure can be extended to additional capabilities. Figure A-2 illustrations “A” and “B” show these functions.

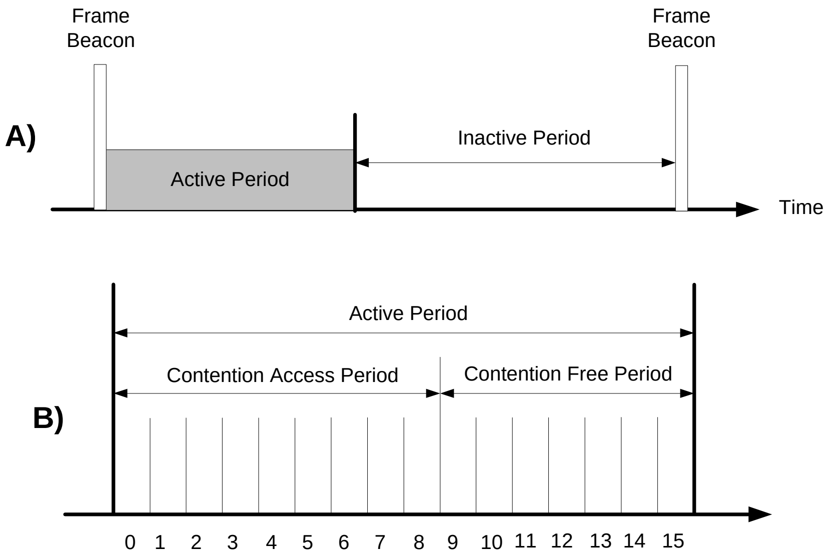


**Figure A-2. Extended Superframe Format**

- Optional Inactive Period (Figure A-2A) - the superframe time between beacons can be divided into an Active Period immediately following the beacon and a sequential Inactive Period. The inactive period allows the coordinator as well as the network devices to enter a low power mode which has the advantage allowing for better coordinator battery life if it cannot be mains powered.
- Adding a Contention Free Period (CFP) within the active period through use of guaranteed time slots (GTSs) (Figure A-2B) - For applications requiring low-latency communication or higher data bandwidth, the coordinator may dedicate a number of active period time slots to GTSs.
    - The active period is always divided into 16 time slots
    - Up to 7 of the last time slots can be dedicated to GTS(s) and these form the CFP.
    - The CFP always follows the CAP.
    - One GTS may occupy more than one time slot
    - All CAP transactions must be completed before the CFP begins
    - One device is assigned per GTS
        - No CCA cycle is used by the GTS assigned device
        - The device must be sure its transaction completes before the end of its GTS period.

## A.4 Summary

The various timing parameters associated with beacon-enabled communication are beyond the scope of this description. The information here is presented as background for understanding the transceiver hardware, specifically the Sequence Manager.