

MC13191

2.4 GHz Low Power Transceiver
for the IEEE® 802.15.4 Standard

Reference Manual

Document Number: MC13191RM
Rev. 1.2
04/2005

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

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

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

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

© Freescale Semiconductor, Inc. 2004, 2005. All rights reserved.

Contents

About This Book

Audience	vii
Conventions	vii
Definitions, Acronyms, and Abbreviations	vii
References	viii

Chapter 1 Introduction

1.1	Overview	1-1
1.2	Features	1-1
1.3	Block Diagrams	1-2
1.4	Data Transfer Mode	1-2
1.5	Packet Structure	1-2
1.6	Receive Path Description	1-2
1.7	Transmit Path Description	1-4

Chapter 2 SPI Register Descriptions

2.1	Overview	2-1
2.2	Mandatory Register Initialization	2-1
2.3	Register Model and Description Details	2-2
2.4	Reset - Register 00	2-4
2.5	RX_Pkt_RAM - Register 01	2-4
2.6	TX_Pkt_RAM - Register 02	2-5
2.7	TX_Pkt_Ctl - Register 03	2-6
2.8	CCA_Thresh - Register 04	2-7
2.9	IRQ_Mask - Register 05	2-8
2.10	Control_A - Register 06	2-10
2.11	Control_B - Register 07	2-11
2.12	Control_C - Register 09	2-12
2.13	CLKO_Ctl - Register 0A	2-13
2.14	GPIO_Dir - Register 0B	2-14
2.15	GPIO_Data_Out - Register 0C	2-16
2.16	LO1_Int_Div - Register 0F	2-18
2.17	LO1_Num - Register 10	2-18
2.18	PA_Lvl - Register 12	2-20
2.19	Tmr_Cmp1_A - Register 1B	2-21
2.20	Tmr_Cmp1_B - Register 1C	2-22
2.21	Tmr_Cmp2_A - Register 1D	2-23
2.22	Tmr_Cmp2_B - Register 1E	2-24
2.23	IRQ_Status - Register 24	2-25
2.24	RST_Ind - Register 25	2-27
2.25	Current_Time_A - Register 26	2-28

2.26	Current_Time_B - Register 27	2-29
2.27	GPIO_Data_In - Register 28	2-30
2.28	Chip_ID - Register 2C	2-31
2.29	RX_Status - Register 2D	2-32
2.30	Timestamp_A - Register 2E	2-33
2.31	Timestamp_B - Register 2F	2-34

Chapter 3

Serial Peripheral Interface (SPI)

3.1	Overview	3-1
3.2	SPI Basic Operation	3-1
3.2.1	SPI Pin Definition	3-1
3.2.1.1	Chip Enable (CE)	3-2
3.2.1.2	SPI Clock (SPICLK)	3-2
3.2.1.3	Master Out / Slave In (MOSI)	3-2
3.2.1.4	Master In / Slave Out (MISO)	3-2
3.2.1.4.1	Setting MISO Output Drive Strength	3-2
3.2.1.4.2	Setting MISO Off Impedance	3-2
3.2.2	SPI Burst Operation	3-3
3.3	SPI Singular Transactions	3-4
3.3.1	SPI Singular Transaction Signalling	3-4
3.3.2	SPI Singular Transaction Protocol	3-5
3.4	Symbol / Data Format	3-6
3.5	SPI Recursive Transactions	3-7
3.5.1	Recursive SPI Register Read	3-7
3.5.2	Recursive SPI Register Write	3-7
3.5.3	Special Case - Packet RAM Access	3-8
3.5.3.1	Recursive Receive Packet RAM Read Access	3-8
3.5.3.1.1	Receive Packet RAM Read Access Flow	3-8
3.5.3.1.2	Receive Packet RAM Read Access Error Conditions	3-9
3.5.3.2	Recursive Transmit Packet RAM Write Access	3-9
3.5.3.2.1	Transmit Packet RAM Write Access Flow	3-10
3.5.3.2.2	Transmit Packet RAM Write Access Error Conditions	3-10
3.6	Program Reset (Writing Address 0x00)	3-11

Chapter 4

Modes of Operation

4.1	Operational Modes Summary	4-1
4.2	Low Power Modes	4-3
4.2.1	Off Mode	4-3
4.2.2	Hibernate Mode	4-3
4.2.3	Doze Mode	4-3
4.2.3.1	Normal Doze Mode	4-3
4.2.3.2	Acoma Doze Mode	4-4

4.3	Active Modes	4-4
4.3.1	Idle Mode	4-4
4.3.2	Controlling Transition to Other Active Modes from Idle.	4-4
4.3.3	Packet Mode Data Transfer TX and RX Operation	4-5
4.3.3.1	Packet Receive Mode	4-5
4.3.3.2	Aborting a Packet Receive Sequence	4-7
4.3.3.3	Packet Transmit Mode	4-7
4.3.4	Clear Channel Assessment (CCA) Modes (including Link Quality Indication)	4-8
4.3.4.1	Clear Channel Assessment Function	4-8
4.3.4.2	Energy Detect Function	4-10
4.3.4.3	Link Quality Indication.	4-11
4.4	Frequency of Operation	4-11
4.4.1	Transmit Power Adjustment	4-11
4.5	2.4GHz PLL Out-of-Lock Interrupt	4-12

Chapter 5 Timer Information

5.1	Event Timer Block	5-1
5.2	Event Timer Time Base	5-1
5.3	Setting Current Time	5-2
5.4	Reading Current Time	5-2
5.5	Latching the Timestamp	5-3
5.6	Event Timer Comparators.	5-3
5.6.1	Timer Compare Fields	5-3
5.7	Timer Disable Bits	5-3
5.7.1	Timer Status Flags	5-4
5.7.2	Timer Interrupt Masks	5-4
5.7.3	Setting Compare Values	5-4
5.8	Intended Event Timer Usage	5-4
5.8.1	Generating Time-Based Interrupts	5-5
5.8.2	Using tmr_cmp2[23:0] to Exit Doze Mode	5-5
5.8.3	Timer-Triggered Transceiver Events	5-6

Chapter 6 Interrupt Description

6.1	Interrupts	6-1
6.1.1	Interrupt Sources	6-1
6.1.2	Output Pin IRQ	6-2
6.1.2.1	Programming IRQ Pullup	6-2
6.1.2.2	Setting IRQ Output Drive Strength	6-2
6.1.3	Interrupts from Exiting Low Power Modes	6-2
6.1.3.1	Exiting Off Mode (Reset)	6-3
6.1.3.2	Exiting Hibernate Mode	6-3
6.1.3.3	Exiting Doze Mode(s).	6-3

Chapter 7

Miscellaneous Functions

7.1	Reset Function	7-1
7.1.1	Input Pin RST	7-1
7.1.2	Software Reset (Writing to Register 00).....	7-1
7.1.3	Reset Indicator Bit (RST_Ind Register 25, Bit 7).....	7-1
7.2	General Purpose Input/Output	7-2
7.2.1	Configuring GPIO Direction	7-2
7.2.2	Setting GPIO Output Drive Strength	7-2
7.2.3	Programming GPIO Output Value	7-2
7.2.4	Reading GPIO Input State	7-2
7.3	Crystal Oscillator	7-2
7.3.1	Crystal Requirements	7-3
7.3.2	Crystal Trim Operation.....	7-3
7.4	Output Clock Pin CLKO.....	7-4
7.4.1	Enable CLKO (clko_en, Control_C Register 09, Bit 5).....	7-4
7.4.2	Setting CLKO frequency (clko_rate[2:0], CLKO_Ctl Register 0A, Bits 2-0).....	7-4
7.4.3	Enable CLKO During Doze Mode (clko_doze_en, Control_B Register 07, Bit 9).....	7-5
7.4.4	Setting CLKO Output Drive Strength (clko_drv[1:0], GPIO_Data_Out Register 0C, Bits 11-10)	7-6
7.5	Input Pin ATTN	7-6

About This Book

This manual describes the Freescale MC13191. It is a 2.4 GHz ISM band transceivers built for the IEEE[®] 802.15.4 Standard. The MC13191 transceiver can function as a standalone transceiver or when combined with a software package and an HCS08 MCU, they form the Freescale IEEE[®] 802.15.4 Standard platform solution.

Audience

This manual is intended for system designers.

Conventions

This document uses the following notational conventions:

- Courier monospaced type indicate commands, command parameters, code examples, expressions, datatypes, and directives.
- Italic type indicates replaceable command parameters.
- All source code examples are in C.

Definitions, Acronyms, and Abbreviations

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

ACK	Acknowledgement Frame
API	Application Programming Interface
BB	Baseband
CCA	Clear Channel Assessment
CRC	Cyclical Redundancy Check
DCD	Differential Chip Decoding
DME	Device Management Entity
FCS	Frame Check Sequence
FFD	Full Function Device
FFD-C	Full Function Device Coordinator
FLI	Frame Length Indicator
GTS	Guaranteed Time Slot
HW	Hardware
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LO	Local Oscillator
MAC	Medium Access Control
MCPS	MAC Common Part Sublayer

MCU	Microcontroller Unit
MLME	MAC Sublayer Management Entity
MSDU	MAC Service Data Unit
NWK	Network
PA	Power Amplifier
PAN	Personal Area Network
PANID	PAN Identification
PHY	PHYsical Layer
PIB	PAN Information Base
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
RF	Radio Frequency
RFD	Reduced Function Device
SAP	Service Access Point
SFD	Start of Frame Delimiter
SPI	Serial Peripheral Interface
SSCS	Service Specific Convergence Layer
SW	Software
VCO	Voltage Controlled Oscillator

References

The following sources were referenced to produce this book:

- [1] IEEE[®] 802.15.4 Standard
- [2] Freescale MC13191 Data Sheet

Chapter 1

Introduction

1.1 Overview

The MC13191 is a short range, low power, 2.4 GHz Industrial, Scientific, and Medical (ISM) band transceivers. The MC13191 contains a complete packet data modem which is compliant with the IEEE® 802.15.4 Standard PHY (Physical) layer. This allows the development of proprietary point-to-point and star networks based on the 802.15.4 packet structure and modulation format. For full 802.15.4 compliance, the MC13191 and Freescale's 802.15.4 MAC software are required.

When combined with an appropriate microcontroller (MCU), the MC13191 provides a cost-effective solution for short-range data links and networks. Interface with the MCU is accomplished using a four wire serial peripheral interface (SPI) connection and an interrupt request output which allows for the use of a variety of processors. The software and processor can be scaled to fit applications ranging from simple point-to-point to star networks.

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

- Remote control and wire replacement in industrial systems such as wireless sensor networks
- Factory automation and motor control
- Energy Management (lighting, HVAC, etc.)
- Asset tracking and monitoring

Potential consumer applications include:

- Home automation and control (lighting, thermostats, etc.)
- Human interface devices (keyboard, mice, etc.)
- Remote entertainment control
- Wireless toys

The transceiver includes a low noise amplifier, 1.0 mW power amplifier (PA), voltage controlled oscillator (VCO), on-board power supply regulation, and full spread-spectrum encoding and decoding. The device supports 250 kbps Offset-Quadrature Phase Shift Keying (O-QPSK) data in 2.0 MHz channels with 5.0 MHz channel spacing. The SPI port and interrupt request output are used for receive (RX) and transmit (TX) data transfer and control.

1.2 Features

- IEEE® 802.15.4 PHY Compliant
 - 16 Channels
 - Supports 250 kbps O-QPSK data in 5.0 MHz channels and full spread-spectrum encode/decode
 - RX sensitivity of -91 dBm (typical) at 1.0% packet error rate
- Recommended power supply range: 2.0 to 3.4 V
- 0 dBm nominal, programmable up to 4 dBm typical maximum output power

- Buffered transmit and receive data packets for simplified use with low cost MCUs
- Three power down modes for power conservation:
 - < 2.5 μ A Off current
 - 2.3 μ A Typical Hibernate current
 - 35 μ A Typical Doze current (no CLKO)
- Two internal timer comparators available to reduce MCU resource requirements
- Programmable frequency clock output for use by MCU
- Onboard trim capability for 16 MHz crystal reference oscillator eliminates need for external variable capacitors and allows for automated production frequency calibration.
- Seven general purpose input/output (GPIO) signals
- Operating temperature range: -40 °C to 85 °C
- Small form factor QFN-32 Package
 - Meets moisture sensitivity level (MSL) 3
 - 260 °C peak reflow temperature
 - Meets lead-free requirements

1.3 Block Diagrams

Figure 1-2 shows a simplified block diagram of the MC13191 transceiver that meets the requirements of the IEEE[®] 802.15.4 PHY. Figure 1-3 shows the basic system block diagram for the MC13191 in an application. Interface with the transceiver is accomplished through a 4-wire SPI port and interrupt request line. The media access control (MAC), drivers, and network and application software (as required) reside on the host processor. The host can vary from a simple 8-bit device up to a sophisticated 32-bit processor depending on application requirements.

1.4 Data Transfer Mode

The MC13191 has a data transfer mode called Packet Mode where data is buffered in on-chip Packet RAMs. There is a TX Packet RAM and an RX Packet RAM, each of which are 64 locations by 16 bits wide.

1.5 Packet Structure

Figure 1-4 shows the packet structure of the MC13191 which is consistent with the IEEE[®] 802.15.4 Standard. Payloads of up to 125 bytes are supported. The MC13191 adds a four-byte preamble, a one-byte Start of Frame Delimiter (SFD), and a one-byte Frame Length Indicator (FLI) before the data. A Frame Check Sequence (FCS) is calculated and appended to the end of the data.

1.6 Receive Path Description

In the receive signal path, the RF input is converted to low IF In-phase and Quadrature (I & Q) signals through two down-conversion stages. An Energy Detect can be performed based upon the baseband energy integrated over a specific time interval. The digital back end performs Differential Chip Detection (DCD),

the correlator “de-spreads” the Direct Sequence Spread Spectrum (DSSS) Offset QPSK (O-QPSK) signal, determines the symbols and packets, and detects the data.

The preamble, SFD, and FLI are parsed and used to detect the payload data and FCS which are stored in RAM. A two-byte FCS is calculated on the received data and compared to the FCS value appended to the transmitted data which generates a Cyclical Redundancy Check (CRC) result. Link Quality is measured over a 64 μ s period after the packet preamble and stored in RAM.

The MC13191 uses a packet mode where the data is processed as an entire packet and stored in Rx Packet RAM. The MCU is notified that an entire packet has been received via an interrupt.

Figure 1-1 shows energy detection reported power versus input power.

NOTE

The IEEE[®] 802.15.4 Standard accuracy and range limits are shown for reference.

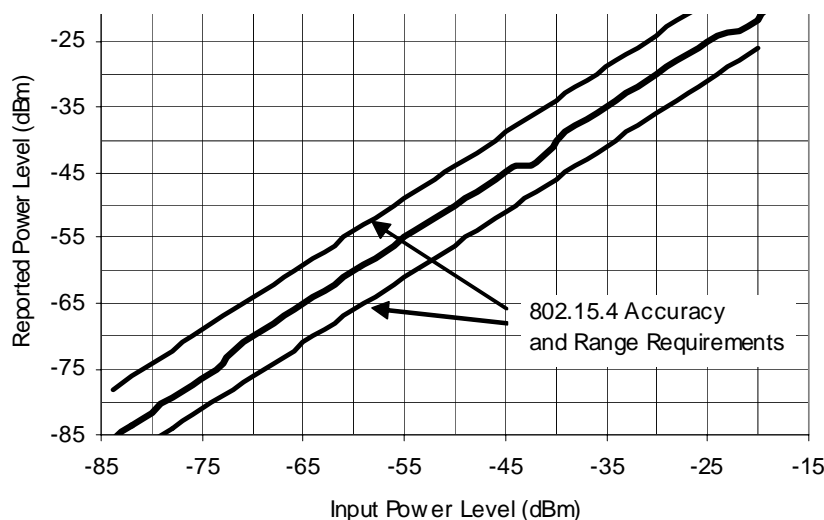


Figure 1-1. Reported Power Level Versus Input Power for Energy Detect or Link Quality Indicator

1.7 Transmit Path Description

For the transmit path, the TX data that was previously stored in TX Packet RAM is retrieved, formed into packets, spread, and then up-converted to the transmit frequency.

Because the MC13191 is used in packet mode, data is processed as an entire packet. The data is first loaded into the TX buffer. The MCU then requests that the MC13191 transmit the data. The MCU is notified via an interrupt when the whole packet has successfully been transmitted.

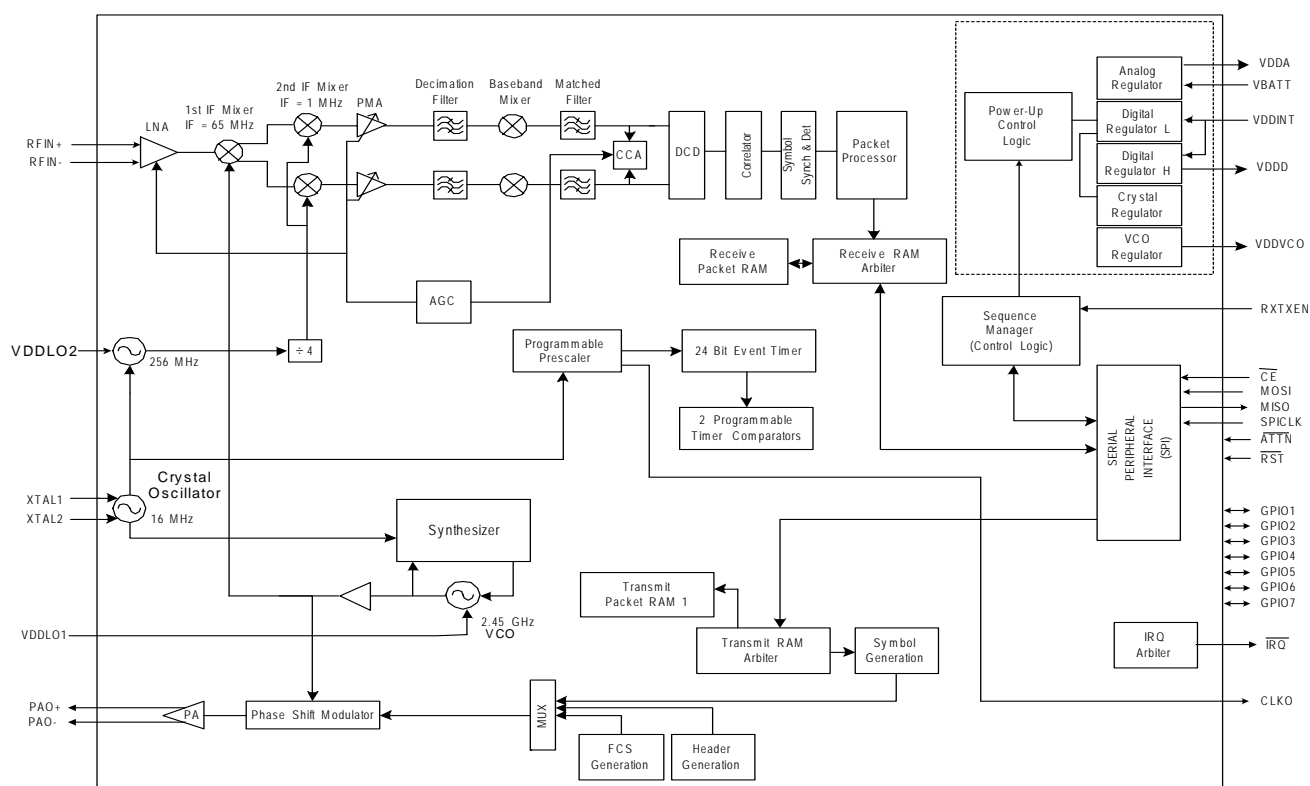


Figure 1-2. MC13191 Simplified Block Diagram

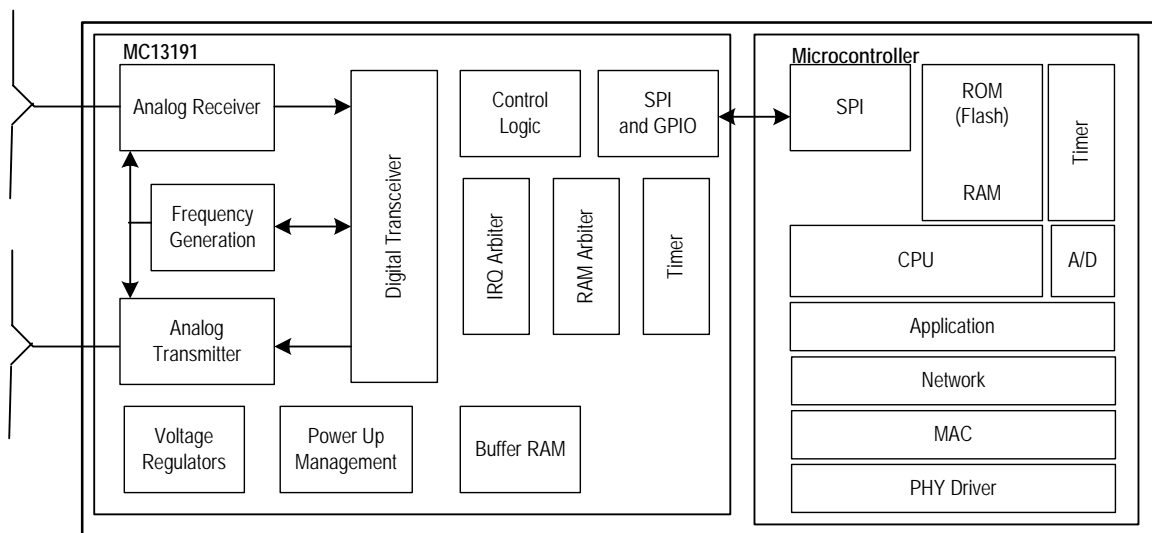


Figure 1-3. System Level Block Diagram

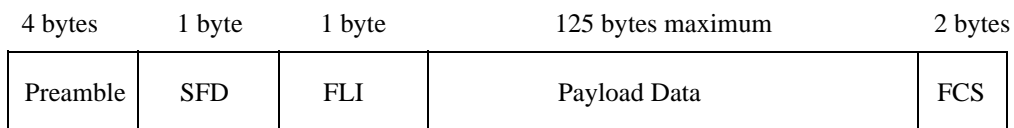


Figure 1-4. MC13191 Packet Structure

Chapter 2

SPI Register Descriptions

2.1 Overview

All control, reading of status, writing of data, and reading of data is done through the MC13191 SPI port. The host microcontroller accesses the transceiver through SPI “transactions” in which multiple bursts of byte-long data are transmitted on the SPI bus. Each transaction is three or more bursts long depending on the transaction type, and these are described in detail in [Section 3.2.2](#).

Transactions are always read accesses or write accesses to register addresses. The associated data for any single register access is always 16 bits in length. This chapter describes all the registers that users should access in the MC13191. Undocumented addresses should not be accessed.

NOTE

Register default values for reserved fields should not be modified unless specifically noted. Freescale recommends that all writes to control register fields that only modify part of the 16-bit word be done as a read-modify-write operation.

2.2 Mandatory Register Initialization

NOTE

Certain hidden registers **MUST** be initialized to given conditions for proper operation of the transceiver. These programmed conditions are given in [Table 2-1](#) (these are not default values).

Table 2-1. Mandatory Register Initialization

ADD (Hex)	BIT	CONDITION
08	1	Set to 1
08	4	Set to 1
11	9:8	Set to 00
06	14	Set to 1

2.3 Register Model and Description Details

Table 2-2 summarizes the MC13191 Register Model and the following sections describe each register in more detail.

Table 2-2. MC13191 SPI Register Table

REGISTER NAME	Add (Hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	00	software_reset															
RX_Pkt_RAM	01	rx_pkt_ram[15:0]															
TX_Pkt_RAM	02	tx_pkt_ram[15:0]															
TX_Pkt_Ctl	03										tx_pkt_length[6:0]						
CCA_Thresh	04	cca_vt[7:0]								power_comp[7:0]							
IRQ_Mask	05	attn_mask			ram_addr_mask	arb_busy_mask		pll_lock_mask	acomma_en				doze_mask			tmr2_mask	tmr1_mask
							cca_mask	tx_sent_mask	rx_rcvd_mask	tmr_trig_en			cca_type[1:0]			xcvr_seq	
Control_A	06																
Control_B	07	tmr_load				miso_hiz_en		clko_doze_en								hib_en	doze_en
Control_C	09																
CLKO_Ctl	0A	xtal_trim[7:0]														clko_rate[2:0]	
GPIO_Dir	0B	gpio1234_drv[1:0]		gpio7_oen	gpio6_oen	gpio5_oen	gpio4_oen	gpio3_oen	gpio2_oen	gpio1_oen	gpio7_ien	gpio6_ien	gpio5_ien	gpio4_ien	gpio3_ien	gpio2_ien	gpio1_ien
GPIO_Data_Out	0C	gpio567_drv[1:0]		miso_drv[1:0]		clko_drv[1:0]		irqb_drv[1:0]		irqb_pup_en	gpio7_o	gpio6_o	gpio5_o	gpio4_o	gpio3_o	gpio2_o	gpio1_o

Table 2-2. MC13191 SPI Register Table (continued)

REGISTER NAME	Add (Hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
LO1_Int_Div	0F									lo1_idiv[7:0]								
LO1_Num	10	lo1_num[15:0]																
PA_Lvl	12									pa_lvl_coarse[1:0]		pa_lvl_fine[1:0]		pa_drv_coarse[1:0]		pa_drv_fine[1:0]		
		tmr_cmp1_dis								tmr_cmp1[23:16]								
Tmr_Cmp1_A	1B																	
Tmr_Cmp1_B	1C	tmr_cmp1[15:0]																
Tmr_Cmp2_A	1D	tmr_cmp2_dis								tmr_cmp2[23:16]								
Tmr_Cmp2_B	1E	tmr_cmp2[15:0]																
IRQ_Status	24	pll_lock_irq	ram_addr_err	arb_busy_err			attn_irq	doze_irq	tmr1_irq	rx_rcvd_irq	tx_sent_irq	cca_irq			tmr2_irq	cca	crc_valid	
RST_Ind	25									reset_ind								
Current_Time_A	26									et[23:16]								
Current_Time_B	27	et[15:0]																
GPIO_Data_In	28		gpio7_i	gpio6_i	gpio5_i	gpio4_i	gpio3_i	gpio2_i	gpio1_i									
Chip_Id	2C	chip_id[8:0]																
RX_Status	2D	cca_final[7:0]									rx_pkt_latch[6:0]							
Timestamp_A	2E									timestamp[23:16]								
Timestamp_B	2F	timestamp[15:0]																

2.4 Reset - Register 00

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

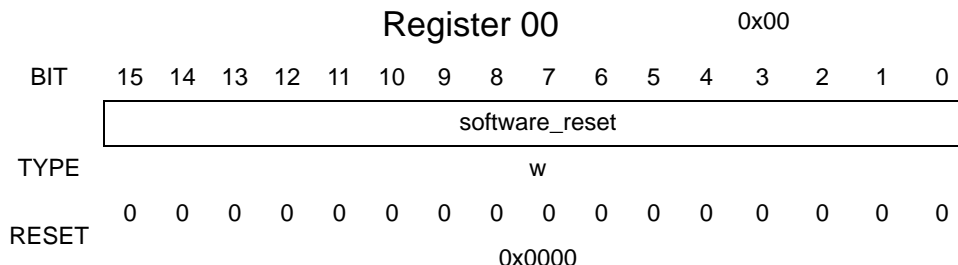


Table 2-3. Register 00 Description

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

2.5 RX_Pkt_RAM - Register 01

The receive Packet RAM register is accessed for RX data transfer. Once a packet has been received, the payload data is stored in the RX Packet RAM and the length of the packet data is contained in Register 2D, Bit 6-0. A recursive read (see [Section 3.5.1](#)) to the RX_Pkt_RAM Register 01 is required to access the RX packet payload data.

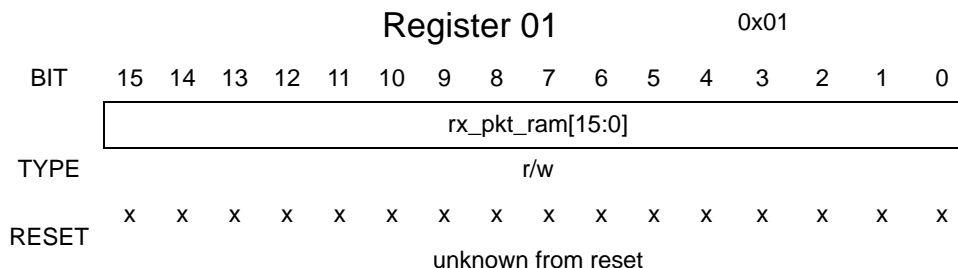


Table 2-4. Register 01 Description

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

2.6 TX_Pkt_RAM - Register 02

The transmit Packet RAM register is accessed for TX data transfer. The packet payload data must be written to the TX Packet RAM and the length of the packet data must be written to TX_Pkt_Ctl Register 03, Bits 6-0. A recursive write (see [Section 3.5.2](#)) to the TX_Pkt_RAM Register 02 is required to load the TX packet payload data.

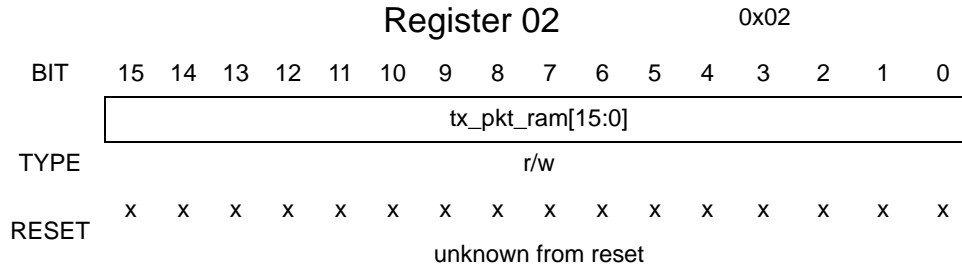


Table 2-5. Register 02 Description

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

2.7 TX_Pkt_Ctl - Register 03

The TX_Pkt_RAM_Ctl Register 03 contains a field tx_pkt_length, Bits 6-0, that defines the length of the payload data for a transmission data packet.

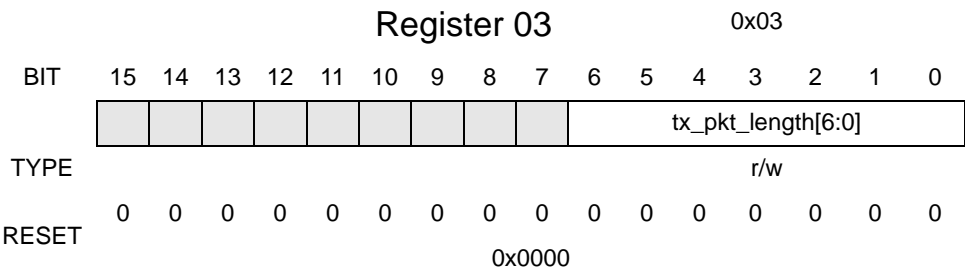


Table 2-6. Register 03 Description

Name	Description	Operation
Bits 15-7	Reserved	Leave default
Bits 6 - 0	tx_pkt_length[6:0] — The Transmit Packet Length bits represent the number of bytes to be transmitted from transmit Packet RAM plus 2 bytes for FCS.	Total transmit payload data length in bytes

2.8 CCA_Thresh - Register 04

The CCA_Thresh Register 04 contains the `cca_vt[7:0]` 8-bit CCA threshold value, Bits 15 - 8. To calculate desired the `cca_vt[7:0]` value:

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

A second field is `power_comp[7:0]`, Bits 7 - 0, which is an offset that is added to the measured value of the average energy from a CCA/ED function or LQI value from an RX function, and the resulting value is stored in `cca_final[7:0]`, RX_Status Register 2D. By using this `power_comp[7:0]` value, users can compensate the `cca_final[7:0]` value for external gain in the RX path. See [Section 4.3.4.1](#) for more detailed information.

Register 04																0x04
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	cca_vt[7:0]								power_comp[7:0]							
TYPE	r/w															
RESET	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1
																0x008D

Table 2-7. Register 04 Description

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

2.9 IRQ_Mask - Register 05

The IRQ_Mask Register 05 provides most, but not all, mask bits for various interrupt sources for the MC13191. If a mask bit is set, its associated status bit being true will generate an interrupt on the MC13191 IRQ pin. The interrupt is cleared when the status bit is read via a SPI transaction.

Register 05																0x05
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	attn_mask			ram_addr_mask	arb_busy_mask		pll_lock_mask	acom_en				doze_mask			tmr2_mask	tmr1_mask
TYPE	r/w			r/w	r/w		r/w	r/w				r/w			r/w	r/w
RESET	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0x8020																

Table 2-8. Register 05 Description

Name	Description	Operation
Bits 14-13, 10, 7-5, 3-2	Reserved	Leave default
Bit 15	attn_mask — The attention interrupt mask bit controls the attn_irq interrupt on the IRQ pin. Default is for the interrupt to be enabled out of reset.	1 = Allows attn_irq to generate an interrupt on the IRQ pin. 0 = When attn_irq status bit is set, IRQ pin is not asserted.
Bit 12	ram_addr_mask — The Packet RAM address error interrupt mask bit controls the ram_addr_err interrupt on the IRQ pin.	1 = Allows ram_addr_err to generate an interrupt on the IRQ. 0 = When ram_addr_err status bit is set, IRQ pin is not asserted.
Bit 11	arb_busy_mask — The Packet RAM arbiter busy error interrupt mask bit controls the arb_busy_err interrupt on the IRQ pin.	1 = Allows arb_busy_err to generate an interrupt on the IRQ pin. 0 = When arb_busy_err status bit is set, IRQ pin is not asserted.
Bit 9	pll_lock_mask — The LO1 unlock detect mask bit controls the pll_lock_irq interrupt on the IRQ pin.	1 = Allows pll_lock_irq to generate an interrupt on the IRQ pin. 0 = When pll_lock_irq status bit is set, IRQ pin is not asserted.
Bit 8	acom_en — The Acoma mode enable bit controls Doze mode. Acoma is an enhanced power save mode within Doze.	1 = The MC13191 stays in Doze until ATTN asserted. Event Timer and Prescaler clocks disabled for additional current savings. 0 = Normal operation. Doze is exited by TC2 match or ATTN assertion.
Bit 4	doze_mask — The Doze timer interrupt mask bit controls the doze_irq interrupt on the IRQ pin.	1 = Allows doze_irq to generate an interrupt on the IRQ pin. 0 = When doze_irq status bit is set, IRQ pin is not asserted.

Table 2-8. Register 05 Description (continued)

Name	Description	Operation
Bit 1	tmr2_mask — The Event Timer two interrupt mask bit controls the tmr2_irq interrupt on the IRQ pin.	1 = Allows tmr2_irq to generate an interrupt on the IRQ pin. 0 = When tmr2_irq status bit is set, IRQ pin is not asserted.
Bit 0	tmr1_mask — The Event Timer one interrupt mask bit controls the tmr1_irq interrupt on the IRQ pin.	1 = Allows tmr1_irq to generate an interrupt on the IRQ pin. 0 = When tmr1_irq status bit is set, IRQ pin is not asserted.

2.10 Control_A - Register 06

The Control_A Register 06 is one of several registers that provide control fields for the MC13191.

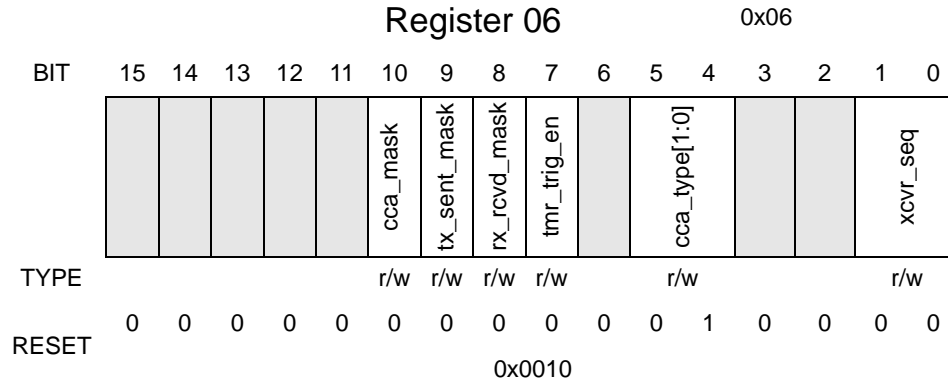


Table 2-9. Register 06 Description

Name	Description	Operation
Bits 15-11, 6,3,2	Reserved	Leave default
Bit 10	cca_mask — The CCA interrupt mask bit controls the cca_irq interrupt on the IRQ pin.	1 = Allows the cca_irq to generate an interrupt on the IRQ pin. 0 = When icca_irq status bit is set, IRQ pin is not asserted.
Bit 9	tx_sent_mask — The transmit sent mask bit controls the tx_sent_irq interrupt on the IRQ pin.	1 = Allows the tx_sent_irq to generate an interrupt on the IRQ pin. 0 = When tx_sent_irq status bit is set, IRQ pin is not asserted.
Bit 8	rx_rcvd_mask — The packet received interrupt mask bit controls the rx_rcvd_irq interrupt on the IRQ pin.	1 = Allows rx_rcvd_irq to generate an interrupt on the IRQ pin. 0 = When rx_rcvd_irq status bit is set, IRQ pin is not asserted.
Bit 7	tmr_trig_en — The timer trigger enable bit determines whether transceiver operation is initiated via Timer Comparator 2 or manually.	1 = The selected transceiver operation is initiated via Timer Comparator 2, or TC2_Prime. 0 = The selected transceiver operation is initiated manually via the RXTXEN pin and by SPI programming.
Bits 5-4	cca_type[1:0] — The clear channel assessment type bits selects one of two possible CCA functions. Algorithm results are reported in field cca_final[7:0], RX_Status Register 2D, Bits 15 - 8.	01 = clear channel assessment 10 = energy detection
Bits 1-0	xcvr_seq[1:0] — The transceiver operation bits select one of four possible transceiver modes.	00 = Idle (default) 01 = CCA /energy detection 10 = Packet Mode RX 11 = Packet Mode TX

2.11 Control_B - Register 07

The Control_B Register 07 is one of several registers that provide control fields for the MC13191.

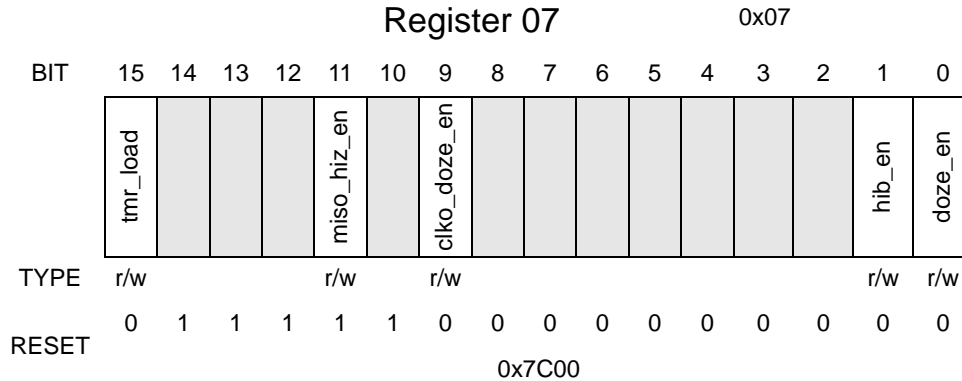
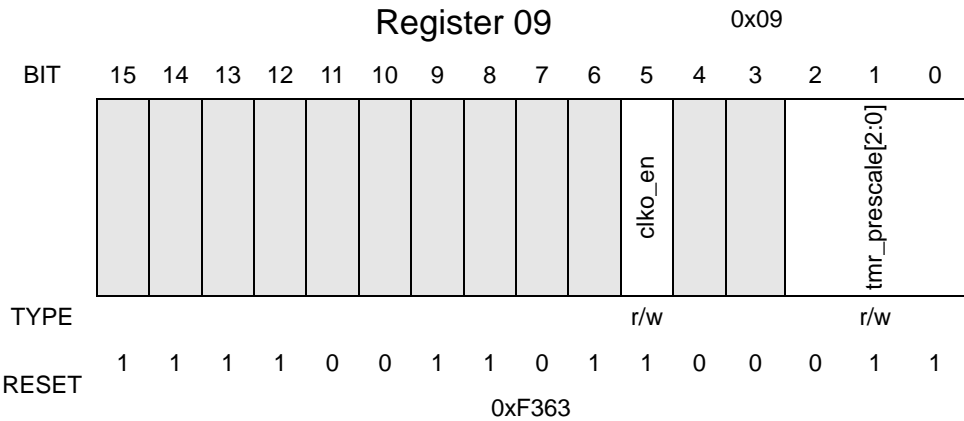


Table 2-10. Register 07 Description

Name	Description	Operation
Bits 14-12, 10, 8-2	Reserved	Leave default
Bit 15	tmr_load — The load Event Timer bit, when programmed from low to high, causes the value of the SPI field tmr_cmp1[23:0] to be loaded into the Event Timer.	Write from 0 to 1 to affect load. Rewrite to 0, before writing another 1 to affect another load.
Bit 11	miso_hiz_en — The MISO high impedance enable bit either tristates or drives the MISO pin to a logic low when \overline{CE} is negated.	1 = MISO pin is tristated when \overline{CE} is negated. 0 = MISO pin is driven to a logic low when \overline{CE} is negated.
Bit 9	clko_doze_en — The CLKO enable in Doze Mode bit controls toggling of the CLKO pin during Doze Mode.	1 = The CLKO pin continues to toggle at selected rate during Doze Mode if CLKO is enabled (clko_en = 1). 0 = The CLKO pin stops toggling 128 xtal or reference cycles after the doze_en bit is programmed to 1. Note: In Doze Mode only, CLKO frequencies of 1.0 MHz or less are available.
Bit 1	hib_en — The hibernate enable bit can set the MC13191 into its lowest power saving mode without a running time base.	1 = Places the MC13191 into its lowest power operating mode without a running time base. 0 = Normal operation.
Bit 0	doze_en — The doze enable bit can set the MC13191 into its lowest power saving mode with a running time base.	1 = Places the MC13191 into its lowest power operating mode with a running time base. 0 = Normal operation.

2.12 Control_C - Register 09

The Control_C Register 09 is one of several registers that provide control fields for the MC13191.



2.13 CLKO_Ctl - Register 0A

The MC13191 provides an output clock with a programmable frequency that can be used to drive another device, such as a microcontroller. The field `xtal_trim[7:0]`, CLKO_Ctl Register 0A, Bits 15-8, alter the capacitive loading to the crystal and affects the oscillator frequency. See [Section 7.3.2](#).

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

Table 2-12. CLKO Frequency

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



Table 2-13. Register 0A Description

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

2.14 GPIO_Dir - Register 0B

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

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

Table 2-14. Register 0B Description

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

Table 2-14. Register 0B Description (continued)

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

2.15 GPIO_Data_Out - Register 0C

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

Register 0C																0x0C
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	gpio567_drv[1:0]		miso_drv[1:0]		clko_drv[1:0]		irqb_drv[1:0]		irqb_pup_en	gpio7_o	gpio6_o	gpio5_o	gpio4_o	gpio3_o	gpio2_o	gpio1_o
TYPE	r/w		r/w		r/w		r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x0080																

Table 2-15. Register 0C Description

Name	Description	Operation
Bits 15-14	gpio567_drv[1:0] — These bits select output drive strength for GPIO5 through GPIO7.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bits 13-12	miso_drv[1:0] - These bits select output drive strength for signal MISO.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bits 11-10	clko_drv[1:0] - These bits select output drive strength for signal CLKO.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bits 9-8	irqb_drv[1:0] - These bits select output drive strength for signal $\overline{\text{IRQ}}$.	00 (default) = lowest drive strength; 11 = highest drive strength.
Bit 7	irqb_pup_en — $\overline{\text{IRQ}}$ pullup enable.	1 = Onboard pullup enabled (nominal 40 k Ω) on $\overline{\text{IRQ}}$ pin (default) 0 = Open drain only (external pullup required).
Bit 6	gpio7_o — GPIO7 output value.	1 = GPIO7 driven high 0 = GPIO7 driven low
Bit 5	gpio6_o — GPIO6 output value.	1 = GPIO6 driven high 0 = GPIO6 driven low
Bit 4	gpio5_o — GPIO5 output value.	1 = GPIO5 driven high 0 = GPIO5 driven low
Bit 3	gpio4_o — GPIO4 output value.	1 = GPIO4 driven high 0 = GPIO4 driven low
Bit 2	gpio3_o — GPIO3 output value.	1 = GPIO3 driven high 0 = GPIO3 driven low

Table 2-15. Register 0C Description (continued)

Name	Description	Operation
Bit 1	gpio2_o — GPIO2 output value.	1 = GPIO2 driven high 0 = GPIO2 driven low
Bit 0	gpio1_o — GPIO1 output value.	1 = GPIO1 driven high 0 = GPIO1 driven low

2.16 LO1_Int_Div - Register 0F

The LO1_Int_Div Register 0F contains the 8-bit integer divide value for the LO1 fractional-N synthesizer that sets transceiver channel frequency. See [Table](#) .

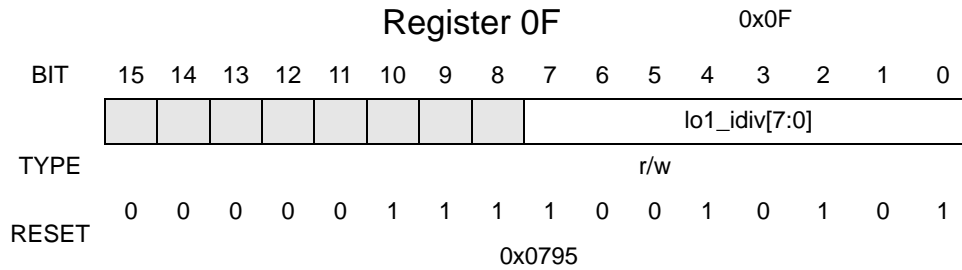


Table 2-16. Register 0F Description

Name	Description	Operation
Bits 15-8	Reserved.	Leave default.
Bits 7-0	lo1_idiv[7:0] — The LO1 integer divide bits represent the integer divide value for the LO1 fractional-N synthesizer.	Default is 149dec (0x95). See Table 2-18 .

2.17 LO1_Num - Register 10

The LO1_Num Register 10 contains the 16-bit integer numerator value for the LO1 fractional-N synthesizer that sets transceiver channel frequency. See [Table 2-18](#).

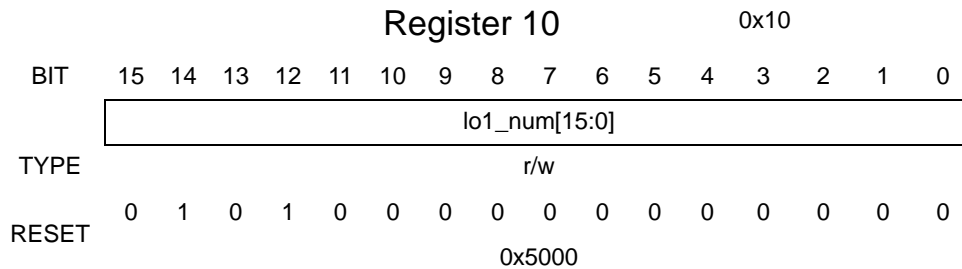


Table 2-17. Register 10 Description

Name	Description	Operation
Bits 15-0	lo1_num[15:0] — These bits represent the numerator of the fractional divide value for the LO1 fractional-N synthesizer.	Default is 20,480dec (0x5000). See Table 2-18 .

Table 2-18. Channel Operation

Channel (Dec)	IEEE 802.15.4 Channel Number	Frequency (MHz)	Integer Setting lo1_idiv[7:0] (Dec / Hex)	Fractional Setting lo1_num[15:0] (Dec / Hex)
1	11	2405	149 / 0x95 (default)	20480 / 0x5000 (default)
2	12	2410	149 / 0x95	40960 / 0xA000
3	13	2415	149 / 0x95	61440 / 0xF000
4	14	2420	150 / 0x96	16384 / 0x4000
5	15	2425	150 / 0x96	36864 / 0x9000
6	16	2430	150 / 0x96	57344 / 0xE000
7	17	2435	151 / 0x97	12288 / 0x3000
8	18	2440	151 / 0x97	32768 / 0x8000
9	19	2445	151 / 0x97	53248 / 0xD000
10	20	2450	152 / 0x98	8192 / 0x2000
11	21	2455	152 / 0x98	28672 / 0x7000
12	22	2460	152 / 0x98	49152 / 0xC000
13	23	2465	153 / 0x99	4096 / 0x1000
14	24	2470	153 / 0x99	24576 / 0x6000
15	25	2475	153 / 0x99	45056 / 0xB000
16	26	2480	154 / 0x9A	0 / 0x0000

Register 12																0x12
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								pa_lvl_coarse[1:0]		pa_lvl_fine[1:0]		pa_drv_coarse[1:0]		pa_drv_fine[1:0]		
TYPE								r/w		r/w						
RESET	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0
																0x00BC

Name	Description	Operation
Bits 15-8	Reserved	Leave default
Bits 7-6	pa_lvl_coarse[1:0] - These bits select the coarse PA power level adjustment.	See Section 4.4.1 for more information.
Bits 5-4	pa_lvl_fine[1:0] - These bits selects fine PA power level adjustment.	See Section 4.4.1 for more information.
Bits 3-2	pa_drv_coarse[1:0] - These bits select the coarse PA drive level adjustment.	Leave default
Bits 1-0	pa_drv_fine[1:0] - These bits select the fine PA drive level adjustment	Leave default

2.19 Tmr_Cmp1_A - Register 1B

The Tmr_Cmp1_A Register 1B contains the disable bit for Timer Comparator 1 and stores the most significant 8 bits of the 24-bit compare value. It is suggested that the timer be disabled during system initialization as the default mode out of reset is timer enabled.

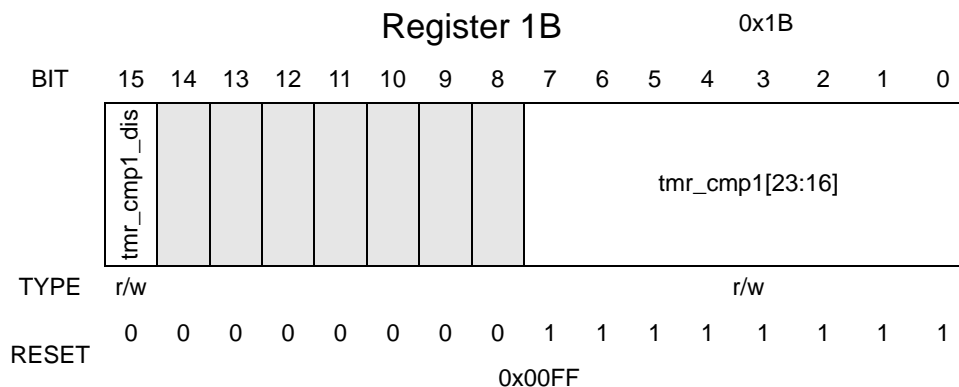


Table 2-20. Register 1B Description

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

2.20 Tmr_Cmp1_B - Register 1C

The Tmr_CMP1_B Register 1C stores the least significant 16 bits of the 24-bit compare value.

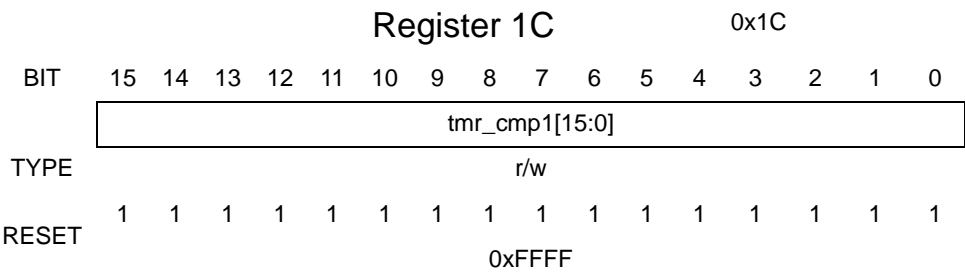


Table 2-21. Register 1C Description

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

2.21 Tmr_Cmp2_A - Register 1D

The Tmr_Cmp2_A Register 1B contains the disable bit for Timer Comparator 2 and stores the most significant 8 bits of the 24-bit compare value. It is suggested that the timer be disabled during system initialization as the default mode out of reset is timer enabled.

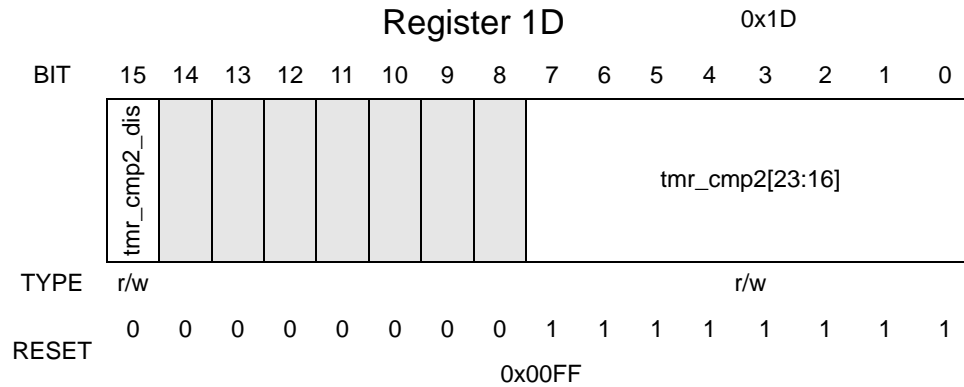


Table 2-22. Register 1D Description

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

2.22 Tmr_Cmp2_B - Register 1E

The Tmr_CMP2_B Register 1C stores the least significant 16 bits of the 24-bit compare value.

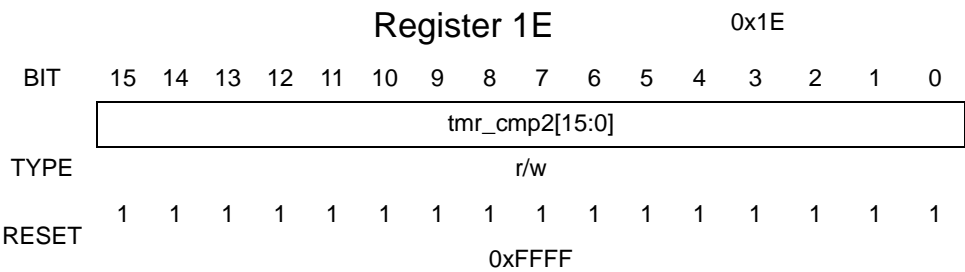


Table 2-23. Register 1E Description

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

2.23 IRQ_Status - Register 24

The IRQ_Status Register 24 contains status bits that can, in turn, cause an interrupt request when enabled. Reading the register clears the status bits and releases an associated interrupt request on $\overline{\text{IRQ}}$.

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

Table 2-24. Register 24 Description

Name	Description	Operation
Bits 12-11,4-3	Reserved	Leave default
Bit 15	pll_lock_irq — The Local Oscillator 1 Lock Detect Interrupt bit indicates whether the LO1 PLL is in or out of lock.	1 = LO1 PLL out of lock. Will cause an interrupt if pll_lock_mask is enabled. 0 = LO1 PLL is locked.
Bit 14	ram_addr_err — The Packet RAM Address Error Interrupt.	A recursive SPI read or write operation to Packet RAM exceeded maximum RAM address.
Bit 13	arb_busy_err — The Packet RAM Arbiter Busy Error Interrupt.	A SPI read or write operation to Packet RAM attempted during packet reception or transmission, respectively.
Bit 10	attn_irq — The Attention Interrupt bit indicates the $\overline{\text{ATTN}}$ pin has been asserted or Power-up complete condition from a reset condition.	The bit being set indicates the $\overline{\text{ATTN}}$ signal has been asserted low or that the MC13191 has reached a Power-up complete condition after software reset (CE released) or a hardware reset ($\overline{\text{RST}}$ released).
Bit 9	doze_irq — The Doze Timer Interrupt bit.	This bit gets set when the 'tmr_cmp2[23:0]' field matches the current Event Timer value while in Doze Mode. The MC13191 returns to Idle state from Doze mode.
Bit 8	tmr1_irq — The Timer Compare 1 Interrupt bit.	This bit gets set when the 'tmr_cmp1[23:0]' field matches the current Event Timer value.
Bit 7	rx_rcvd_irq — The RX Packet Received Interrupt.	RX Packet Mode reception complete and transceiver has returned to Idle Mode.

Table 2-24. Register 24 Description (continued)

Name	Description	Operation
Bit 6	tx_sent_irq — The TX Packet Sent Interrupt.	TX Packet Mode operation complete and transceiver has returned to Idle Mode.
Bit 5	cca_irq — The Clear Channel Assessment Ready Interrupt bit.	This bit is set when the 'Clear Channel Assessment' operation is complete.
Bit 2	tmr2_irq — The Timer Compare 2 Interrupt bit.	This bit gets set when the 'tmr_cmp2[23:0]' field matches the current Event Timer value, or alternately if 'tc2_prime[15:0]' field matches the current Event Timer[15:0] when enabled.
Bit 1	cca — The Clear Channel Assessment bit indicates channel busy or channel idle.	1 = channel busy detected 0 = channel idle detected Note: For cca_type[1:0], Register 6, Bits 5:4 = 10, Energy Detect mode, CCA is not calculated and cca is held low.
Bit 0	crc_valid — The RX CRC Result bit denotes if the CRC is correct or not.	1 = RX CRC correct 0 = RX CRC incorrect (default)

2.24 RST_Ind - Register 25

The RST_Ind Register 25 contains the reset indicator bit. This bit is useful for the MCU to determine if the transceiver has returned from a Hibernate condition via an $\overline{\text{ATTN}}$ signal or recovered from a reset condition.

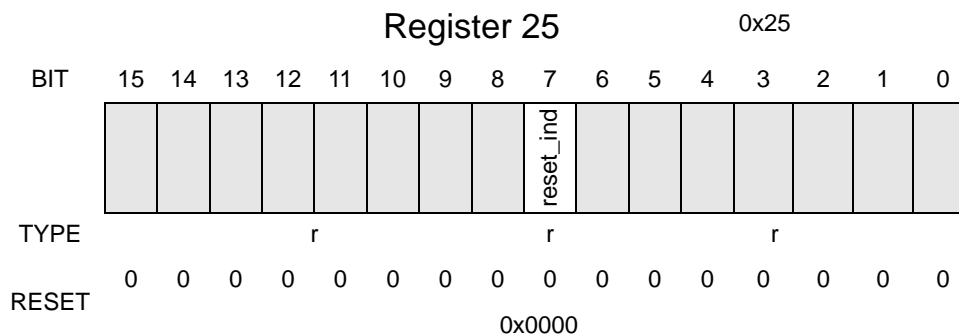


Table 2-25. Register 25 Description

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

2.25 Current_Time_A - Register 26

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

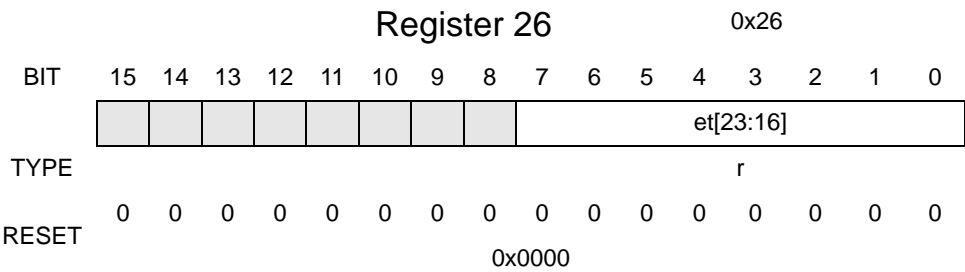


Table 2-26. Register 26 Description

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

2.26 Current_Time_B - Register 27

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

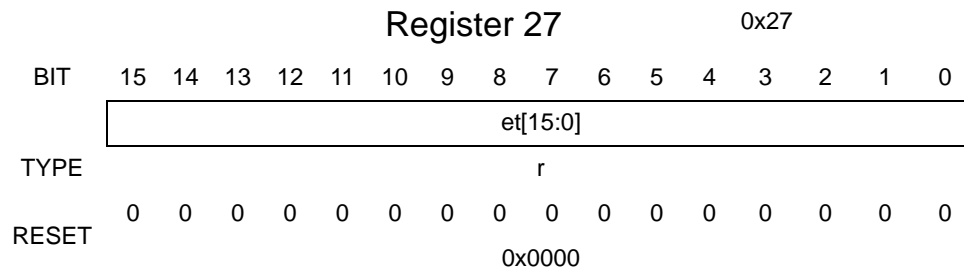


Table 2-27. Register 27 Description

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

2.27 GPIO_Data_In - Register 28

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

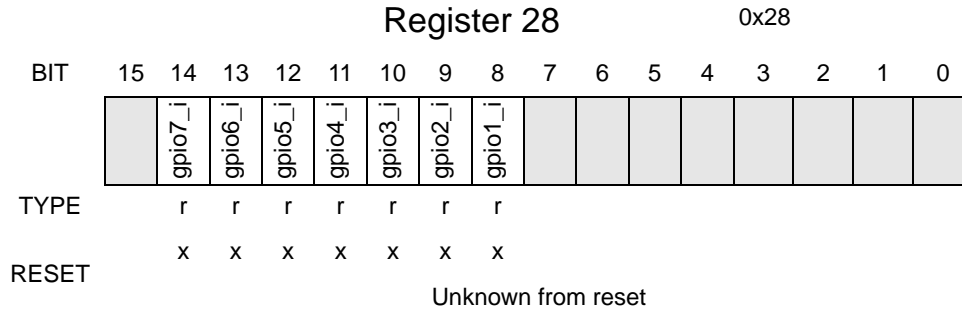


Table 2-28. Register 28 Description

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

2.28 Chip_ID - Register 2C

The Chip_ID Register 2C is read to get the 9-bit chip version code contained in the chip_id[8:0] field.

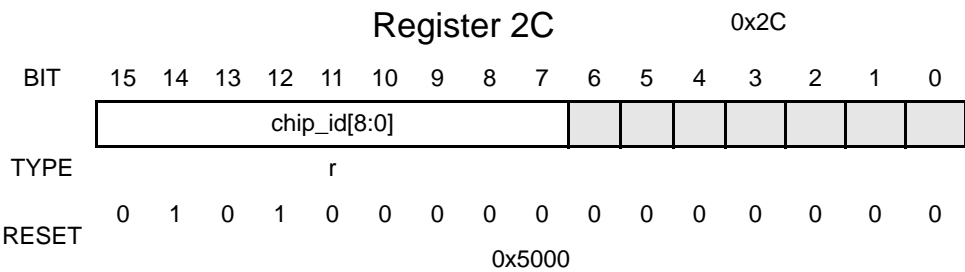


Table 2-29. Register 2C Description

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

2.29 RX_Status - Register 2D

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

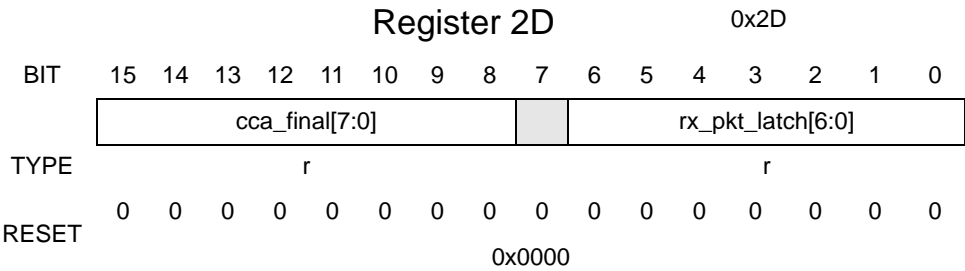


Table 2-30. Register 2D Description

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

2.30 Timestamp_A - Register 2E

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



Table 2-31. Register 2E Description

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

2.31 Timestamp_B - Register 2F

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

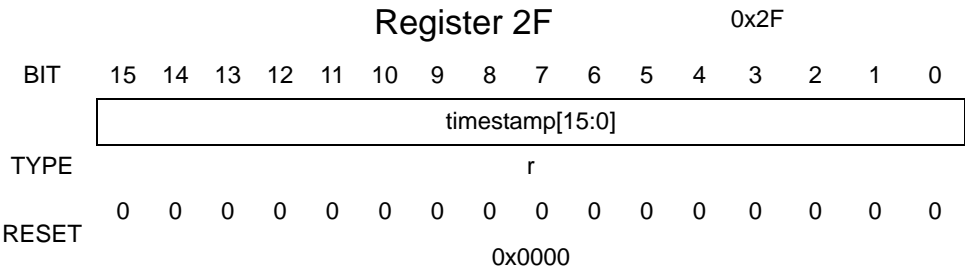


Table 2-32. Register 2F Description

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

Chapter 3

Serial Peripheral Interface (SPI)

3.1 Overview

Control of the MC13191 and data transfers are accomplished by means of a 4-wire Serial Peripheral Interface (SPI). The SPI port is a fully static design that requires no additional clocks besides SPICLK for accessing internal registers, although Packet RAM accesses do require the reference oscillator to be running. This allows for lower power when the SPI must stay alive while the rest of the device is in power-down.

Although the normal SPI protocol is based on 8-bit transfers, the MC13191 imposes a higher level transaction protocol that is based on multiple 8-bit transfers per transaction. In its simplest form, a singular SPI read or write transaction consists of an 8-bit header transfer followed by two 8-bit data transfers. The header denotes access type and register address. The following bytes are read or write data. The SPI also supports recursive 'data burst' transactions in which additional data transfers can occur. The recursive mode is primarily intended for Packet RAM access and fast configuration of the MC13191. Partial word accesses are not supported.

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

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

3.2 SPI Basic Operation

The MC13191 operates as a SPI Slave only. The host microcontroller supplies the interface clock and acts as SPI master.

3.2.1 SPI Pin Definition

The MC13191 SPI signals of CE, CPICLK, MOSI, and MISO are defined in the following paragraphs. A typical interconnection to a microcontroller is shown in [Figure 3-1](#).

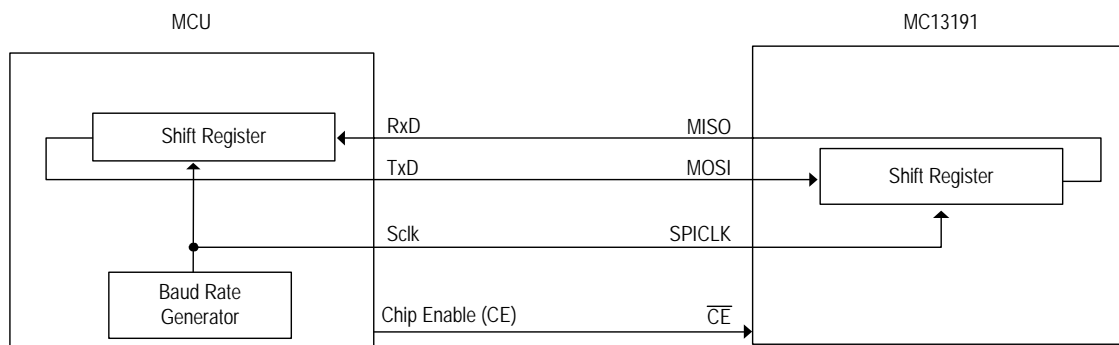


Figure 3-1. Typical SPI Connection with an MCU.

3.2.1.1 Chip Enable (\overline{CE})

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

3.2.1.2 SPI Clock (SPICLK)

The host drives the SPI Clock (SPICLK) input to the MC13191. Data is clocked into the master or slave on the leading (rising) edge of the return-to-zero SPICLK and data out changes state on the trailing (falling) edge of SPICLK.

NOTE

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

3.2.1.3 Master Out / Slave In (MOSI)

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

3.2.1.4 Master In / Slave Out (MISO)

The MC13191 presents data to the master on its MISO output. This output is user configurable for both drive strength and its off state.

3.2.1.4.1 Setting MISO Output Drive Strength

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

NOTE

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

3.2.1.4.2 Setting MISO Off Impedance

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

Writing `miso_hiz_en` to “0” causes MISO to be active low when \overline{CE} is negated. This condition can be used when the transceiver is the only SPI slave.

3.2.2 SPI Burst Operation

The SPI port of an MCU transfers data in bursts of 8 bits with most significant bit (MSB) first. The master (MCU) can send a byte to the slave (transceiver) on the MOSI line and the slave can send a byte to the master on the MISO line. Although an MC13191 transaction is three or more SPI bursts long, the timing of a single SPI burst is shown in [Figure 3-2](#).

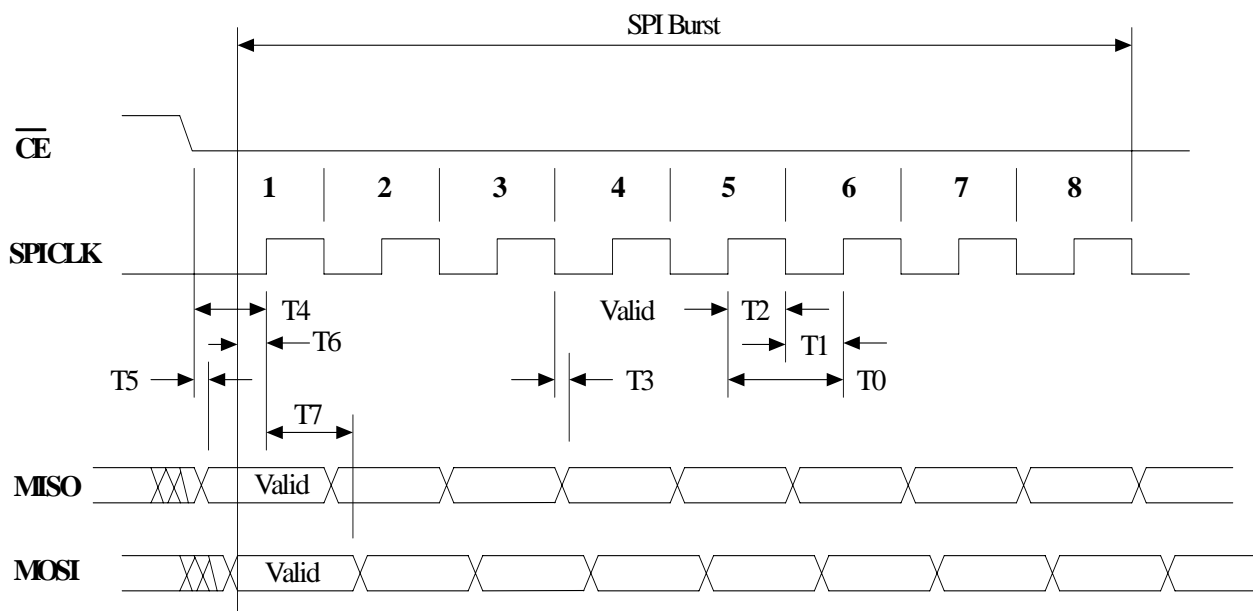


Figure 3-2. SPI Burst Timing Diagram

Table 3-1. SPI Timing Specifications

Symbol	Parameter	Min	Typ	Max	Unit
T0	SPICLK period	125			ns
T1	Pulse width, SPICLK low	62.5			ns
T2	Pulse width, SPICLK high	62.5			ns
T3	Delay time, MISO data valid from falling SPICLK		15		ns
T4	Setup time, \overline{CE} low to rising SPICLK		15		ns
T5	Delay time, MISO valid from \overline{CE} low		15		ns
T6	Setup time, MOSI valid to rising SPICLK		15		ns
T7	Hold time, MOSI valid from rising SPICLK		15		ns

3.3 SPI Singular Transactions

Although the SPI port of an MCU transfers data in bursts of 8 bits, the MC13191 requires that a complete SPI transaction be framed by \overline{CE} , and there will be 3 or more bursts per transaction. There are generally two classes of transactions, which are singular and recursive.

3.3.1 SPI Singular Transaction Signalling

The assertion of \overline{CE} to low signals the start of a transaction. The first SPI burst is a write of an 8-bit header to the transceiver (MOSI is valid) that defines a 6-bit address of the internal resource being accessed and identifies the access as being a read or write operation. In this context, a write is data written to the MC13191 and a read is data written to the SPI master. The following SPI bursts will be either the write data (MOSI is valid) to the transceiver or read data from the transceiver (MISO is valid).

Although the SPI bus is capable of sending data simultaneously between master and slave, the MC13191 never uses this mode. The number of data bytes (payload) will always be 2 for a singular access. After the final SPI burst, \overline{CE} is negated to high to signal the end of the transaction. Should a SPI programming attempt fail to provide the MC13191 with at least 24 rising SPICLK edges prior to \overline{CE} negating, no register data will be changed.

Figure 3-3 shows a read access transaction and Figure 3-4 shows a write access transaction.

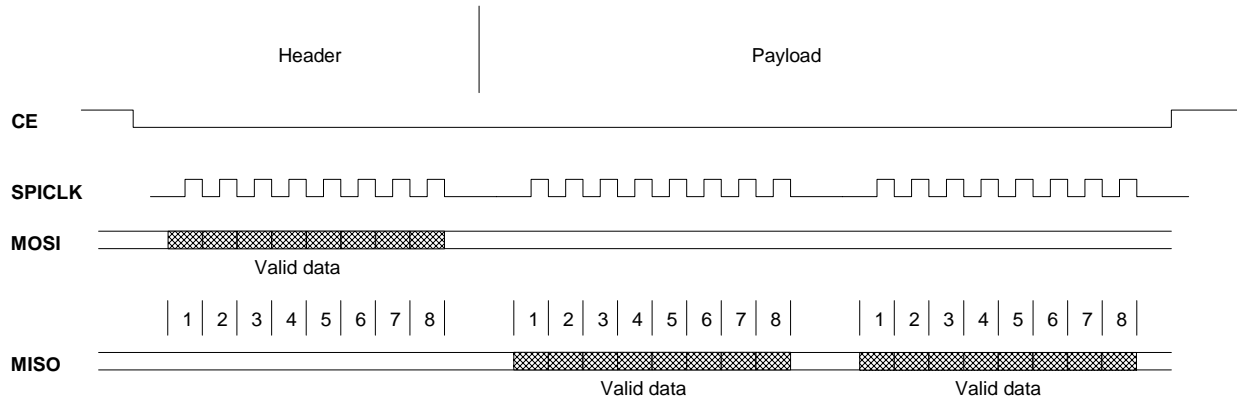


Figure 3-3. Singular Read Access

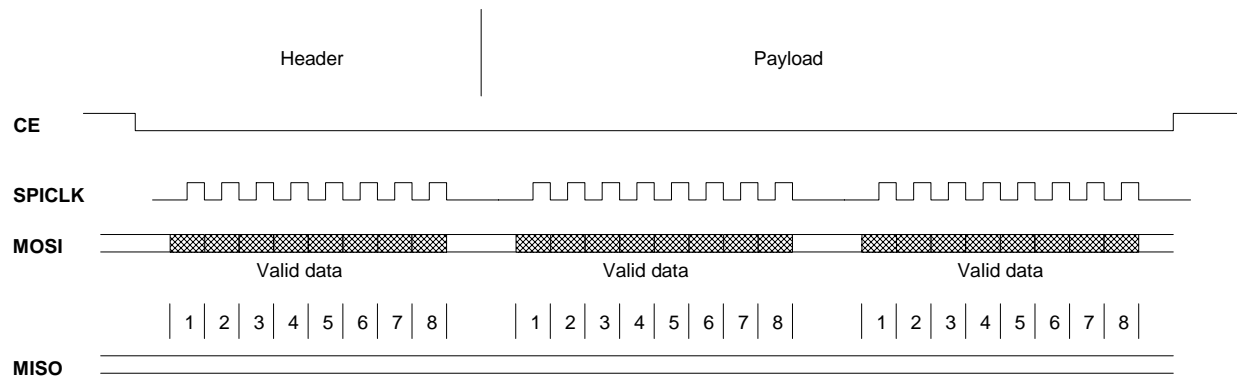


Figure 3-4. Singular Write Access

3.3.2 SPI Singular Transaction Protocol

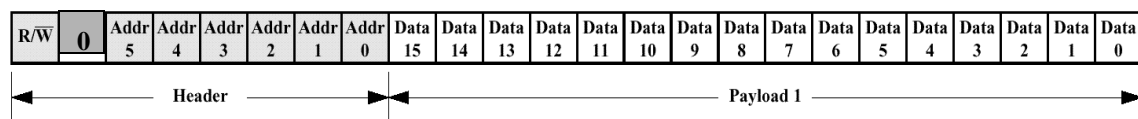


Figure 3-5. SPI Header and Payload Definition

A SPI transaction is divided into a header field and a payload. The header field is always 8 bits, while the payload data is in multiples of 2 bytes or 16 bits. A singular SPI transaction contains 24 bits of information. The header field contains a $\overline{R/\overline{W}}$ bit and a 6-bit register address, as shown in Figure 3-5. The $\overline{R/\overline{W}}$ bit identifies the transfer as a read ($\overline{R/\overline{W}} = 1$) or write ($\overline{R/\overline{W}} = 0$). The lower 6 bits in the header determines which of the 64 possible SPI locations is to be accessed for a read or write operation although not all possible addresses are implemented. The register address field also provides the starting address for a recursive read or write operation as described later.

The register data is presented MSB first.

3.4 Symbol / Data Format

When the transceiver receives over-the-air symbols, they are assembled as 4 symbols per 16-bit word. They are then presented to the RX Packet RAM. The ordering of symbols vs. word bit ordering is shown in [Figure 3-6](#) which details the RX data flow through the device. When the symbols are read via the SPI bus, 2 SPI bursts per 16-bit word are required and the MSB is presented first such that the symbols appear to the MCU in reverse order.

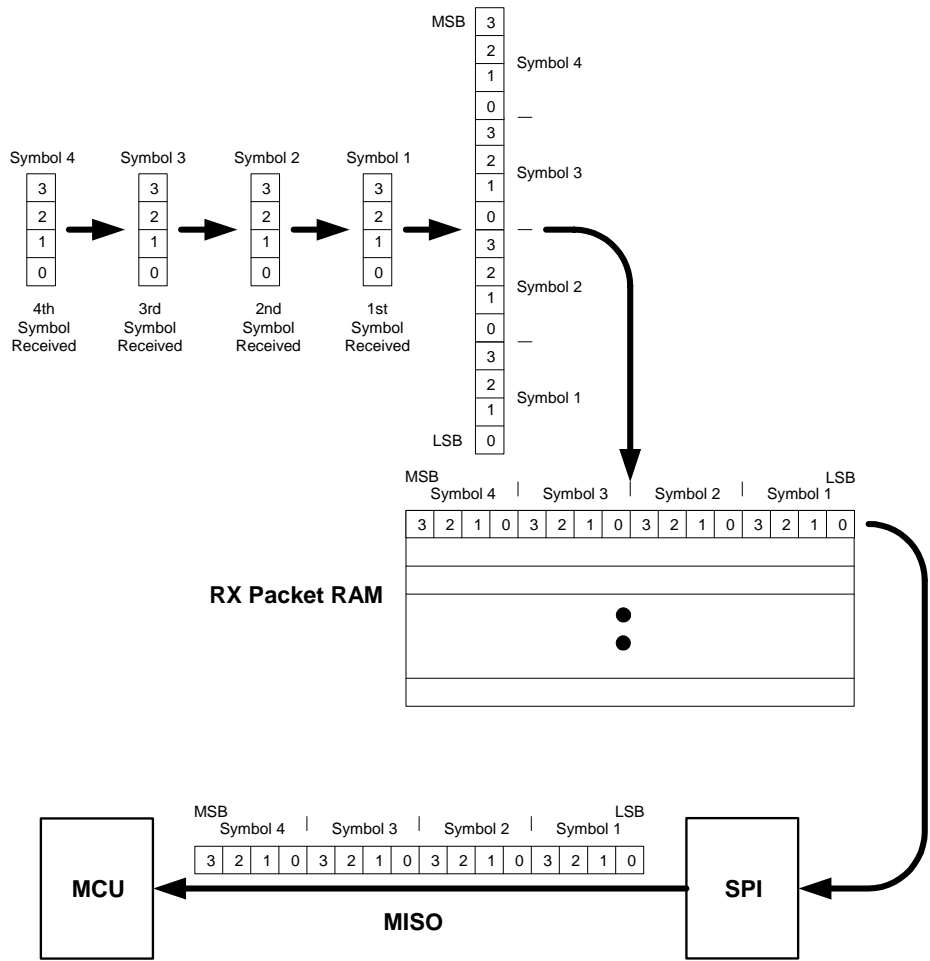


Figure 3-6. RX Symbol Flow Diagram

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

3.5 SPI Recursive Transactions

The MC13191 SPI also incorporates a recursive or ‘data burst’ transaction capability. This allows multiple sequential registers to be accessed with only one header field. Recursive reads and writes provide significant reduction in SPI overhead and a corresponding increase in programming speed.

1. The primary intent is for the software to be able to rapidly configure the MC13191.
2. Recursive reads and writes are convenient for accessing SPI register values which extend beyond the 16-bit SPI register format. Examples include writing Timer Comparator values (tmr_cmp1[23:0] through tmr_cmp4[23:0]) and reading the Time Stamp (timestamp[23:0]).
3. Recursive access capability enables the contents of Packet RAM to be read or written in an expedient manner.

3.5.1 Recursive SPI Register Read

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

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

3.5.2 Recursive SPI Register Write

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

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

3.5.3 Special Case - Packet RAM Access

Packet RAM access is a special case access when the MC13191 is used in the Packet Mode. The MC13191 contains three embedded 128-byte 'Packet RAMs' used to facilitate reception and transmission of packet data. One RAM is dedicated for receive packet data and two are dedicated for transmit packet data. The Packet RAMs are configured in 64-word by 16-bit format and are both read and write accessible via the MC13191 SPI. The recursive data 'burst' access mode is used to efficiently access Packet RAM data.

Although the Packet RAMs are completely static, any RAM access requires the MC13191 to be in an active state; the reference clock circuit must be active. RAM read and write operations are prohibited while the device is in Hibernate or Doze mode.

Reading and writing the MC13191 Packet RAMs is accomplished by SPI burst accesses to dedicated Packet RAM register addresses within the register map. The RX_Pkt_RAM Register 01 is mapped to the Rx Packet RAM, and TX_Pkt_RAM Register 02 is mapped to the Tx Packet RAMs (choice is selected by the tx_ram2_select bit TX_Pkt_Ctl Register 03, Bit 15). The 16-bit data payload of the SPI access maps directly to the 16-bit word in the Packet RAM.

3.5.3.1 Recursive Receive Packet RAM Read Access

The receive Packet RAM is normally accessed when the MC13191 is in packet data mode and a valid frame has been received (as indicated by rx_rcvd_irq and crc_valid). The number of receive data bytes in the queue is shown by the rx_pkt_latch[6:0] field (this includes the full payload with the 2 CRC bytes).

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

3.5.3.1.1 Receive Packet RAM Read Access Flow

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

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

NOTE

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

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

- c) The first word (or 2 bytes) read during a Packet RAM read should be discarded as the internal Packet RAM address is not accessed for the first word read operation. This has the effect of adding 2 bytes to the byte count.
- 3. Do a recursive SPI read transaction where:
 - a) MCU asserts \overline{CE} low.
 - b) MCU sends the MC13191 the first SPI burst with header field of R/W bit = 1 and address field Addr[5:0] = 0x01 for the RX_Pkt_RAM register address.
 - c) MCU reads MC13191 data with the number of SPI byte bursts as calculated in Number 2 above. Note that the first two bytes read from the MC13191 are discarded and that the number of SPI read bursts must be an even number. For an odd number of bytes, the one byte is also discarded.
 - d) MCU negates \overline{CE} high.

3.5.3.1.2 Receive Packet RAM Read Access Error Conditions

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

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

3.5.3.2 Recursive Transmit Packet RAM Write Access

The transmit Packet RAM is normally accessed when the MC13191 is in packet data mode and a frame is to be transmitted. The number of transmit data bytes in the transmit queue is loaded into the tx_pkt_length[6:0] field (this represents the full payload which includes the data bytes stored in the transmit Packet RAM plus the 2 CRC bytes).

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

3.5.3.2.1 Transmit Packet RAM Write Access Flow

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

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

1. Determine which of the two Transmit Packet RAMs are to be used - If Transmit Packet RAM2 is to be used, set status bit `tx_ram2_select`, `TX_Pkt_Ctl` Register 03, Bit 15. The default is Transmit Packet RAM1 selected. Note that the `tx_ram2_select` status determines which transmit Packet RAM is accessed by an SPI transaction as well as which RAM is used during transmit mode.
2. Calculate the number of SPI bursts that are required to write the Tx packet data, noting the following:
 - a) The CRC bytes are not written to Transmit Packet RAM.
 - b) The maximum number of Tx packet data bytes is 125.
 - c) All data written during an access must be done on 16-bit or 2-byte boundaries. Therefore, for an odd number of bytes, the byte count must be rounded up to an even number and an extra dummy byte will be written.
3. Do a recursive SPI write transaction where:
 - a) MCU asserts \overline{CE} low.
 - b) MCU sends the MC13191 the first SPI burst with header field of R/\overline{W} bit = 0 and address field `Addr[5:0] = 0x02` for the `TX_Pkt_RAM` register address.
 - c) MCU writes the MC13191 data with the number of SPI byte bursts as calculated in Step 2. The number of SPI write bursts must be an even number.
 - d) MCU negates \overline{CE} high.

3.5.3.2.2 Transmit Packet RAM Write Access Error Conditions

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

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

3.6 Program Reset (Writing Address 0x00)

A special access is a software reset capability known as a “Software Reset”. When R/ \overline{W} Register Address 0x00 is written, an internal chip reset of the digital core is generated. All synchronous logic in the MC13191 digital core is reset and the SPI register fields are returned to their default values. This Software Reset has the same effect on the MC13191 digital core as asserting the external \overline{RST} pin, except RAM contents are retained.

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

Chapter 4

Modes of Operation

4.1 Operational Modes Summary

The MC13191 has a number of passive operational modes that allow for low-current operation as well as modes where the transceiver is active. These modes are summarized, along with the transition times, in Table 4-1. Figure 4-1 and Figure 4-2 show the state diagrams for different operational modes.

Table 4-1. MC13191 Mode Definitions and Transition Times

Mode	Definition	Transition Time
Off	$\overline{\text{RST}}$ asserted. All IC functions Off, Leakage only. Digital outputs are tri-stated including IRQ. Any RAM buffer data is lost.	25 ms to Idle
Hibernate	Crystal Reference Oscillator Off. SPI not functional. IC Responds to $\overline{\text{ATTN}}$. Data is retained.	20 ms to Idle
Doze	Crystal Reference Oscillator On but CLK0 is available only if Register 7, Bit 9 = 1 for frequencies of 1 MHz or less. (SPI not functional.) Responds to $\overline{\text{ATTN}}$ and can be programmed to enter Idle Mode through an internal timer comparator.	$(300+1/\text{CLK0}) \mu\text{s}$ to Idle
Idle	Crystal Reference Oscillator On with CLK0 output available. SPI active.	
Receive	Crystal Reference Oscillator On. Receiver On.	144 μs from Idle
Transmit	Crystal Reference Oscillator On. Transmitter On.	144 μs from Idle
CCA / Energy Detect	Crystal Reference Oscillator On. Receiver On.	144 μs from Idle

Idle mode is normally the state from which other states are derived. Low power states include the Off, Hibernate, and Doze modes. The Off state is the lowest power and is caused by the hardware reset. Transition from the Off to Idle mode occurs when $\overline{\text{RST}}$ is negated to high. Once in Idle mode, the SPI is active and used to control the MC13191. Transition to Hibernate or Doze modes is enabled via the SPI.

There are active states of Idle, Transmit (TX), Receive (RX), and Clear Channel Assessment (CCA) modes. The transition from the Idle state to the TX or RX is caused by writing to the `xcvr_seq` field. Packet RX and TX can also be modified as timer-based sequences using `Tmr_Cmp2`. Figure 4-1 and Figure 4-2 show state diagrams transceiver operations without and with timer-initiated sequences.

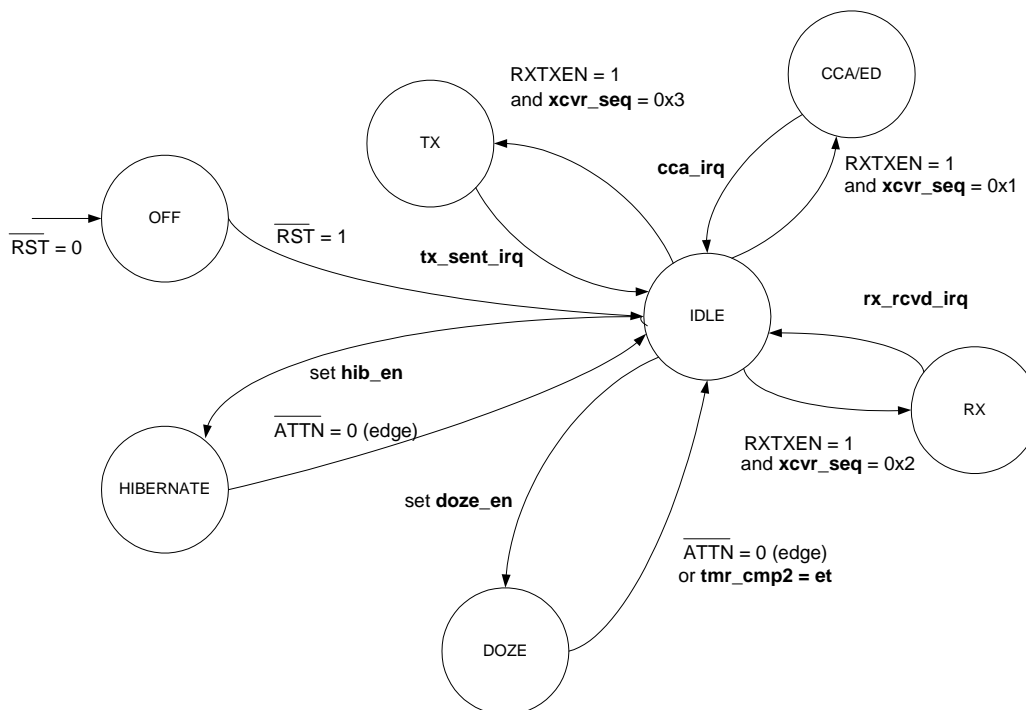


Figure 4-1. State Diagram for Packet Mode Without Timer Enabled States

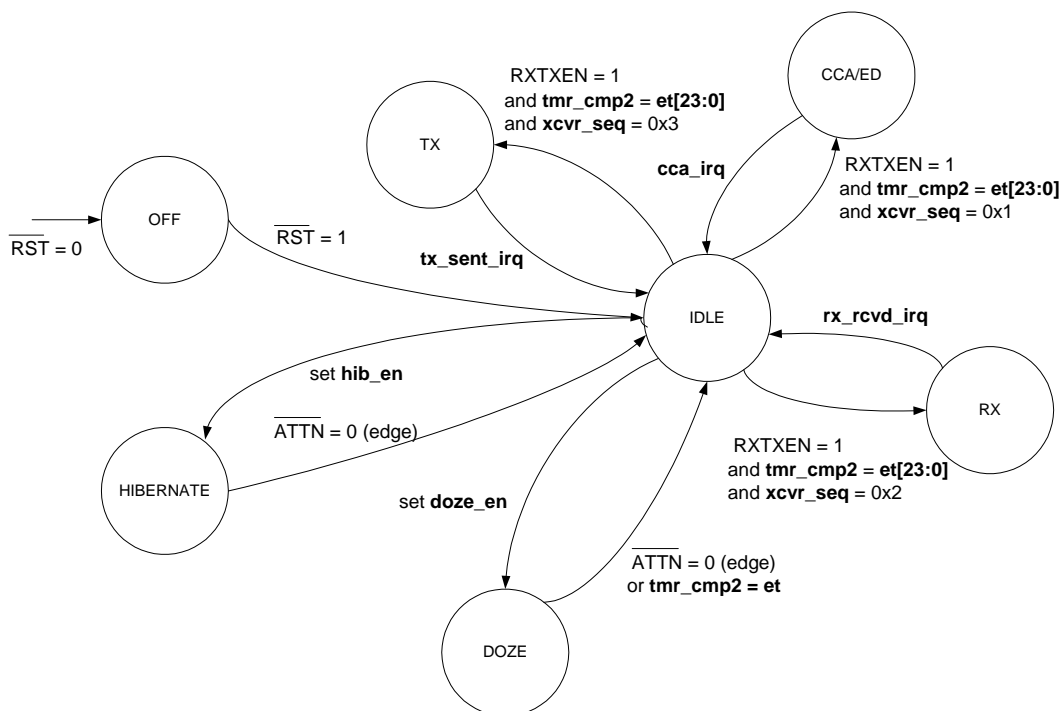


Figure 4-2. State Diagram for Packet Mode With Tmr_Cmp2 Enabled States ($tmr_trig_en = 1$)

4.2 Low Power Modes

The MC13191 supports several low-power modes where the transceiver circuitry is not active. Each mode has a different advantage, these modes are described in the following sections.

4.2.1 Off Mode

The Off or Reset condition has the absolutely lowest power, and is controlled by the $\overline{\text{RST}}$ input. As long as $\overline{\text{RST}}$ is asserted low the MC13191 remains in the Off mode. All functions are disabled and no RAM data is retained. Current draw is attributed to leakage only.

To exit Off mode, $\overline{\text{RST}}$ is negated high. The MC13191 then moves to Idle mode within 25 milliseconds.

4.2.2 Hibernate Mode

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

Hibernate mode is entered from Idle by programming `hib_en`, Control_B Register 07, Bit 1, to “1”. Hibernate is then entered 128 CLKO cycles after `hib_en` is set. The only way to exit from Hibernate mode is to assert $\overline{\text{ATTN}}$ which will cause the MC13191 to go to Idle mode or to assert $\overline{\text{RST}}$.

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

4.2.3 Doze Mode

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

4.2.3.1 Normal Doze Mode

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

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

Doze mode is entered from Idle by programming `doze_en`, Control_B Register 07, Bit 0, to “1”. Doze mode is then entered 128 CLKO cycles after `doze_en` is set.

The intended primary way to exit Doze mode is through a “wake-up” timer and return to Idle at a pre-determined time. This will occur when the Event Timer equals the value in `tmr_cmr2[23:0]`,

Tmr_Cmp2, Registers 1D and 1E. When the match occurs, The MC13191 exits Doze, sets status bit `doze_irq`, IRQ_Status Register 24, Bit 9, and returns to Idle. An interrupt request will be generated if the `doze_mask` IRQ_Mask Register 05, Bit 4, has been set.

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

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

4.2.3.2 Acoma Doze Mode

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

4.3 Active Modes

There are four active modes for the MC13191 which include, Idle, Transmit (TX), Receive (RX) and Clear Channel Assessment (CCA)/Energy Detect (ED).

4.3.1 Idle Mode

Idle Mode is the default mode after leaving one of the low-power modes and is the basic active state from which all other activity is initiated. In Idle Mode, the receiver hardware and transmitter hardware are shut down waiting for a command. The command can instruct the MC13191 to transition to Receive mode, Transmit mode, CCA mode, or to one of the low-power modes. The transition to RX Mode or TX Mode or CCA / ED mode (in its variations) is called by writing to the `xcvr_seq[1:0]` field, Control_A Register 06, Bits 1 - 0.

Once CCA, Receive, or Transmit is entered, the MC13191 will transition back to the Idle mode upon completion of the selected operation. At the end of the operation, the `xcvr_seq[1:0]` field will not be cleared to an Idle value although the transceiver returns to the Idle condition. In this case, a read from the `xcvr_seq[1:0]` field will return the code of the last programmed operation.

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

4.3.2 Controlling Transition to Other Active Modes from Idle

Reviewing the state diagrams in [Figure 4-1](#) and [Figure 4-2](#) shows that the input signal RXTXEN must be asserted to allow transition from Idle to other active states. The recommended procedure is that RXTXEN is taken low while setting-up the desired function (writing required registers) and then after SPI transactions, the MCU raises RXTXEN to a high state enabling the transition. For timed functions (using

either `tmr_cmp2`), the same procedure holds with the exception that the transition will be delayed until the timer function completes.

4.3.3 Packet Mode Data Transfer TX and RX Operation

The Idle mode is the condition from which RX and TX modes are initiated. Writing to the `xcvr_seq[1:0]` field arms the transition to the desired mode. However, the `RXTXEN` signal must also be high for the transition to occur and if the Event Timer is enabled, the transition will be synchronized to the timer compare event. Once Receive or Transmit is entered, the MC13191 will transition back to the Idle mode upon completion of the selected operation.

Table 4-2 shows the transceiver sequence field modes.

Table 4-2. Transceiver Sequence Field (`xcvr_seq[1:0]`)

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

The selected mode is controlled by:

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

4.3.3.1 Packet Receive Mode

Receive mode is the state where the transceiver is waiting for an incoming data frame. The packet receive mode allows the MC13191 to receive the whole packet without intervention from the microcontroller. The entire packet payload is stored in RX Packet RAM and the microcontroller fetches the data after determining the length and validity of the RX packet.

The MC13191 waits for preamble followed by a Start of Frame Delimiter. From there, the Frame Length Indicator is used to determine length of the frame and calculate CRC. The receive function provides the following frame information/data:

1. The frame payload data - accessed through `rx_pkt_ram[15:0]` RX_Pkt_RAM Register 01.
2. CRC valid status - reported by `crc_valid`, IRQ_Status Register 24, Bit 0.
3. Payload data length - reported by `rx_pkt_latch[6:0]`, RX_Pkt_Latch Register 2D, Bits 6 - 0.

4. Link quality indicator (LQI) - this is a measure of the received energy that occurs during the received frame. Once a preamble is detected, the received energy is measured over a 64 μ s period and stored in `cca_final[7:0]`, `RX_Pkt_Latch Register 2D`, Bits 15 - 8.

NOTE

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

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

The application software needs to verify that:

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

NOTE

Because an even number of bytes in packet mode can occasionally cause the RX data to read as all zeroes, the Freescale SMAC causes all frames to be padded on TX such that frame length is an odd number of bytes. If users write their own software, they should also pad TX frame length to be an odd number of bytes.

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

1. The RX frequency must be programmed.
2. If not already low, the MCU sets `RXTXEN` low.
3. Control bit `tmr_trig_en` = 0.
4. `rx_rcvd_mask`, `Control_A Register 06`, Bit 8 is programmed to “1” to enable an interrupt request when the RX packet has been received.
5. Transceiver sequence is programmed to `xcvr_seq[1:0]` = 0x2 for receive.
6. `RXTXEN` must be asserted and held high.
7. When a packet is successfully received, the following are reported:
 - a) `rx_pkt_latch[6:0]`, `RX_Pkt_Latch Register 2D`, Bits 6 - 0 - reports the length of the packet payload including 2 bytes of CRC data.
 - b) `crc_valid`, `IRQ_Status Register 24`, Bit 0 - reports the results of the CRC check, where a “1” indicates valid CRC.
 - c) `cca_final[7:0]`, `RX_Pkt_Latch Register 2D`, Bits 15 - 8 - reports Link Quality Indicator.
 - d) `rx_rcvd_irq`, `IRQ_Status Register 24`, Bit 7- reports the completion of packet reception, where a “1” indicate complete status. Also, an interrupt is generated due to the valid status.
8. In response of the interrupt request from the MC13191, the microcontroller does the following:
 - a) Determines the validity of the frame by reading and checking `rx_rcvd_irq` and `crc_valid`. Determines a valid length for the frame by reading `rx_pkt_latch[6:0]`.

- b) Reads the payload data from RX Packet RAM using a recursive read from rx_pkt_ram[15:0] RX_Pkt_RAM Register 01.

4.3.3.2 Aborting a Packet Receive Sequence

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

4.3.3.3 Packet Transmit Mode

Packet transmit mode allows the MC13191 to send the whole packet without intervention from the microcontroller. The entire packet payload is pre-loaded in TX Packet RAM, the MC13191 sends the frame, and then the transmit complete status is given to the MCU.

NOTE

The packet transmit mode is used by the Freescale SMAC software. On the RX side, an even number of bytes in the frame length can occasionally cause the RX data to read as all zeroes. As a result, the SMAC always pads the data as required to make the frame length an odd number of bytes. If users write their own software, they should also pad TX frame length to be an odd number of bytes.

NOTE

Once a TX sequence is started, it should not be aborted. Wait for the tx_sent_irq status and interrupt. If the TX sequence is aborted, it is best practice to reset the transceiver.

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

1. The TX frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. Control bit tnr_trig_en = 0.
4. tx_sent_mask, Control_A Register 06, Bit 9 is programmed to “1” to enable an interrupt request when the TX packet has been sent.
5. The MCU loads the value of the number of data bytes plus two (for FCS) into tx_pkt_length[6:0] TX_Pkt_Ctl Register 03, Bits 6 - 0.
6. The MCU then pre-loads the number of actual data bytes into tx_pkt_ram[15:0] TX_Pkt_RAM register 02 with a recursive SPI write. An odd number of data bytes requires stuffing a dummy byte due to the 16-bit SPI data format.
7. Transceiver sequence is programmed to xcvr_seq[1:0] = 0x3 for transmit.
8. RXTXEN must be asserted and held high.
9. When the packet is successfully transmitted, tx_sent_irq reports the completion of packet transmission, where a “1” indicates a complete status. Also, an interrupt is generated due to the valid status.

10. In response of the interrupt request from the MC13191, the microcontroller reads the status to clear the interrupt and check successful transmission.

4.3.4 Clear Channel Assessment (CCA) Modes (including Link Quality Indication)

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

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

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

The CCA modes are associated with the following register fields:

1. `cca_type[1:0]`, Control_A Register 06, Bits 5 - 4, determines channel energy assessment algorithm where value 0x1 selects CCA and value 0x2 selects energy detect.
2. `cca_vt[7:0]`, CCA_Thresh Register 04, Bits 15 - 8, sets the threshold level for the CCA function.
3. The average power of the signal is displayed in field `cca_final[7:0]`, RX_Pkt_Latch Register 2D, Bits 15 - 8. This field is used for CCA, ED, and as LQI during an RX operation.
4. Status bit `cca`, IRQ_Status Register 24, Bit 1, is used only for the CCA algorithm and is set to “1” when a busy channel is detected.
5. Status bit `cca_irq_status`, IRQ_Status Register 24, Bit 5, is set to “1” when the power measurement is complete for CCA or ED. If `cca_mask`, Control_A Register 06, Bit 10, is set an interrupt request will be also generated when the `cca_irq_status` is set.
6. For CCA Mode, `tx_strm`, `rx_strm` and `use_strm_mode` control bits must always be cleared to zero.

4.3.4.1 Clear Channel Assessment Function

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

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

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

The value stored in `cca_final[7:0]` is also affected by an offset stored in `power_comp[7:0]`, CCA_Thresh Register 04, Bits 7-0. The default value of `power_comp[7:0]` = 0x8D and is added to the measured value of during the CCA/ED/LQI function to give a correct reading. The default value can be moved up or down to compensate for external gain or loss. As an example, if one were putting in -30 dBm signal and the `cca_final[7:0]` was reporting -36 dBm, then the correction value in `power_comp[7:0]` should be increased by 2 * delta or 2 * 6 in this case. The new compensated value of `power_comp[7:0]` = 0x8D + 0x0C = 0x99.

Since the AGC is set to a fixed gain during the CCA procedure, input signals above -65 dBm will not be reflected correctly due to saturation.

NOTE

The IEEE® 802.15.4 Standard accuracy and range limits are shown for reference.

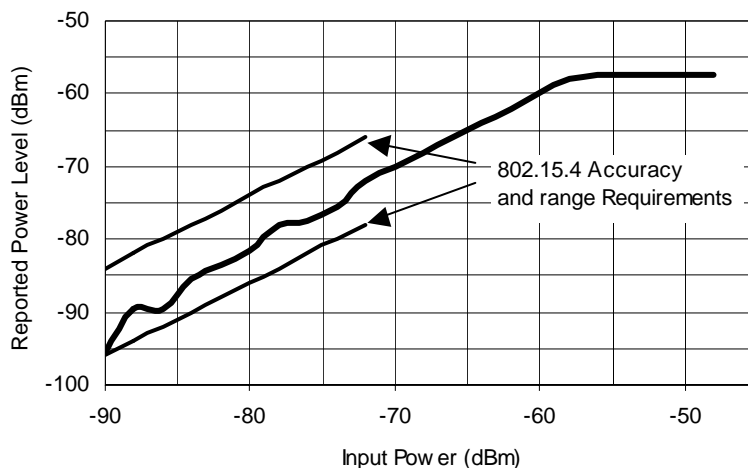


Figure 4-3. CCA Reported Power Level vs. Input Power

The value contained in `cca_final[7:0]`, is compared to the preset threshold of `cca_vt[7:0]`, CCA_Thresh Register 04, Bits 15 - 8. If `cca_final[7:0]` is equal to or less than the threshold, then `cca` is set to 1, indicating a busy channel. If `cca_final[7:0]` is greater than the threshold, `cca` remains at 0. Once the CCA operation is complete, `cca_irq` is asserted.

The value of `cca_vt[7:0]` is calculated:

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

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

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

1. The RX frequency must be programmed.
2. If not already low, the MCU sets `RXTXEN` low.
3. The CCA threshold must be programmed (`cca_vt[7:0] = 0xA4` as an example).
4. `cca_mask`, Control_A Register 06, Bit 10 is programmed to “1” to enable an interrupt request when the CCA operation is complete.
5. `cca_type[1:0]`, Control_A Register 06, Bits 5 - 4, is programmed to “01” to select the CCA algorithm.
6. Transceiver sequence is programmed to `xcvr_seq[1:0] = 0x1` for CCA mode.
7. `RXTXEN` must be asserted and held high.

8. When the measurement is complete, the following are reported:
 - a) `cca_final[7:0]`, RX_Pkt_Latch Register 2D, Bits 15 - 8 - reports the average power level
 - b) `cca`, IRQ_Status Register 24, Bit 1, is set to “1” when a busy channel is detected.
 - c) `cca_irq`, IRQ_Status Register 24, Bit 5, is set to “1” to indicate complete status. Also, an interrupt is generated due to the valid status.
9. In response of the interrupt request from the MC13191, the microcontroller does the following:
 - a) Determines the busy status of the channel by reading and checking `cca_irq` and `cca`.
 - b) If required the power level can be determined by reading `cca_final[7:0]`.

4.3.4.2 Energy Detect Function

With the energy detect algorithm, the exact same procedure results as the CCA operation without the threshold comparison. The receiver warms-up from Idle in 144 μ s and the received power is measured over the next 128 μ s (8 symbol periods). The average power is calculated and stored in `cca_final[7:0]`.

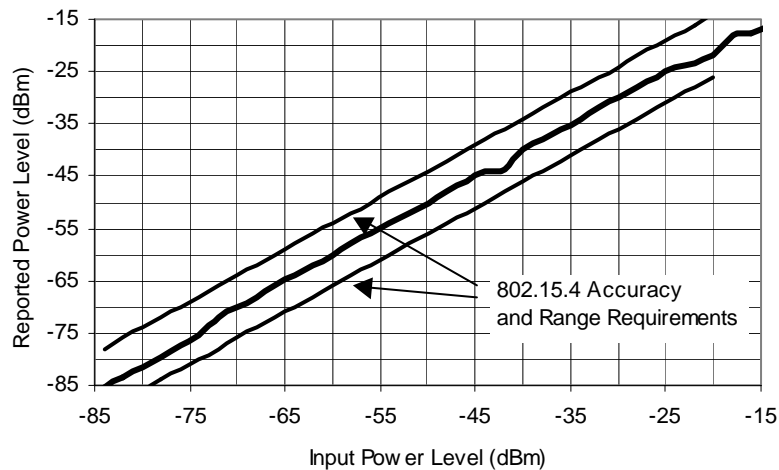


Figure 4-4. ED and LQI Reported Power vs. Input Power

Status bit `cca` is unaffected by energy detect. Once the energy detect operation is complete, `cca_irq` is asserted.

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

1. The RX frequency must be programmed.
2. If not already low, the MCU sets `RXTXEN` low.
3. `cca_mask`, Control_A Register 06, Bit 10 is programmed to “1” to enable an interrupt request when the CCA operation is complete.
4. `cca_type[1:0]`, Control_A Register 06, Bits 5 - 4, is programmed to “10” to select the ED algorithm.
5. Transceiver sequence is programmed to `xcvr_seq[1:0] = 0x1` for CCA mode.

6. RXTXEN must be asserted and held high.
7. When the measurement is complete, the following are reported:
 - a) cca_final[7:0], RX_Pkt_Latch Register 2D, Bits 15 - 8 - reports the average power level
 - b) cca_irq, IRQ_Status Register 24, Bit 5, is set to “1” to indicate complete status. Also, an interrupt is generated due to the valid status.
8. In response of the interrupt request from the MC13191, the microcontroller can determine the power level by reading cca_final[7:0].

4.3.4.3 Link Quality Indication

Link Quality Indication is a measure of the signal quality during an actual receive operation. Its value is stored in field cca_final[7:0], RX_Pkt_Latch Register 2D, Bits 15 - 8. The format for the LQI is the same as CCA:

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

Typical values of LQI returned from an RX operation are from about -95dBm to about -18dBm giving a cca_final[7:0] range of decimal values 190 (0xBE) to 36 (0x24). These are typical and may vary.

4.4 Frequency of Operation

The MC13191 is designed to operate in the 2.4 GHz band, covering 16 channels and using 5 MHz of spacing between each channel. The MC13191 uses two local oscillators (LO). The first LO synthesizer is the main LO for the receiver and the carrier generator for the transmitter. This block is comprised of a Fractional-N (Frac-N) PLL frequency synthesizer.

The fractional and integer components of the Frac-N must be programmed properly to perform a transceiver operation on a particular channel. The channels and the respective, required bit values of the integer setting are shown in the LO1_Int_Div Register 0F, Bits 7-0 (lo1_idiv[7:0]) and the fractional setting are shown in LO1_Num Register 10 (lo1_num[15:0]). See [Table 2-16](#).

4.4.1 Transmit Power Adjustment

[Table 4-3](#) shows the device power output versus SPI register settings for pa_lvl_course[1:0], PA_Lvl Register 12, Bits 7-6 and pa_lvl_fine[1:0], PA_Lvl Register 12, Bits 5-4.

Table 4-3. MC13191 Power Output vs. SPI Settings (Register 12)

PA Power Level Coarse Adjust Reg 12[7:6]	PA Power Level Fine Adjust Reg 12[5:4]	Typical Differential Power at Output Contact (dBm)
0	0	-16.6
0	1	-16.0
0	2	-15.3
0	3	-14.8
1	0	-8.8

Table 4-3. MC13191 Power Output vs. SPI Settings (Register 12) (continued)

PA Power Level Coarse Adjust Reg 12[7:6]	PA Power Level Fine Adjust Reg 12[5:4]	Typical Differential Power at Output Contact (dBm)
1	1	-8.1
1	2	-7.5
1	3	-6.9
2	0	-1.0
2	1	-0.5
2	2	0.0
2	3	0.4 (default)
3	0	2.1
3	1	2.8
3	2	3.5
3	3	3.6

4.5 2.4GHz PLL Out-of-Lock Interrupt

Successful wireless data transmission and reception is predicated on the proper channel frequency being maintained internally by the MC13191. Sophisticated control circuitry and design techniques assure that the internal 2.4GHz local oscillator stays on the selected channel frequency. In the unusual event that the PLL loses lock, the host is notified via the 'Out-of-Lock Interrupt'. If lock indication circuitry indicates an out-of-lock condition, status bit `pll_lock_irq`, `IRQ_Status` Register 24, Bit 15, is set and any RX or TX operation in progress is automatically terminated. The MC13191 returns immediately to the IDLE state to await interrupt service routine handling. Also, the $\overline{\text{IRQ}}$ is asserted provided the mask bit `pll_lock_mask`, `IRQ_Mask` Register 05, Bit 9, is set.

Chapter 5

Timer Information

5.1 Event Timer Block

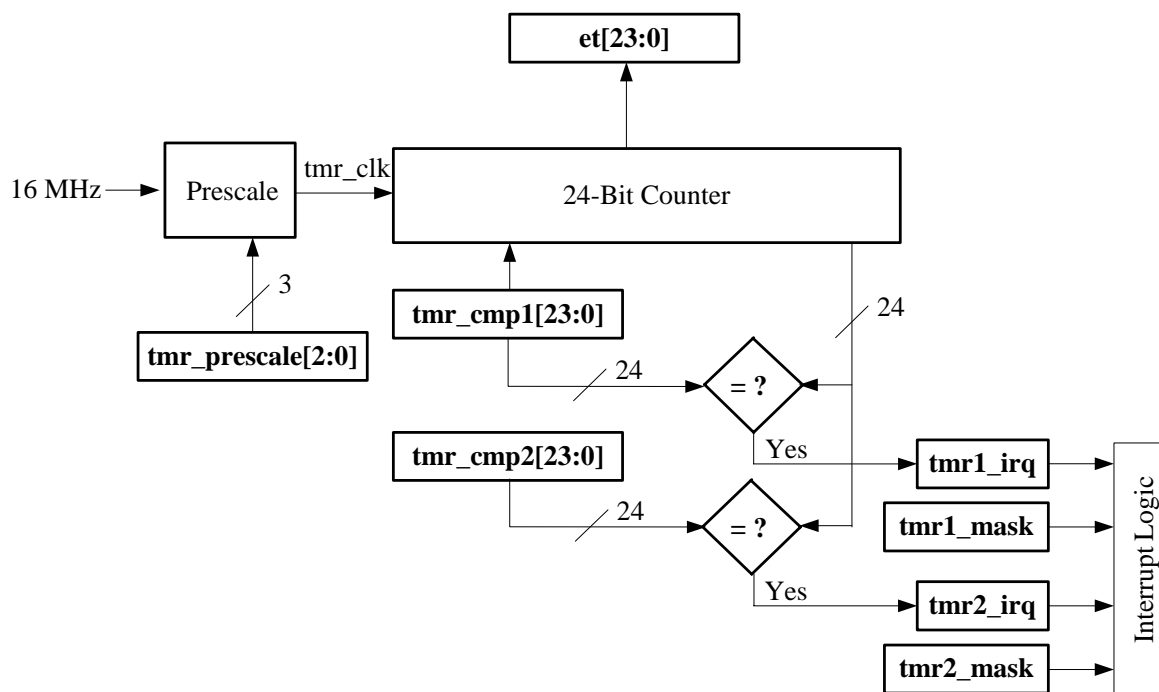


Figure 5-1. Event Timer Block Diagram

The MC13191 contains an internal Event Timer block that manages system timing. A simplified block diagram is shown in [Figure 5-1](#). The Event Timer consists of a prescaler and a 24-bit counter which increment whenever the crystal clock is operating. Interrupts to the MCU may be generated when the “current time” of the counter (et[23:0]) matches several pre-determined values set in registers via SPI write operations. The current time is accessible at any time via a SPI read operation, as well as, programmable via a SPI write operation. The Event Timer provides the following functions:

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

5.2 Event Timer Time Base

The Event Timer’s base clock (tmr_clk) is derived from a programmable prescaler which is clocked by the 16 MHz crystal source. The prescaler provides counter input frequencies from 2 MHz down to 15.625 kHz, which sets the granularity and resolution of the current time. The prescaler, and thus the Event

Timer only increment when the crystal oscillator is active. The field `tmr_prescale[2:0]` Control_C Register 9, Bits 2-0 ([Section 2.12](#)) establishes the `tmr_clk` frequency as shown in [Table 5-1](#).

Table 5-1. Event Timer Prescaler Settings

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

The 24-bit counter automatically rolls over upon reaching its maximum value, and the corresponding maximum possible Event Timer durations are also provided in [Table 5-1](#).

5.3 Setting Current Time

“Current Time” is defined as the value of the Event Timer internal counter. The current time is programmable, but does not have to be programmed. In the reset condition, the MC13191 current time is set to zero. Current time advances from zero at the `tmr_clk` clock rate and rolls over to zero after reaching its maximum value.

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

1. `Tmr_Cmp1_A` Register 1B, Bits 7-0, `tmr_cmp1[23:16]`
2. `Tmr_Cmp1_B` Register 1C, Bits 15-0, `tmr_cmp1[15:0]`
3. `Control_B` Register 07, Bit 15, `tmr_load`

When field `tmr_load` is programmed to high, the value of “current time” is set to the value in `tmr_cmp1[23:0]`. Thus, `tmr_cmp1[23:0]` is first programmed to the desired current time value, then `tmr_load` is programmed to 1, which initiates the timer load. The change to the “current time” value occurs within two crystal clock cycles, after which normal incrementing resumes on the next rising `tmr_clk` edge. So, `tmr_load` is not required to be programmed to zero for the Event Timer to resume normal operation. However, loading the Event Timer is a positive edge-triggered event, so `tmr_load` must be programmed low prior to the next attempt to load the Event Timer.

5.4 Reading Current Time

The current value of the Event Timer can be read via the SPI using `et[23:16]`, `Current_Time_A` Register 26, Bits 7-0, and `et[15:0]`, `Current_Time_B` Register 27, Bits 15-0. The “current time” may be obtained using two single SPI reads, or one recursive 2-word SPI read (or as part of a longer recursive read operation as well). It is important to realize that the Event Timer may increment during these recursive SPI read

operations, or between successive SPI reads if single SPI reads are used. During such an access, the MC13191 latches the “current time” to protect the host from obtaining an incorrect value. The “current time” least significant 16 bits (LSB) are latched when the most significant 8 bits (MSB) SPI location is read. The LSB is unlatched after the “current time” LSB location is read. This guarantees a stable value until the host completes a read of both words constituting the “current time” before it is allowed to update.

The preferred procedure to obtain the “current time” value from the MC13191 is to perform a 2-word recursive read of the “current time” starting at the MSB address.

5.5 Latching the Timestamp

The MC13191 has the ability create a Timestamp or to latch a copy of the “current time” while continuing to increment its internal counter. This timestamp value latched within the Event Timer corresponds to the beginning of a receive packet where the actual payload data begins after the FLI has been received. The timestamp[23:0] (Register 2E, Bits 7-0 and Register 2F, bits 15-0) value is read from the MC13191 by the host. When timestamp[23:0] is latched, its value corresponds to the “current time” value coincident with the reception of rx_pkt_latch[6:0], RX_Pkt_Latch Register 2D, Bits 6-0. The timestamp remains latched until another packet is received, at which point the timestamp[23:0] is updated and re-latched.

5.6 Event Timer Comparators

The MC13191 incorporates two full 24-bit programmable fields that compare to the Event Timer’s “current time”. The intent of these compares is to enable the host to schedule events relative to the “current time”. When a match between the “current time” and any one of the two timer compare values occurs, a corresponding flag is sent to internal interrupt logic. This causes the appropriate bit in the IRQ_Status Register 24 to be set, and depending on the interrupt mask control bit, generate an interrupt event on the IRQ pin.

5.6.1 Timer Compare Fields

There are two 24-bit timer compare fields:

1. tmr_cmp1[23:0], Tmr_Cmp1_A Register 1B, Bits 7-0, and Tmr_Cmp1_B Register 1C, Bits 15-0.
2. tmr_cmp2[23:0], Tmr_Cmp2_A Register 1D, Bits 7-0, and Tmr_Cmp2_B Register 1E, Bits 15-0.

5.7 Timer Disable Bits

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

1. tmr_cmp1_dis, Tmr_Cmp1_A Register 1B, Bit 15.
2. tmr_cmp2_dis, Tmr_Cmp2_A Register 1D, Bit 15.

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

5.7.1 Timer Status Flags

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

1. `tmr1_irq`, `IRQ_Status` Register 24, Bit 8.
2. `tmr2_irq`, `IRQ_Status` Register 24, Bit 2.

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

5.7.2 Timer Interrupt Masks

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

1. `tmr1_mask`, `IRQ_Mask` Register 05, Bit 0.
2. `tmr2_mask`, `IRQ_Mask` Register 05, Bit 1.

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

5.7.3 Setting Compare Values

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

It is important to realize that not all bits of the timer compare value are updated simultaneously within the SPI. To prevent the Event Timer from generating a false match to a partially updated timer compare value, the compare hardware is inhibited temporarily. The inhibit feature initiates when the address of the MSB location of the timer compare field is decoded on a SPI write, and ends when a write to the LSB field is completed. Thus, once a SPI write to the MSB location starts, the comparator is disabled until a SPI write to the LSB location is completed. The preferred procedure for software to change a timer compare value within the MC13191 is to perform a 2-word recursive write of the timer compare field starting at the MSB address.

5.8 Intended Event Timer Usage

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

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

NOTE

The timer_compare functions exit reset with the timer function enabled but with the interrupts masked off. Users should disable all timers and clear the IRQ_Status Register via a read as part of system initialization after reset.

5.8.1 Generating Time-Based Interrupts

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

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

5.8.2 Using tmr_cmp2[23:0] to Exit Doze Mode

The Event Timer provides a timer-based mechanism to bring the MC13191 out of Doze mode. The MC13191 is put into Doze mode when doze_en, Control_B Register 07, Bit 0, is programmed high. While in Doze mode, a match between “current time” and field tmr_cmp2[23:0] causes the MC13191 to exit Doze mode and return to Idle Mode.

The general procedure is as follows:

1. Read the “current time” value from et[23:0].
2. Add an offset to this value to equal desired “future time” to exit Doze mode.
3. Program field tmr_cmp2[23:0] to value “future time”.
4. Program doze_mask, Register 05, Bit 4, to 1.
5. Program doze_en, Register 7, Bit 0, to 1. The MC13191 then enters Doze mode. (Note that the control bit tmr_cmp2_dis has no effect on this mode).
6. When “current time” equals tmr_cmp2[23:0], the MC13191 exits Doze mode, and doze_irq, Register 24, Bit 9, gets set. An external interrupt is also generated because doze_mask is set.

NOTE

The MC13191 can always be taken out of Doze Mode by asserting $\overline{\text{ATTN}}$ or $\overline{\text{RST}}$. Also, if acom_en IRQ_Mask Register 05, Bit 8 is set before entering Doze mode, the Event Timer logic is disabled for additional power savings and only $\overline{\text{ATTN}}$ or $\overline{\text{RST}}$ will cause exit of Doze mode.

5.8.3 Timer-Triggered Transceiver Events

An Event Timer can be used to initiate the MC13191 transceiver operations such as transmit and receive. The desired operation can be scheduled to commence at a future time greater than the “current time” by using the MC13191 timer-triggered operation capability. Timer-triggered operations are invoked by using tmr_cmp2 [23:0]. A time greater than the “current time” is programmed into the appropriate compare field and tmr_trig_en , Control_A Register 6, Bit 7 is programmed high. When the “current time” advances to match the value set in the compare field, the selected operation sequence will commence automatically without intervention from the host. This allows the host to arm the MC13191 to execute a desired operation at a future time, and go off to perform other necessary system functions.

The general procedure is as follows:

1. Desired frequency must be programmed.
2. If not already low, the MCU sets RXTXEN low.
3. Read the “current time” value from et [23:0].
4. Add an offset to this value to equal desired “future time” to initiate selected operation.
5. Program field tmr_cmp2 [23:0] to value “future time”.
6. Program tmr_cmp2_dis to 0 to enable the compare function.
7. If desired, program tmr2_mask , IRQ_Mask Register 05, Bit 1 high to enable an interrupt when the timer compare function completes and starts the transceiver.
8. For a TX operation only, load tx_pkt_length [6:0] and payload data into tx_pkt_RAM [15:0].
9. Program tmr_trig_en , Control_A Register 6, Bit 7 high to enable a timer-based operation.
10. Program the MC13191 for the desired transceiver operation via xcvr_seq [1:0].
11. Assert the RXTXEN pin and hold high.
12. When “current time” equals tmr_cmp2 [23:0], the MC13191 initiates the selected transceiver operation. When tmr2_irq , IRQ_Status Register 24, Bit 2 is set to 1, an external interrupt is generated if the interrupt mask bit (tmr2_mask) was set high.

NOTE

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

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

Chapter 6

Interrupt Description

6.1 Interrupts

Interrupts provide a way for the MC13191 to inform the host microcontroller (MCU) of onboard events without requiring the MCU to constantly query MC13191 status.

For a given event, the interrupt flow is as follows.

- The source interrupt mask is enabled.
- The source function is enabled.
- The source event occurs causing the source status flag to be set and the $\overline{\text{IRQ}}$ pin to be asserted low.
- The $\overline{\text{IRQ}}$ pin stays asserted until the $\overline{\text{IRQ}}$ _Status Register is read to determine the source of the interrupt. Reading the IRQ_Status Register clears the status bits and releases the $\overline{\text{IRQ}}$ signal to be negated high.

When multiple source events have occurred, the MCU must use the IRQ_Status register contents to determine all the present events that caused an interrupt, prioritize them, and respond to all of the them. This is done through the interrupt service routine of the MCU.

6.1.1 Interrupt Sources

Table 6-1 lists the interrupt status bits, mask bits, and source description.

Table 6-1. MC13191 Interrupt Sources

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism ¹
1	pll_lock_irq	pll_lock_mask	PLL out of lock.	Read IRQ_Status Reg
2	ram_addr_err	ram_addr_mask	RAM address error - a recursive access to Packet RAM has exceeded the maximum RAM address.	Read IRQ_Status Reg
3	arb_busy_err	arb_busy_mask	Arbitration busy error - a SPI access to Packet RAM was attempted during packet reception or transmission.	Read IRQ_Status Reg
4	attn_irq	attn_mask	The $\overline{\text{ATTN}}$ signal has been asserted or the MC13191 has reached a Power-up complete condition after a reset.	Read IRQ_Status Reg
5	doze_irq	doze_mask	While in Doze mode, a tmr_cmp2 match has occurred and the MC13191 will return to Idle mode.	Read IRQ_Status Reg
6	rx_rcvd_irq	rx_rcvd_mask	The current RX packet has been received, data in Packet RAM is ready to be read, and the transceiver has returned to Idle Mode.	Read IRQ_Status Reg

Table 6-1. MC13191 Interrupt Sources (continued)

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism ¹
7	tx_sent_irq	tx_sent_mask	The current TX packet in Packet RAM has been completely transmitted, and the transceiver has returned to Idle Mode.	Read IRQ_Status Reg
8	cca_irq	cca_mask	The Clear Channel Assessment operation has been completed.	Read IRQ_Status Reg
9	tmr1_irq	tmr1_mask	Tmr_cmp1 match has been made.	Read IRQ_Status Reg or set tmr_cmp1_dis bit
10	tmr2_irq	tmr2_mask	Tmr_cmp2 match has been made. (Not functional when Tmr_cmp2 is used to exit Doze Mode).	Read IRQ_Status Reg or set tmr_cmp2_dis bit

¹ Although some status bits can be cleared by other means, reading IRQ_Status register will always clear all status bits.

6.1.2 Output Pin $\overline{\text{IRQ}}$

The $\overline{\text{IRQ}}$ signal is an open drain output that is asserted low when an interrupt request is pending. The signal is released to high by reading the IRQ_Status register via an SPI transaction. $\overline{\text{IRQ}}$ is an open drain output that requires a passive pullup and it also can be programmed for drive strength.

6.1.2.1 Programming $\overline{\text{IRQ}}$ Pullup

A passive pullup is required on $\overline{\text{IRQ}}$ and may be done via two methods:

1. Use the onboard (nominal 40 Kohm) pullup resistor - Set irqb_pup_en bit, GPIO_Data_Out Register 0C, Bit 7, to activate. This is the default mode.
2. Use an external resistor (value should be greater than 4 kilohms).

6.1.2.2 Setting $\overline{\text{IRQ}}$ Output Drive Strength

$\overline{\text{IRQ}}$ output drive strength is programmed by writing to irqb_drv[1:0], GPIO_Data_Out Register 0C. There are 4 levels of drive strength with field value 00 for lowest and value 11 for greatest. The default value is 00.

NOTE

It is suggested the user program $\overline{\text{IRQ}}$ for greatest drive strength for best performance.

6.1.3 Interrupts from Exiting Low Power Modes

The MC13191 has three low power modes and interrupt generation differs somewhat for each mode.

6.1.3.1 Exiting Off Mode (Reset)

The transceiver is put in reset and stays in reset (Off Mode) through the assertion of $\overline{\text{RST}}$. The initialization done at reset enables `attn_mask` which allows an interrupt request when `attn_irq` is set. One condition that sets `attn_irq` is when the transceiver exits reset after $\overline{\text{RST}}$ is released high. As a result, an interrupt request will always be generated by `attn_irq` status when reset is exited.

6.1.3.2 Exiting Hibernate Mode

Hibernate is normally only exited through assertion of $\overline{\text{ATTN}}$ (obviously $\overline{\text{RST}}$ can still override). The `attn_irq` status will be set by the assertion of $\overline{\text{ATTN}}$. If an interrupt is desired to signify the event, the `attn_mask` bit must be set before entering Hibernate. The interrupt request will then be generated due to the `attn_irq` being set true upon exit from Hibernate.

6.1.3.3 Exiting Doze Mode(s)

Doze can be exited via assertion of $\overline{\text{ATTN}}$ or through use of `tmr_cmp2` (again reset can override). Asserting $\overline{\text{ATTN}}$ will always cause Doze to be exited even if the timer option is enabled. If an interrupt is desired, set `attn_mask` before entering Doze which will cause the interrupt when the `attn_irq` status is set upon exiting Doze due to $\overline{\text{ATTN}}$.

Alternately, Doze has the option of using `tmr_cmp2` to exit, except for Acoma Mode which cannot use the timer. When `tmr_cmp2` match occurs the `doze_irq` status will be set. An interrupt request will also occur if `doze_mask` bit has been enabled.

Chapter 7

Miscellaneous Functions

7.1 Reset Function

The MC13191 can be placed in one of two reset conditions either through hardware input $\overline{\text{RST}}$ or by writing to Reset Register 00.

7.1.1 Input Pin $\overline{\text{RST}}$

Asserting input pin $\overline{\text{RST}}$ low places the transceiver in a complete reset condition (Off Mode and power down), and the device stays in this reset mode until $\overline{\text{RST}}$ is released high. After $\overline{\text{RST}}$ is released, the transceiver will transition to the Idle Mode within 25 milliseconds

7.1.2 Software Reset (Writing to Register 00)

Writing to Reset Register 00 causes a reset condition where the digital logic is reset, but the transceiver is not powered down. The device is forced to the Idle Mode and the SPI registers are all reset and forced to their default condition although all data in the Packet RAMs is retained. The reset is held as long as $\overline{\text{CE}}$ remains asserted and is released when $\overline{\text{CE}}$ is negated high.

7.1.3 Reset Indicator Bit (RST_Ind Register 25, Bit 7)

It is useful to determine if the transceiver has powered-up from a reset condition or from a low power state that was released via the $\overline{\text{ATTN}}$ signal. The reset indicator bit (reset_ind, RST_Ind Register 25, Bit 7) is cleared during a reset operation but not during a low power mode such as Doze or Hibernate. The reset_ind bit gets set by the first read of Register 25 after a reset operation and stays set until another reset operation.

When exiting reset, an interrupt is generated by attn_irq, IRQ_Status Register 24, Bit 10, (the default condition is with the interrupt mask enabled). This same interrupt can be enabled for exiting Hibernate or Doze via an $\overline{\text{ATTN}}$ assertion. As a result, the reset_ind bit can determine if the power-up condition is from the reset condition or a Doze or Hibernate condition.

After exiting reset and responding to the attn_irq interrupt, users should read Register 25 which in turn sets the reset_ind bit. Thereafter, if the transceiver is put into Doze or Hibernate and then later awakened by an $\overline{\text{ATTN}}$ assertion, the attn_irq interrupt is also used, but the reset_ind is set signifying that the chip was not reset and does not need re-initialized.

7.2 General Purpose Input/Output

The MC13191 has seven general purpose input/output (GPIO) pins (GPIO1 through GPIO7). Features include:

- CMOS logic levels with +/- 1 mA load current.
- Programmable as inputs or outputs.
- During reset outputs are disabled and exit reset as inputs.
- Not capable of generating an interrupt.
- Once programmed as an output, a GPIO keeps its state if the transceiver transitions to Doze or Hibernate mode.

7.2.1 Configuring GPIO Direction

The GPIO are configured using GPIO_Dir Register 0B. Each I/O has a `gpiox_oen` output enable bit and a `gpiox_ien` input enable bit. Exiting reset, the default condition for these enable bits is that the `gpiox_ien` bits are set to 1 which enables the pins as inputs and `gpiox_oen` are cleared.

NOTE

If any bit is programmed to be an input and output simultaneously, the input condition overrides.

7.2.2 Setting GPIO Output Drive Strength

If any GPIO are programmed as outputs, their drive strength is programmable. GPIO1 through GPIO4 are programmed as a group for drive strength by writing to control field `gpio1234_drv[1:0]`, GPIO_Dir Register 0B, and GPIO5 through GPIO7 are programmed as a group by writing to `gpio567_drv[1:0]`, GPIO_Data_Out Register 0C. There are 4 levels of drive strength with field value 00 for lowest and value 11 for greatest.

7.2.3 Programming GPIO Output Value

GPIO_Data_Out Register 0C has a `gpiox_o` bit for each GPIO pin that establishes the corresponding output's state when that I/O is programmed as an output. Setting a `gpiox_o` to 1 sets the output high.

7.2.4 Reading GPIO Input State

GPIO_Data_In Register 28 has a `gpiox_i` bit for each GPIO pin. When a GPIO is programmed as an input, its state can be determined by reading the corresponding `gpiox_i` bit in the register.

7.3 Crystal Oscillator

The crystal oscillator for the MC13191 uses the following external pins:

1. XTAL1 - reference oscillator input.
2. XTAL2 - reference oscillator output. Note that this pin should not be loaded to be used as a reference source or to measure frequency; instead use CLK0 to measure or supply 16 MHz.

The external crystal circuit is shown in [Figure 7-1](#).

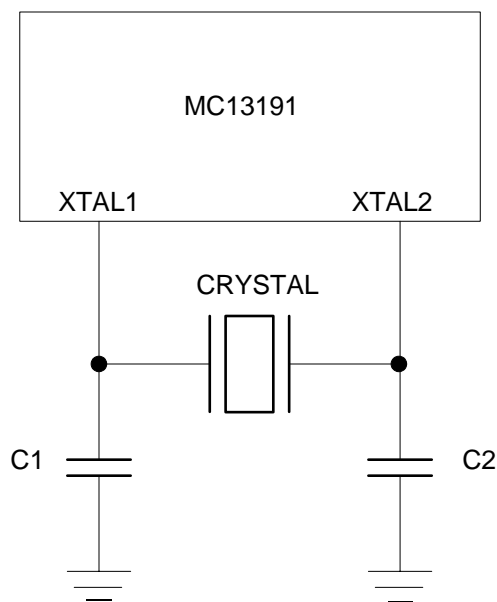


Figure 7-1. Crystal oscillator circuit

7.3.1 Crystal Requirements

The MC13191 requires that only a 16 MHz crystal with a <9 pF load capacitance can be used. The load capacitance limitation is required due to internal oscillator circuit and the ability to trim the oscillator as described in the next section. A tight frequency tolerance on the crystal may also be required due to the IEEE[®] 802.15.4 specification which demands that frequency tolerances be kept within ± 40 ppm. This requirement is for the oscillator circuit, not just the crystal. This is covered in detail in the MC13191 Data Sheet.

7.3.2 Crystal Trim Operation

The MC13191 uses the 16 MHz crystal oscillator with warp capability as the reference oscillator for the system. The warp capability is done by the MC13191 and is controlled by programming CLK0_Ctl Register 0A, Bits 15-8 (xtal_trim[7:0]). The trimming procedure varies the frequency by a few hertz per step, depending on the type of crystal. The high end of the frequency spectrum is set when xtal_trim[7:0] is set to zero. As xtal_trim[7:0] is increased, the frequency is decreased. Accuracy of this feature can be observed by varying xtal_trim[7:0] and using a spectrum analyzer or frequency counter to track the change in frequency of the crystal signal. The reference oscillator frequency can be measured at the CLK0 contact by programming CLK0_Ctl Register 0A, Bits 2-0, to value 000. The crystal frequency should not be monitored at IC pins 26 or 27 (XTAL1 or XTAL2) because this will load the oscillator and alter the frequency.

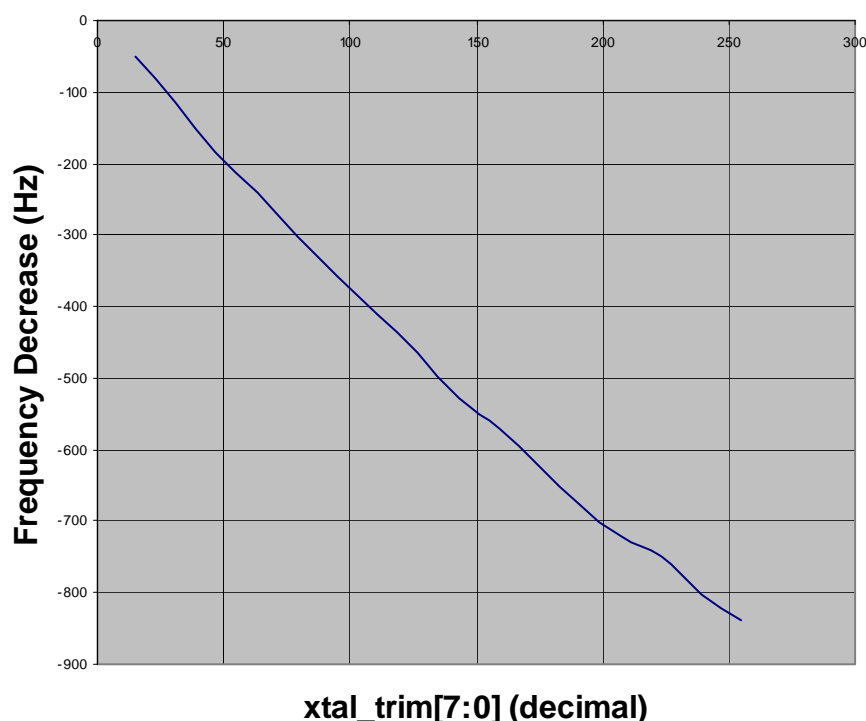


Figure 7-2. Crystal Frequency Variation vs. xtal_trim[7:0]

Figure 7-2 shows typical oscillator frequency decrease versus the value programmed in xtal_trim[7:0].

7.4 Output Clock Pin CLKO

The MC13191 can supply a clock output useful as a frequency source for a microcontroller, frequency test point, or reference for other uses. The clock output is available on signal CLKO and can be turned on or off as a power saving measure (default is CLKO active). CLKO is controlled by a number of control fields.

7.4.1 Enable CLKO (clko_en, Control_C Register 09, Bit 5)

Setting clko_en, Contro_C Register 9, Bit 5, to 1 enables the CLKO signal. The default condition out of reset is that the clock out is enabled at the default frequency of 32.768+ kHz set by field clko_rate[2:0].

7.4.2 Setting CLKO frequency (clko_rate[2:0], CLKO_Ctl Register 0A, Bits 2-0)

The 3-bit field clko_rate[2:0], CLKO_Ctl Register 0A, Bits 2-0, selects the output frequency based on the programmed value. Frequencies from 16 MHz to 16 kHz are available. Default frequency is 32.768+ kHz with a field value of clko_rate[2:0] = 110. Table 2-12 lists the CLKO frequencies versus clko_rate[2:0] program value.

7.4.3 Enable CLKO During Doze Mode (clko_doze_en, Control_B Register 07, Bit 9)

Bit clko_doze_en, Control_B register 07, Bit 9, is used to control CLKO during Doze mode. If clko_doze_en is set to 1 before entering Doze mode, CLKO will continue to toggle while the MC13191 is in Doze mode. The CLKO frequency must be set for 1 MHz or lower. Default out of reset is clko_doze_en = 0 as a power-saving measure.

If clko_doze_en = 0 and CLKO was enabled, CLKO will stop toggling 128 reference clock (16 MHz) cycles after the doze_en bit is programmed to 1. CLKO will automatically re-start after exiting Doze with the exception of the two lowest frequencies.

NOTE

The two lowest frequencies of 16.393+ kHz and 32.786+ kHz will not restart directly when exiting Doze mode. To restart CLKO for these frequencies, the clko_en, Control_C Register 09, Bit 2, must be cleared and set again.

7.4.4 Setting CLKO Output Drive Strength (clko_drv[1:0], GPIO_Data_Out Register 0C, Bits 11-10)

The CLKO output drive strength can be programmed to 4 different levels by writing to clko_drv[1:0], GPIO_Data_Out Register 0C, Bits 11-10. The default value is the lowest drive value of 00. Note that for higher frequencies such as 16 MHz, the CLKO must be programmed for highest drive. Table 7-1 shows output drive strength for maximum frequency and maximum load capacitance.

Table 7-1. CLKO Drive Strength Versus clko_drv[1:0] Value

Drive Strength (clko_drv[1:0])	Max Freq (MHz)	Max C _{load} (pF)
00	1	20
01	8	20
10	16	20
11	16	20 < C _{load}

7.5 Input Pin $\overline{\text{ATTN}}$

The attention or $\overline{\text{ATTN}}$ signal is used to exit either Doze mode or Hibernate mode.

NOTE

Doze mode may also be exited via a tmr_cmp2 compare event. A transition event from high to low (assertion) on $\overline{\text{ATTN}}$ is required to exit either mode.

The $\overline{\text{ATTN}}$ assertion low event can also generate an interrupt. The interrupt status bit is attn_irq, IRQ_Status Register 24, Bit 10, and the interrupt mask bit is attn_mask, IRQ_Mask Register 05, Bit 15.

Index

A

- abbreviations
 - defined 1-vii
- accessible registers 2-1
- acoma doze 4-4
- acronyms
 - defined 1-vii
- active states 4-1
- ATTN
 - doze mode exit 7-6
 - input pin 7-6

C

- CCA
 - functionality 4-8
 - operational sequence 4-9
- CCA mode
 - register fields 4-8
- clear channel assesment 4-8
 - reported power 4-9
- CLKO 7-4
 - drive strength 7-6
 - enable/disable 7-4
 - enable 7-4
 - enable in doze 7-5
 - low frequency start 7-5
 - setting frequency 7-4
- conventions 1-vii
- crystal
 - requirements 7-3
 - trim operation 7-3
 - trim settings 7-3
- crystal oscillator
 - external circuit 7-3
 - pins 7-2
- current time
 - defined 5-2
 - latching 5-3

E

- energy detect 4-10
 - operational sequence 4-10
 - reported 4-10
- entering acoma mode 4-4
- entering doze mode 4-3
- entering hibernate mode 4-3
- event timer
 - base 5-1
 - block diagram 5-1
 - comparators 5-3
 - components 5-1
 - functions 5-1

- intended usage 5-4
- prescaler settings 5-2
- preventing false match 5-4
- reading current time 5-2
- setting compare values 5-4
- setting current time 5-2
- timestamp latching 5-3

- exiting doze mode
 - procedure 5-5
- external gain
 - compensation 2-7

F

- Frac-N 4-11
- frequency
 - operating 4-11

G

- generating time based interrupts
 - procedure 5-5

GPIO

- direction 7-2
- drive strength 7-2
- features 7-2
- input state read 7-2
- output value 7-2

- GPIO note 7-2

I

- initiating transceiver event (stream mode)
 - procedure 5-6

interrupts

- flow 6-1
- generating time based 5-5
- IRQ 6-2
- sources 6-1

IRQ

- drive strength 6-2
- pullup 6-2
- signal 6-2

- IRQ output drive 6-2

- IRQ programming 6-2

L

- link quality 4-11
 - reported 4-10
- local oscillators 4-11
- low current operation 4-1
- low power states 4-1

M

- mandatory register initialization 2-1

MISO

- drive strength 3-2
- off impedance 3-2

modes

- active 4-4
- clear channel assesment 4-8
- doze 4-1, 4-3
- exiting doze 5-5
- hibernate 4-1, 4-3
- idle 4-1, 4-4
- low power 4-3
- off 4-1, 4-3
- receive 4-1, 4-5
- summary 4-1
- transition times 4-1
- transmit 4-1

N

- normal doze 4-3

O

- out of lock 4-12

P

- packet receive mode notes 4-6
- packet receive sequence 4-6
- packet transmit mode notes 4-7
- packet transmit sequence 4-7
- PLL out of lock 4-12
- power adjust
 - transmit 4-11

R

- reading current time
 - preferred procedure 5-3
 - registers 5-2
- register
 - 00 (Reset) 2-4
 - 01 (RX_Pkt_RAM) 2-4
 - 02 (TX_Pkt_RAM) 2-5
 - 03 (TX_Pkt_Ctl) 2-6
 - 04 (CCA_Thresh) 2-7
 - 05 (IRQ_Mask) 2-8
 - 06 (Control_A) 2-10
 - 07 (Control_B) 2-11
 - 09 (Control_C) 2-12
 - 0A (CLKO_Ctl) 2-13
 - 0B (GPIO_Dir) 2-14
 - 0C (GPIO_Data_Out) 2-16
 - 0F (LO1_Int_Div) 2-18
 - 10 (LO1_Num) 2-18
 - 12 (PA_Lvl) 2-20
 - 1B (Tmr_Cmp1_A) 2-21

- 1C (Tmr_Cmp1_B) 2-22
- 1D (Tmr_Cmp2_A) 2-23
- 1E (Tmr_Cmp2_B) 2-24
- 24 (IRQ_Status) 2-25
- 25 (RST_Ind) 2-27
- 26 (Current_Tiame_A) 2-28
- 27 (Current_Time_B) 2-29
- 28 (GPIO_Data_In) 2-30
- 2C (Chip_ID) 2-31
- 2D (RX_Status) 2-32
- 2E (Timestamp_A) 2-33
- 2F (Timestamp_B) 2-34
- hidden registers note 2-1
- overview 2-1
- reserved fields note 2-1
- SPI mapping table 2-2

reset

- indicator 7-1
- input pin 7-1
- software 7-1

reset conditions 7-1

RX packet RAM

- read access errors 3-9
- read access flow 3-8

S

serial peripheral interface 3-1

setting current time

- SPI registers 5-2

SPI

- burst operation 3-3
- clock format note 3-2
- data format 3-6
- defined 3-1
- header and payload definition 3-5
- MISO 3-2
- MOSI 3-2
- operation 3-1
- packet RAM access 3-8
- pins 3-1
- recursive RAM read 3-8
- recursive read 3-7
- recursive transactions 3-7
- recursive write 3-7
- reset 3-11
- singular transactions 3-4
- software reset 3-11
- SPICLK 3-2
- symbol flow 3-6
- symbol order vs. word bit order 3-6

state diagrams 4-1

stream mode

- timer_triggered events 5-6

T

timer

- compare fields 5-3
- disable bits 5-3
- interrupt masks 5-4
- status flags 5-4

tmr_cmp2

- exiting doze mode 5-5

transceiver events

- timer triggered 5-6

transceiver sequence field 4-5

transition control 4-4

TX packet RAM

- write access 3-9
- write access errors 3-10
- write access flow 3-10

W

warp capability 7-3

